|  |
| --- |
| Cricket Application |
| Microservice Specification Document |

|  |
| --- |
| Manish Tamta  7-22-2024 |

Table of Contents

[Overview 3](#_Toc172544247)

[Player Service 3](#_Toc172544248)

[Introduction 3](#_Toc172544249)

[Dependencies 3](#_Toc172544250)

[Technologies 3](#_Toc172544251)

[Testing 3](#_Toc172544252)

[Schema 3](#_Toc172544253)

[Endpoints 3](#_Toc172544254)

[Team service 7](#_Toc172544255)

[Introduction 7](#_Toc172544256)

[Dependencies 7](#_Toc172544257)

[Technologies 7](#_Toc172544258)

[Testing 7](#_Toc172544259)

[Schema 8](#_Toc172544260)

[Endpoints 8](#_Toc172544261)

[Match microservices. 13](#_Toc172544262)

[Introduction 13](#_Toc172544263)

[Dependencies 13](#_Toc172544264)

[Technologies 13](#_Toc172544265)

[Testing 13](#_Toc172544266)

[Schema 14](#_Toc172544267)

[Endpoints 15](#_Toc172544268)

[Commentary Service: 26](#_Toc172544269)

[Introduction 26](#_Toc172544270)

[Dependencies 26](#_Toc172544271)

[Technologies 26](#_Toc172544272)

[Testing 26](#_Toc172544273)

[Schema 27](#_Toc172544274)

[Endpoints 27](#_Toc172544275)

[Search Service 31](#_Toc172544276)

[Introduction 31](#_Toc172544277)

[Dependencies 31](#_Toc172544278)

[Technologies 31](#_Toc172544279)

[Testing 31](#_Toc172544280)

[Schema 31](#_Toc172544281)

[Endpoints 32](#_Toc172544282)

[Other Aspects 34](#_Toc172544283)

[Observability and Monitoring 34](#_Toc172544284)

[Scalability 34](#_Toc172544285)

## Overview

Cricket app microservices specifications document which involves detailing of each microservices, eg purpose, scope, schema, dependencies, technologies, testing and endpoints.

## Player Service

### Introduction

#### Purpose

* It is designed to manage and provide info about players.
* To maintain players data.

#### Scope

* Player information including name, dob, nationality etc.
* Search players.
* Api for third party integration (dream 11 etc.)
* Implementing data validation and error handling mechanisms.
* Scalability and performance, ensuring the service can handle expected traffic during peak hours.
* Ensuring data privacy and security for users accessing player’s data.

### Dependencies

* Match service (to get individual score, statistics etc.)
* DB system

### Technologies

* **Framework:** Spring boot can be used.
* **Db:** Relational database can be used.
* **ORM:** Hibernate can be used.
* **Version control:** GitHub can be used.
* **Documentation:** Confluence can be used

### Testing

* **Unit Testing**: To ensure each individual method functioning correctly. JUNIT can be used for Spring boot applications.
* **Functional Testing:** To ensure business logic and expected behaviors are correctly working. Tool like Postman can be used to test functionality.
* **Load Testing:** To ensure test if max no of expected requests can be handled. Tool like JMeter can be used for load testing.

### Schema

Player:

playerId:

firstname:

lastname:

dob:

nationality

role:

### Endpoints

#### createPlayer

**Description:** Add a player for a given input, name, dob, nationality, role etc.

**API Path**: POST /player

**Request**:

{

    "name": "Rishabh Pant",

    "dob": "1997-10-04",

    "nationality": "Indian",

    "role": [1,2]

}

**Response**:

{"message":   "Player added successfully"}

**Success code:** 201

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### getPlayerDetails

**Description:** Get player details for a given id

**API Path**: GET /player/{id}

**Request**: NA

**Response**:

{

    "id":”100”,

    "firstname": "Rishabh",

“lastname”: “Pant”,

    "dob": "19971004",

    "nationality": "Indian",

    "role": [1,2]

“roleNames”: [“WICKET\_KEEPER”,"BATTER"],

}

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication:** NA

**Authorization:** NA

**Encryption**: HTTPS to encrypt data in transit

#### updatePlayer

**Description:** Update player information

**API Path**: PUT /player/{id}

**Request**:

{

    "firstname": "Rishabh",

“lastname”: “Pant”,

    "dob": "19971004",

    "nationality": "Indian",

    "role": [1,2]

}

**Response:** {“message”, “Player updated successful”}

**Success code**:200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### deletePlayer

**Description:** delete player by id.

**API Path**: DELETE /player/{id}

**Request: NA**

**Response**: {"message": “Player deleted successful"}

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### searchPlayers

**Description:** Search players by given search string

**API Path**: GET /player/search?<query string>

**Query String**: name, dob, nationality, role

**Response**:

[

{

    "id":”100”,

    "name": "Rishabh Pant",

    "dob": "19971004",

    "nationality": "Indian",

    "roleNames": ["BATTER","WICKET\_KEEPER"],

“stats”: {

“matches”: 50,

“runs”:1000,

“over”:0,

“wicket”:40

}

},

…

]

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security:**

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

## Team service

### Introduction

#### Purpose

* It is designed to manage and provide info about the teams.
* To maintain team data.
* Manage association b/w team and players.

#### Scope

* Team management (create, update, delete, team information)
* Player management in team (add, remove)
* Team statistics e.g. wins, losses, draws.
* Api for third party integration (dream 11 etc.)
* Implementing data validation and error handling mechanisms.
* Scalability and performance, ensuring the service can handle expected traffic during peak hours.
* Ensuring data privacy and security for users accessing player’s data.

### Dependencies

* Match service
* Player service
* DB system

### Technologies

* **Framework**: Spring boot can be used.
* **Db**: Relational database can be used.
* **ORM**: Hibernate can be used.
* **Version control:** GitHub can be used.
* **Documentation:** Confluence can be used.

### Testing

* **Unit Testing**: To ensure each individual method functioning correctly. JUNIT can be used for Spring boot applications.
* **Functional Testing:** To ensure business logic and behaviors are correctly working. Tool like Postman can be used to test functionality.
* **Load Testing:** To ensure test if max no of expected requests can be handled. Tool like JMeter can be used for load testing.

### Schema

Team:

id:

name:

country:

captain:

players:

wins:

looses:

draws:

Team\_Player:

team\_id:

player\_id:

### Endpoints

#### createTeam

**Description:** Create team for given inputs: name, country, captain etc.

**API Path**: POST /team

**Request**:

{

    "name": "India-A",

    "country": "India",

    "captain": player\_id,

}

**Response**:

{"message":   "Team created successfully"}

**Success code:** 201

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### getTeam

**Description:** Get team details for a given id

**API Path**: GET /team/{id}

**Request**: NA

**Response**:

{

    "id":"100",

    "name": "India-A",

    "country": "Indian",

    "captain": {

                "id":"100",

                "name": "Subhman Gill",

                "dob": "19990908",

                "nationality": "Indian",

                "role": ["batter"]

                },

    "players": [{

                "id":"100",

                "name": "Subhman Gill",

                "dob": "19990908",

                "nationality": "Indian",

                "role": ["batter"]

                },

      …

            ],

}

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

#### updateTeam

**Description:** Update player information

**API Path**: PUT /team/{id}

**Request**:

{

    "name": "India-A",

    "country": "India",

    "captain": player\_id,

“players”: [ “id\_1”, “id2”]

   }

**Response**:

{“message":   "Team updated successfully"}

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### deleteTeam

**Description:** Remove team.

**API Path**: DELETE /team/{id}

**Request: NA**

**Response**: {"message":   "Team deleted successfully"}

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### addPlayerToTeam

**Description:** Add or associate specific player to defined team.

**API Path**: POST /team/{teamId}/players/{player\_id}

**Request**: NA

**Response**:

{“message”: “Player <player\_name> added to team <team\_name>”}

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### removePlayerFromTeam

**Description:** Remove specific player from the team.

**API Path**: DELETE /team/{teamId}/players/{playerId}

**Request: NA**

**Response**: {"message":   "Player <player\_name> successfully removed from team <team\_name>"}

**Success cod:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### searchTeam

**Description:** Search team by given search criteria.

**API Path**: GET /teams/search?<query string>

**Request:** NA

**Query String**: name, nationality, wins, looses, draws etc.

**Response**:

[

{

    "id":"100",

    "name": "India-A",

    "country": "Indian",

    "captain": {

                "id":"100",

                "name": "Subhman Gill",

                "dob": "19990908",

                "nationality": "Indian",

                "role": ["batter"]

                },

    "players": [{

                "id":"100",

                "name": "Subhman Gill",

                "dob": "19990908",

                "nationality": "Indian",

                "role": ["batter"]

                },

      …

            ],

    "wins": 100,

    "looses": 50,

    "draws": 5,

}

…

]

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

## Match microservices.

### Introduction

#### Purpose

* It is designed to handle all aspects of match-related data.
* Provide match schedules (team, venue, date Time)
* Provide real-time update (scores, wickets, overs)
* Provide historical match data, results.
* Players and teams’ performance (statistics).

#### Scope

* Live match update
* Match details (team, venue, match status)
* Search matches for upcoming or past matches.
* Real time scoreboard update and display.
* Past scoreboard.
* Api for third party integration (dream 11 etc)
* Implementing data validation and error handling mechanisms.
* Scalability and performance, ensuring the service can handle high traffic during peak hour.
* Ensuring data privacy and security for users accessing matches.

### Dependencies

* Player service: to integrate players data to match data.
* Team service: to display team details in a match.
* DB system

### Technologies

* **Framework**: Spring boot can be used.
* **Db**: Relational database can be used.
* **ORM**: Hibernate can be used.
* **Version control:** GitHub can be used.
* **Documentation:** Confluence can be used.

### Testing

* **Unit Testing**: To ensure each individual method functioning correctly. JUNIT can be used for Spring boot applications.
* **Functional Testing:** To ensure business logic and behavior are correctly working. Tool like Postman can be used to test functionality.
* **Load Testing:** To ensure test if max no of expected requests can be handled. Tool like JMeter can be used for load testing.

### Schema

Match:

matchId:

team1:

team2:

venue:

dateTime:

format: T20/ODI/TEST

status

Inning:

inning\_id:

match\_id:

team\_id:

inning\_number:

runs:

wickets:

overs:

wides:

no\_balls:

byes:

leg\_byes:

batters: score\_id

bowlers: score\_id

batters:

batter\_id:

inning\_id:

player\_id:

runs:

balls\_faced:

fours:

sixes:

out\_by:

out\_type:

run\_out\_by:

bowlers:

bowler\_id:

inning\_id:

player\_id:

overs:

wickets:

wides:

no\_balls:

runs\_given:

score:

id:

inning\_id:

over: (0.1,0.2,0.3 etc.)

runs: (in each ball how much run is scored)

runs\_by: player\_id (player scored runs or by extra)

run\_type: (score by batsman/ extras (bye, leg\_bye, no ball, wide\_ball etc.)

wicket:

wicket\_type: (run out/balled/ hit-wicket etc)

wicket\_by: player\_id

no\_ball:

wide\_ball:

### Endpoints

#### createMatch

**Description:** create a match for a given inputs - schedule, teams, venue, format, status.

**API Path**: POST /match

**Request**:

{

    "team1": {

        "name": "India",

        "players": ["Ind1", "Ind2", "Ind3"],

    },

    "team2": {

        "name": "England",

        "players": ["Eng1", "Eng2", "Eng3"],

    },

    "status": "scheduled",

    "venue": "Wankhede Mumbai",

    "date\_time": "20240704 14:00",

    "format": "ODI"

}

**Response**:

{"message":”Match created successfully”}

**Success code:** 201

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### getMatchDetails

**Description:** get the match details of a given matchId.

**API Path**: GET /match/{id}

**Request:** NA

**Response**:

{

    "match\_id":"100",

    "team1": {

        "name": "India",

        "players": ["Ind1", "Ind2", "Ind3"],

        "score": {

            "runs":300,

            "wicket": 6,

            "overs":50

        }

    },

    "team2": {

        "name": "England",

        "players": ["Eng1", "Eng2", "Eng3"],

        "score": {

            "runs":220,

            "wicket": 7,

            "overs":45.0

        }

    },

    "status": "Inprogress",

    "venue": "Wankhede Mumbai",

    "date\_time": "20240704 14:00",

    "format": "ODI"

}

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

#### updateMatch

**Description:** update match details by given inputs.

**API Path**: PUT /match /{id}

**Request:**

{

    "team1": {

        "name": "India",

        "players": ["Ind1", "Ind2", "Ind3"],

    },

    "team2": {

        "name": "England",

        "players": ["Eng1", "Eng2", "Eng3"],

    },

    "status": "completed",

    "venue": "Wankhede Mumbai",

    "date\_time": "20240704 14:00",

    "format": "ODI"

},

**Response**: {"message":   "Match updated successfully"}

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### deleteMatch

**Description:** delete match by id.

**API Path**: DELETE /match/{id}

**Request/Response**: {"message":   "Match deleted successfully"}

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### searchMatches

**Description:** Search matches by given search string

**API Path**: GET /match/search?<query string>

**Request:** NA

**Query String**: match\_id, team\_name, status, venue, datetime\_range, format

**Response**:

[

{

    "match\_id":"100",

    "team1": {

        "name": "India",

        "players": ["Ind1", "Ind2", "Ind3"],

        "score": {

            "runs":300,

            "wicket": 6,

            "overs":50

        }

    },

    "team2": {

        "name": "England",

        "players": ["Eng1", "Eng2", "Eng3"],

        "score": {

            "runs":220,

            "wicket": 7,

            "overs":45.0

        }

    },

    "status": "Inprogress",

    "venue": "Wankhede Mumbai",

    "date\_time": "20240704 14:00",

    "format": "ODI"

},

…

]

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

#### createInning

**Description:** Create a scoreboard for a given match id

**API Path**: POST /scoreboards/inning

**Request**:

{

    "match\_id":"100",

    "team\_id":"100",

    "inning\_number":"1",

    "runs":"",

    "wickets":"",

    "overs":"",

    "wides":"",

    "no\_balls":"",

    "byes":"",

    "leg\_byes":"",

    "overthrough":"",

    "batters":[{

                "batter\_id":"",

                "inning\_id":"",

                "player\_id":"",

                "runs":"",

                "balls\_faced":"",

                "fours":"",

                "sixes":"",

                "out\_by":"",

                "out\_type":"",

                "run\_out\_by":""

            }],

    "bowlers":[ {

                "bowler\_id":"",

                "inning\_id":"",

                "player\_id":"",

                "overs":"",

                "wickets":"",

                "wides":"",

                "no\_balls":"",

                "runs\_given":""

            }]

    }

**Response**:

{"message":”Inning created successfully”}

**Success code:** 201

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### updateInning

**Description:** Update inning with the latest information

**API Path**: PUT /scoreboard/inning/{inning\_id}

**Request**:

{

    "match\_id":"100",

    "team\_id":"100",

    "inning\_number":"1",

    "runs":"150",

    "wickets":"5",

    "overs":"18.0",

    "wides":"5",

    "no\_balls":"3",

    "byes":"2",

    "leg\_byes":"3",

    "overthrough":"5",

    "batters":{

                "batter\_id":"",

                "inning\_id":"",

                "player\_id":"",

                "runs":"",

                "balls\_faced":"",

                "fours":"",

                "sixes":"",

                "out\_by":"",

                "out\_type":"",

                "run\_out\_by":""

            },

    "bowlers": {

                "bowler\_id":"",

                "inning\_id":"",

                "player\_id":"",

                "overs":"",

                "wickets":"",

                "wides":"",

                "no\_balls":"",

                "runs\_given":""

            }

    }

**Response**: {"message":   "Inning updated successfully"}

**Success code**: 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### createScoreBoard

**Description:** Create a scoreboard for a given match id

**API Path**: POST /scoreboards/

**Request:**

{

"inning\_id":"100",

"over":"1.1",

"runs":"4",

"runs\_by":"player\_id",

"run\_type":"",

"wicket":"",

"wicket\_type":"",

"wicket\_by":"",

"no\_ball":"",

"wide\_ball":"

}

**Response**:

{"message":” Scoreboard created successfully”}

**Success code:** 201

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### UpdateScoreBoard

**Description:** Create a scoreboard for a given match id

**API Path**: PUT /scoreboards/{score\_id}

**Request:**

{

        "over":"1.1",

        "runs":"4",

        "runs\_by":"player\_id",

        "run\_type":"",

        "wicket":"",

        "wicket\_type":"",

        "wicket\_by":"",

        "no\_ball":"",

        "wide\_ball":""

}

**Response**:

{"message":   "Scoreboard updated successfully"}

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### getScoreBoard

**Description:** Get scoreboard by id

**API Path**: GET /scoreboard/{scoreboard\_id}

**Request**: NA

**Response**:

{

"scoreboard\_id": "100",

"match\_id": "100",

"overs": "15",

"wickets": "3",

"Result" : ["Winner":"100"],

Innings: [{

    "match\_id":"100",

    "team\_id":"100",

    "inning\_number":"1",

    "runs":"150",

    "wickets":"5",

    "overs":"18.0",

    "wides":"5",

    "no\_balls":"3",

    "byes":"2",

    "leg\_byes":"3",

    "overthrough":"5",

    "batters":[{

                "batter\_id":"",

                "inning\_id":"",

                "player\_id":"",

                "runs":"",

                "balls\_faced":"",

                "fours":"",

                "sixes":"",

                "out\_by":"",

                "out\_type":"",

                "run\_out\_by":""

            }],

    "bowlers":[ {

                "bowler\_id":"",

                "inning\_id":"",

                "player\_id":"",

                "overs":"",

                "wickets":"",

                "wides":"",

                "no\_balls":"",

                "runs\_given":""

            }]

    }

]

}

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: NA

**Authorization**: NA

**Encryption**: HTTPS to encrypt data in transit

#### getPlayerBattingPerformance

**Description:**  Get Player performance in a match

**API Path**: GET /scoreboard/{match\_id}/player/{player\_id}?type=batting

**Request**: NA

**Response**:

{

                "inning\_id":"",

                "player\_id":"",

                "runs":"",

                "balls\_faced":"",

                "fours":"",

                "sixes":"",

                "out\_by":"",

                "out\_type":"",

                "run\_out\_by":"",

“strike\_rate”:””,

}

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

#### getPlayerBowlingPerformance

**Description:**  Get Player performance in a match

**API Path**: GET /scoreboard/{match\_id}/player/{player\_id}?type=bowling

**Request**: NA

**Response**:

{

                "match\_id":"",

                "inning\_id":"",

                "player\_id":"",

                "overs":"",

                "wickets":"",

                "wides":"",

                "no\_balls":"",

                "runs\_given":"",

“economy”:””

            }

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**:

**Authorization**:

**Encryption**: HTTPS to encrypt data in transit

#### getTeamPerformance

**Description:**  Get Team performance in a match

**EndPoints**: GET /scoreboard/{match\_id}/team/{team\_id}

**Request**: NA

**Response**:

{

    "team\_id": "100",

    "match\_id": "100",

    "score": "100",

    "wickets": "5",

    "overs": "20",

    "players": {

        {

            "player\_id":"100",

            "runs": "100",

            "fours": "2",

            "sixes": "1",

            "overs": "15",

            "wickets": "3",

        }

    }

}

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

## Commentary Service:

### Introduction

#### Purpose

* Provide ball by ball commentary.
* Provide past matches commentary.
* Provide milestones, specific event related commentary eg wicket, sixes, fours etc.

#### Scope

* Provide ball by ball detailed description.
* Provide past matches commentary.
* Provide commentary related to specific event eg wicket, sixes, fours
* Api for third party integration (dream 11 etc)
* Implementing data validation and error handling mechanisms, to ensure data integrity and availability.
* Scalability and performance, ensuring the service can handle expected traffic during peak hours.

### Dependencies

* Match service.
* DB system

### Technologies

* **Framework**: Spring boot can be used.
* **Db**: Relational database can be used.
* **ORM**: Hibernate can be used.
* **Version control**: GitHub can be used.
* **Documentation**: Confluence can be used.

### Testing

* **Unit Testing**: To ensure each individual method functioning correctly. JUNIT can be used for Spring boot applications.
* **Functional Testing:** To ensure business logic and behavior are correctly working. Tool like Postman can be used to test functionality.
* **Load Testing -** To ensure test if max no of expected requests can be handled. Tool like JMeter can be used for load testing.

### Schema

Commentary:

id:

over:

match:

description:

event:

CommentryEvent

Id:

commentary\_id:

over:

event:

### Endpoints

#### createCommentary

**Description:** Create commentary for a given match and ball

**API Path**: POST /commentary/

**Request**:

{

        "match": "100",

        "over":1.2,

        "description": "player cought and bold",

        "event": ["OUT"]

}

**Response**: {"message":   "Commentary created successfully"}

**Success code:** 201

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### getCommentary

**Description:** get commentary of a match

**API Path**: GET /commentary/{match\_id}

**Request**: NA

**Response**:

{

    "match\_id": "100",

    "commentary": [

        {

            "over": 1.1,

            "description": " 1 run scored",

            "event": []

        },

        {

            "over": 1.2,

            "description": "Player out",

            "event": ["OUT"]

        },

…

    ]

}

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

#### updateCommentary

**Description:** Update commentary for a given match and ball

**API Path**: PUT /commentary/

**Request**:

{

        "match": "100",

        "over":1.2,

        "description": "1 run scored",

        "event": [""]

}

**Response:** {"message":   "Commentary updated successfully"}

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication**: JWT tokens for secure API access.

**Authorization**: Role-based access control

**Encryption**: HTTPS to encrypt data in transit

#### getEventCommentary

**Description:** get commentary of a match

**API Path**: GET /commentary/event/{event}

**Request**: NA

**Response**:

{

    "match\_id": "100",

    "commentary": [

        {

            "over": 1.1,

            "description": "Run out",

            "event": [“OUT”]

        },

        {

            "over": 5.2,

            "description": "Player out",

            "event": ["OUT"]

        },

…

    ]

}

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

#### getOverCommentary

**Description:** get commentary of an over in a match

**EndPoints**: GET /commentary/{match\_id}/{over}

**Request**: NA

**Response**:

{

    "match\_id": "100",

    "commentary": [

        {

            "over": 1.1,

            "description": " 1 run scored",

            "event": []

        },

        {

            "over": 1.2,

            "description": "Four",

            "event": ["Four"]

        },

{

            "over": 1.3,

            "description": "Player out",

            "event": ["OUT"]

        },

{

            "over": 1.4,

            "description": "Six scored",

            "event": ["Sixed"]

        },

{

            "over": 1.5,

            "description": "Player run out",

            "event": ["OUT"]

        },

{

            "over": 1.6,

            "description": "No run",

            "event": []

        },

    ]

}

**Success code:** 200

**Error code:** 400, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

## Search Service

### Introduction

#### Purpose

* It is designed to handle indexing and query data efficiently.
* Indexing content (player, teams, matches)
* Handling search queries
* Centralized search needs, scale independently, searching different data source, easier maintenance.

#### Scope

* Find relevant content of players, teams, and matches.
* Implementing data validation and error handling mechanisms.
* Scalability and performance, ensuring the service can handle high traffic during peak hour.
* Ensuring data privacy and security for users accessing matches.

### Dependencies

* Player service
* Team service
* Match service.
* DB system

### Technologies

* **Framework**: Spring boot can be used.
* **Db**: Relational database can be used.
* **ORM**: Hibernate can be used.
* **Version control:** GitHub can be used.
* **Documentation:** Confluence can be used.

### Testing

* **Unit Testing**: To ensure each individual method functioning correctly. JUNIT can be used for Spring boot applications.
* **Functional Testing:** To ensure business logic and behavior are correctly working. Tool like Postman can be used to test functionality.
* **Load Testing -** To ensure test if max no of expected requests can be handled. Tool like JMeter can be used for load testing.

### Schema

Match:

matchId:

team1:

team2:

venue:

dateTime:

format: T20/ODI/TEST

status

Player:

playerId:

name:

dob:

nationality

role:

Team:

id:

name:

country:

captain:

players:

wins:

looses:

draws:

Team\_Player:

team\_id:

player\_id:

### Endpoints

#### searchMatches

**Description:** search matches for given query string.

**Endpoints**: GET /search/matches?q=<query-string>

**Request**: NA

**Query String**: team\_name, status, venue, datetime\_range, format

**Response**:

[

{

    "match\_id":"100",

    "team1": {

        "name": "India",

        "players": ["Ind1", "Ind2", "Ind3"],

        "score": {

            "runs":300,

            "wicket": 6,

            "overs":50

        }

    },

    "team2": {

        "name": "England",

        "players": ["Eng1", "Eng2", "Eng3"],

        "score": {

            "runs":220,

            "wicket": 7,

            "overs":45.0

        }

    },

    "status": "Inprogress",

    "venue": "Wankhede Mumbai",

    "date\_time": "20240704 14:00",

    "format": "ODI"

},

…

]

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

#### searchTeams

**Description:** search matches for given query string.

**Endpoints**: GET /search/teams?q=<query-string>

**Request**: NA

**Query String**: name, country, wins, looses, draws etc.

**Response**:

[

{

    "id":"100",

    "name": "India-A",

    "country": "Indian",

    "captain": {

                "id":"100",

                "name": "Subhman Gill",

                "dob": "19990908",

                "nationality": "Indian",

                "role": ["batter"]

                },

    "players": [{

                "id":"100",

                "name": "Subhman Gill",

                "dob": "19990908",

                "nationality": "Indian",

                "role": ["batter"]

                },

      …

            ],

    "wins": 100,

    "looses": 50,

    "draws": 5,

}

…

]

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

#### searchPlayers

**Description:** search matches for given query string.

**Endpoints**: GET /search/players?q=<query-string>

**Request**: NA

**Query String:** player\_name, nationality, scores, matches runs, wickets etc.

**Response:**

[

{

    "id":”100”,

    "name": "Rishabh Pant",

    "dob": "19971004",

    "nationality": "Indian",

    "role": ["batter","wicketkeeper"],

“stats”: {

“matches”: 50,

“runs”:1000,

“over”:0,

“wicket”:40

}

},

…

]

**Success code:** 200

**Error code:** 400, 401, 403, 500

**Error Response:**

{"error":   "Error\_Msg"}

**Security**:

**Authentication: NA**

**Authorization: NA**

**Encryption**: HTTPS to encrypt data in transit

## Other Aspects

### Observability and Monitoring

To ensure system is gaining insights into the behavior and performance of a system.

For centralized logging tool like Loki (log aggregation) can be used and for monitoring Prometheus (metrics collection), Grafana (visualization) can be used.

### Scalability

To ensure service is scalable for different load and load can be distributed.

Load balancer and k8s can be used.