|  |  |
| --- | --- |
| CRIC-APP Specification | Specifications Document  Hemant Dhamal    Created On: 29-07-2024  Revised On: NA  Version No: 1.0 |

Contents

[Introduction 5](#_Toc171634052)

[1 User Management Service 5](#_Toc171634053)

[1.1 Overview 5](#_Toc171634054)

[1.2 Data model specification 5](#_Toc171634055)

[1.3 API Endpoints 5](#_Toc171634056)

[1.4 Dependencies 9](#_Toc171634057)

[1.5 Scalability 9](#_Toc171634058)

[2 Match Details Service 9](#_Toc171634059)

[2.1 Overview 9](#_Toc171634060)

[2.2 Data model specification 10](#_Toc171634061)

[2.3 API Endpoints 10](#_Toc171634062)

[2.4 Dependencies 15](#_Toc171634063)

[2.5 Scalability 15](#_Toc171634064)

[3 Live Score Service 15](#_Toc171634065)

[3.1 Overview 15](#_Toc171634066)

[3.2 Data model specification 15](#_Toc171634067)

[3.3 API Endpoints 16](#_Toc171634068)

[3.4 Dependencies 20](#_Toc171634069)

[3.5 Scalability 20](#_Toc171634070)

[4 Team Management Service 21](#_Toc171634071)

[4.1 Overview 21](#_Toc171634072)

[4.2 Data model specification 21](#_Toc171634073)

[4.3 API Endpoints 22](#_Toc171634074)

[4.4 Dependencies 27](#_Toc171634075)

[4.5 Scalability 27](#_Toc171634076)

[5 Player Statistics Service 27](#_Toc171634077)

[5.1 Overview 27](#_Toc171634078)

[5.2 Data model specification 27](#_Toc171634079)

[5.3 API Endpoints 28](#_Toc171634080)

[5.4 Dependencies 34](#_Toc171634081)

[5.5 Scalability 34](#_Toc171634082)

# Introduction

A cricket score website like ESPNcricinfo typically provides real-time match updates, statistic, news, and other cricket-related content. To build such a system using a microservices architecture, break down the functionality into smaller, independent services that can be developed, deployed, and scaled separately.

Following are 5 Microservices for a cricket score website:

* User management service
* Match details service
* Live score service
* Team management service
* Player statistic service

# Microservices

# User Management Service

## Overview

* User management service manages user accounts, authentication, and authorization. Provides APIs for user registration, login, and profile management. Handles user preferences and settings.
* In this application, there are two types of users – Admin, User.
* Admin – Can access and update data.
* User – Can only read the data created by Admin user.
* In case of authentication, we can use token-based approach, session-based approach.

## Data model specification

User

{

"userId": "<UUID>",

"userName": "<String> ",

"password": "<String>",

“isAdmin”: “<Boolean>”

}

## API Endpoints

### Login

Authenticate the user to allow access to specific functions in the system based on assigned permissions.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| POST | api/v1/auth/login | no |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| username | Yes | Body | string | Login id of user |
| password | Yes | Body | string | Password of user |

##### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | Json | User logged in successfully | {  "message": "User logged in successfully"  } |
| 404 | Json | User failed to login | {  "error": “User not found”  } |

#### Implementation Details

##### Method Name: AuthController.login()

1. Authentication – Not required
2. Validation – Required

### Logout

Logs out the current logged in user.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| POST | api/v1/auth/logout | yes |

##### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | User logged out successfully | {  Message: “<Logged out successfully >”  } |
|  |  |  |  |

#### Implementation Details

##### Method Name: AuthController.logout()

1. Authentication - Required
2. Validation – Required

### Validate Username

Validates entered username during user signup and create user to avoid duplication of username.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| POST | api/v1/signup/validateUsername | No |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| value | Yes | body | string | Enter username |

##### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | Json | Username validation successful | {  “error”: false  } |
| 200 | Json | Username is already registered | {  “error”: true,  "message": "Username is already exist"  } |
| 404 | Json | Invalid request parameters | {  "message": "Username not found"  } |

#### Implementation Details

##### Method Name: SignupController.validateUsername()

1. Authentication – Not required
2. Validation - Required

### Create new user

Register new user.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| POST | api/v1/users | yes | Admin |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| role | Yes | body | string | Admin role |
| firstName | Yes | body | string | First name of the user |
| lastName | Yes | body | string | Last name of the user |
| email | Yes | body | string | Email Address of the user |
| phoneNumber | Yes | body | string | Phone Number of the user |
| username | Yes | body | string | Display name for the user |
| password | Yes | body | string | Password of the user |

##### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 201 | Json | Successfully registered user | {  “error”: false,  “message”: “Successfully registered user”  } |
| 400 | Json | User Registration unsuccessful | {  “error”: true,  “message”: <error message>  } |
| 404 | Json | Invalid request parameters | {  “message”: “User not found”  } |

#### Implementation Details

##### Method Name: signUpController.registerUser()

1. Authentication - Required
2. Validation - Required

### Fetch User Details

Fetches user information based on user id.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | api/v1/users/{username} | yes |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| username | Yes | path | string | Username |

##### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | Json | Fetched user details successfully | {  “user”: <JSON object containing user details>  “error”: false  } |
| 404 | Json | User not found | {  "message": "User not found "  } |

#### Implementation Details

##### Method Name: UsersController.fetchUser()

1. Authentication – Not required
2. Validation - Required

### User Update

Update user’s info like email, phone number and activate or deactivate user. Only user users can perform this action.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | User Role |
| PUT | api/v1/users/{username} | yes | User, Admin |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| username | Yes | req param | string | User user name |
| firstName | Yes | body | string | First name of the user |
| lastName | Yes | body | string | Last name of the user |
| email | Yes | body | string | Email Address of the user |
| phoneNumber | Yes | body | string | Phone Number of the user |
| role | Yes | body | string | Role assigned to the user |
| username | Yes | body | string | Display name for the user |
| Status | Yes | Body | String | Set status active or inactive |

##### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | Json | User Updated successfully | {  “error”: false,  “message”: “User updated successful.”  } |
| 404 | Json | User not found | {  “message”: “User not found”  } |

#### Implementation Details

##### Method Name: UsersController.updateUser()

1. Authentication - Required
2. Validation - Required

### Change Password

Change password of application users. This api is used to change the password on first time login after password reset or after user creation by user user

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | User Role |
| PUT | api/v1/update-password/{username} | Yes | User, Admin |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| username | Yes | body | string | Username |
| username | Yes | body | string | Display name for the user |
| oldPassword | Yes | body | string | Old password of the user |
| newPassword | Yes | body | string | New password of the user |

##### Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | Json | Password updated successfully | {  “error”: false,  “message”: “Password updated successfully”  } |
| 400 | Json | Password update unsuccessful | {  “error”: true,  “message”: “Password update unsuccessful”  } |

#### Implementation Details

##### Method Name: UsersController.updatePassword()

1. Authentication - Required
2. Validation - Required

## Dependencies

This is an independent service.

# Match Details Service

## Overview

* This service manages details about cricket matches (teams, players, venue, start time, etc.).
* Provides APIs for team lineups, and match status.
* Handles CRUD operations for match data.
* Integrates with external data sources for match information.

## Data model specification

Match

{

"matchId": "<UUID>",

"leagueName": "<String> ",

"teamName1": "<String>",

"teamName2": "<String>",

“team1”: “<Json>”,

“team2”: “<Json>”,

"venue": "<String> ",

"dateTime": "<Date>",

"matchFormat": "<String>"

}

Statistic

{

“matchId”: “<UUID>”,

“mom”: “<String>”,

“winner”: “<String>”

“highestRuns”: {“PlayerName”: “<String>”,

“Runs”: “<Number>”

},

“highestWickets”: {“PlayerName”: “<String>”,

“Wickets”: “<Number>”

},

“bestEconomy”: {“playerName”: name,

“bestEconomy”: economy},

“bestStrikeRate”: {“playerName”: name,

“bestStrikeRate”: strikeRate},

“totalSixes”: sixes,

“totalFours”: fours

}

## API Endpoints

### Create match information

Create a match details.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| POST | /api/v1/matches/ | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| leagueName | Yes | body | string | League name |
| teamName1 | Yes | body | string | Team name |
| team1 | Yes | body | string | Team1 |
| teamName2 | Yes | body | string | Team name |
| Team2 | Yes | body | string | Team2 |
| venue | yes | body | string | Ground location |
| dateTime | Yes | body | localDateTime | Date and time of match |
| matchFormat | Yes | body | string | Set match format |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning match data | {  “error”: false,  “matchId”: matchId,  “message”: “Created match details”  } |
| 404 | json | Error occurred while fetching data | {  “error”: true,  “message”: error.message  } |

#### Implementation Details

##### Method Name: MatchController.createMatchDetails()

1. Authentication - Required
2. Validation - Required

### Update match information

Update a match details.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/matches | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| matchId | Yes | body | string | Match Id |
| leagueName | Yes | body | string | League name |
| teamName1 | Yes | body | string | Team name |
| team1 | Yes | body | string | Team1 |
| teamName2 | Yes | body | string | Team name |
| Team2 | Yes | body | string | Team2 |
| venue | yes | body | string | Ground location |
| dateTime | Yes | body | localDateTime | Date and time of match |
| matchFormat | Yes | body | string | Set match format |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating match details | {  “error”: false,  “message”: “Updated match details”  } |
| 404 | json | Error occurred while fetching data | {  “error”: true,  “message”: error.message  } |

#### Implementation Details

##### Method Name: MatchController.updateMatchDetails()

1. Authentication - Required
2. Validation - Required

### Get match information

Fetches a match information.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/matches/{matchId} | No |

##### Input Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | in | Description |
| matchId | yes | path | Match Id |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning match details | {  “leagueName”: leagueName,  “teamName1”: teamName1,  “team1”: {[{“name”:”playerName”,”role”:”batsman”},  {“name”:”playerName”,”role”:”all-rounder”},  {“name”:”playerName”,”role”:”bowler”}  ]},  “teamName2”: teamName2,  “team2”: {[{“name”:”playerName”,”role”:”batsman”},  {“name”:”playerName”,”role”:”all-rounder”},  {“name”:”playerName”,”role”:”bowler”}  ]},  “venue”: venue,  “grounddetails”: grounddetails,  “dateAndTime”: dateAndTime,  “matchFormat”: matchFormat  } |
| 404 | json | Error occurred while fetching match details | {  “error”: true,  “message”: error.message  } |

#### Implementation Details

##### Method Name: MatchController.fetchMatchDetails()

1. Authentication – Not required
2. Validation – Required

### Update statistic

Update statistic.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/matches/{matchId}/statistic | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| matchId | Yes | path | int | Match Id |
| mom | Yes | body | string | Man of the match award winner |
| winner | Yes | body | string | Winning team |
| highestRuns | Yes | body | json | Highest runs scored by player |
| highestWickets | Yes | body | json | Highest wickets taken by player |
| bestEconomy | Yes | body | json | Player who bowled with best economy |
| bestStrikeRate | Yes | body | json | Player who achieved best Strike Rate |
| totalSixes | Yes | body | int | Total sixes by both teams |
| totalFours | Yes | body | int | Total fours by both teams |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating statistic | {  “error”: false,  “message”: “Updated statistic  } |
| 404 | json | Error occurred while updating data | {  “error”: true,  “message”: error.message  } |

#### Implementation Details

##### Method Name: MatchController.updateStatistics()

1. Authentication - Required
2. Validation - Required

### Get statistic

Fetches statistic.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/matches/{matchId}/statistic | No |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| matchId | Yes | body | uuid | Match Id |
| mom | Yes | body | string | Man of the match award winner |
| winner | Yes | body | string | Winning team |
| highestRuns | Yes | body | json | Highest runs scored by player |
| highestWickets | Yes | body | json | Highest wickets taken by player |
| bestEconomy | Yes | body | json | Player who bowled with best economy |
| bestStrikeRate | Yes | body | json | Player who achieved best strike rate |
| totalSixes | Yes | body | int | Total sixes by both teams |
| totalFours | Yes | body | int | Total fours by both teams |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning statistic | {  “matchId”: matchId,  “mom”: mom,  “winner”: winner,  “highestRuns”: {“playerName”: name,  “runs”:runs},  “highestWickets”: {“playerName”: name,  “wickets”: wickets},  “bestEconomy”: {“playerName”: name,  “bestEconomy”: economy},  “bestStrikeRate”: {“playerName”: name,  “bestStrikeRate”: strikeRate},  “totalSixes”: sixes,  “totalFours”: fours  } |
| 404 | json | Error occurred while fetching commentary data | {  “error”: true,  “message”: error.message  } |

#### Implementation Details

##### Method Name: MatchController.fetchStatistics()

1. Authentication – Not required
2. Validation - Required

## Dependencies

This service is loosely depending upon,

1. User Management Service
2. Player Statistic Service
3. Live Score Service

# Live Score Service

## Overview

* Provides real-time score updates for ongoing matches.
* Exposes APIs for current score, wickets, overs, run rate, etc.
* Implements WebSocket for push notifications.
* Handles data input from scorers or automated systems.

## Data model specification

Score

{

“matchId”: “<UUID>”,

“toss”: <String>,

“playingTeam1”: “<Json>”,

“playingTeam2”: “<Json>”,

“inning1batting”: “<Json>”,

“inning1bowling”: “<Json>”,

“inning2batting”: “<Json>”,

“inning2bowling”: “<Json>”,

"totalRuns": "<Number>",

"totalWickets": "<Number> ",

"playingBatsmanName1": "<String>",

"playingBatsmanName2": "<String>",

"playingBowlerName": "<String> ",

“perBallEvent”:”<Json>”

"overCount": "<Number>"

“extras”: “<Json>”,

“fow”:”<Json>”

}

Commentary

{

“over”: “<Number>”,

“event”: ”<String>”,

“commentary”: “<String>”

}

## API Endpoints

### Create match score

Create a new match score.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| POST | /api/v1/scores | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| matchId | Yes | body |  | Match Id |
| totalRuns | Yes | body | int | Team’s total runs |
| totalWickets | Yes | body | int | Team’s total wicket |
| batsmanName1 | Yes | body | string | Batsman name1 |
| batsmanScore1 | Yes | body | int | Batsman score1 |
| batsmanName2 | Yes | body | string | Batsman name2 |
| batsmanScore2 | Yes | body | Int | Batsman score2 |
| bowlerName | Yes | body | string | Bowler name |
| overCount | Yes | body | float | Overs count |
| economy | Yes | body | float | Economy |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Creating match score details | {  “error”: false,  “scoreId”: scoreId,  “message”: “Created match score details”  } |
| 401 | json | Error occurred while fetching data | {  “message”: “Unauthorized access”  } |

#### Implementation Details

##### Method Name: ScoreController.createMatchScoreDetails()

1. Authentication - Required
2. Validation - Required

### Update match score

Update a match details.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/scores/{matchScoreId} | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| scoreId | Yes | path | string | Match score Id |
| totalRuns | Yes | body | int | Team’s total runs |
| totalWickets | Yes | body | int | Team’s total wicket |
| batsmanName1 | Yes | body | string | Batsman name1 |
| batsmanScore1 | Yes | body | int | Batsman score1 |
| batsmanName2 | Yes | body | string | Batsman name2 |
| batsmanScore2 | Yes | body | Int | Batsman score2 |
| bowlerName | Yes | body | string | Bowler name |
| overCount | Yes | body | float | Overs count |
| economy | Yes | body | float | Economy |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating match score deatils | {  “error”: false,  “message”: “Updated match score details”  } |
| 404 | json | Error occurred while updating match score | {  “message”: “Match score not found”  } |

#### Implementation Details

##### Method Name: ScoreController.updateMatchScoreDetails()

1. Authentication - Required
2. Validation - Required

### Get match score

Fetches a match information.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/scores/{matchScoreId} | No |

##### Input Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | in | Description |
| matchScoreId | Yes | path | Match Id |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning match score details | {  “scoreId”: scored,  “totalRuns”: totalRuns,  “totalWickets”: totalWickets,  “batsmanName1”: batsmanName1,  “batsmanScore1”: batsmanScore1,  “batsmanName2”: batsmanName2,  “batsmanScore2”: batsmanScore2,  “bowlerName”: bowlerName,  “overDetails”: overDetails,  “overCount”: overcount,  “economy”: economy  } |
| 404 | json | Error occurred while fetching score details | {  “message”: “Match score not found”  } |

#### Implementation Details

##### Method Name: ScoreController.fetchMatchDetails()

1. Authentication – Not required
2. Validation - Required

### Update commentary details

Create a match details.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/matches/{matchId}/commentary | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| scoreId | Yes | Path | UUID | Match score Id |
| commentary | Yes | body | Json | Ball by ball Detailed commentary |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating commentary details | {  “error”: false,  “message”: “Updated commentary details”  } |
| 404 | json | Error occurred while updating commentary details | {  “message”: “Commentary details not found”  } |

#### Implementation Details

##### Method Name: ScoreController.updateCommentaryDetails()

1. Authentication - Required
2. Validation - Required

### Get commentary details

Fetches a match information.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/matches/{matchId}/commentary | No |

##### Input Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | in | Description |
| matchId | yes | path | Match Id |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating commentary | {  “commentaryDetails”: Ball-by-ball commentary details  } |
| 404 | json | Error occurred while fetching commentary | {  “message”: “Commentary details not found”  } |

#### Implementation Details

##### Method Name: ScoreController.fetchCommentaryDetails()

1. Authentication - Required
2. Validation - Required

### List all match score

Fetches a match information.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/scores | No |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning list of match score details | {matchScores: [  “totalRuns”: totalRuns,  “totalWickets”: totalWickets,  “batsmanName1”: batsmanName1,  “batsmanScore1”: batsmanScore1,  “batsmanName2”: batsmanName2,  “batsmanScore2”: batsmanScore2,  “bowlerName”: bowlerName,  “overDetails”: overDetails,  “overCount”: overcount,  “economy”: economy  ]} |
| 404 | json | Error occurred while fetching score details | {  “message”: “Match scores not found”  } |

#### Implementation Details

##### Method Name: ScoreController.listMatchDetails()

1. Authentication - Required
2. Validation – Required

## Dependencies

This service is loosely depending upon,

1. User Management Service
2. Match Details Service
3. Player Statistic Service

# Team Management Service

## Overview

* Stores and manages team data.
* Provides APIs for team profiles, statistic, and squad details.
* Calculates real-time statistic during ongoing matches.
* Handles historical data import and updates.

## Data model specification

TeamProfile

{

“teamId”: “<Long>”

"teamName": "<String>",

"teamHistory": "<String> "

}

Squad

{

“squadId”: “<Long>”

“teamId”: “<Long >”,

“matchId”: “<Long >”,

“players”: <Interger[]>

}

TeamStatistic

{

“Id”: “<Long>”,

“teamName”: “<String>”,

“captain”: “<String>”,

“vcaptain”: “<String>”,

“iccTitles”: “<Integer>”

“playerName”: <String>,

“totalMatches”: {

“format”: “<String>”,

“opponentTeam”: “<String>”,

“count”: <Long>

}

“mom”: {

“playerId”: “<Long>”,

“playerName”: “<String>”,

“highestMoM”: “<Integer>”

}

## API Endpoints

### Create team profile

Create a team profile.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| POST | /api/v1/teams/ | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| teamName | Yes | body | string | Team name |
| history | Yes | body | string | Detailed team history |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 201 | json | Creating team’s profile | {  “message”: “Team's profile created successfully.”  } |
| 400 | json | Invalid input | {  “error”: “Invalid input parameters”  } |
| 401 | Json | Unauthorized request | {  “message”: “Unauthorized request”  } |

#### Implementation Details

##### Method Name: TeamController.createTeamProfile()

1. Authentication - Required
2. Validation - Required

### Update team profile

Update team profile.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/teams/ | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| teamName | Yes | body | string | Team name |
| history | Yes | body | string | Detailed team history |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating team’s profile | {  “message”: “Updated team’s profile”  } |
| 401 | json | Unauthorized access | {  “message”: “Unauthorized access”  } |
| 404 | json | Team profile not found | {  “message”: “Team profile not found in the records”  } |

#### Implementation Details

##### Method Name: MatchController.updateMatchDetails()

1. Authentication - Required
2. Validation - Required

### Get team profile

Fetches a team profile.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/teams/{teamName} | No |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | in | Type | Description |
| teamName | Yes | path | String | Team Id |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning team’s profile | {  "teamId": TeamId,  "teamName": "TeamName",  "history": "Team history"  } |
| 404 | json | Team profile not found | {  "message": “Team profile not found in the records”  } |

#### Implementation Details

##### Method Name: TeamController.fetchTeamProfile()

1. Authentication – Not required
2. Validation – Required

### Create squad details

Update a squad details.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/squads | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| teamId | Yes | body | Number | Team Id |
| matchId | Yes | body | Number | Match Id |
| players | Yes | body | Number Array | Player squad details |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 201 | json | Updating squad details | {  “message”: “Squad created successfully”  } |
| 400 | json | Invalid parameters | {  "error": "{squadId=Duplicate squad for the same team and match}"  } |
| 401 | json | Unauthorized access | {  “message”: “Unauthorized access”  } |

#### Implementation Details

##### Method Name: MatchController.updateSquadDetails()

1. Authentication - Required
2. Validation - Required

### Update squad details

Update a squad details.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/squads | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| teamId | Yes | body | Number | Team Id |
| matchId | Yes | body | Number | Match Id |
| players | Yes | body | Number Array | Player squad details |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating squad details | {  “message”: “Squad updated successfully”  } |
| 401 | json | Unauthorized access | {  “message”: “Unauthorized access”  } |
| 404 | json | Squad not found | {  “message”: “Squad not found in the records”  } |

#### Implementation Details

##### Method Name: MatchController.updateSquadDetails()

1. Authentication - Required
2. Validation – Required

### Get squad details

Fetches a squad information.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/squads/{teamId}/{matchId} | No |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | Type | Description |
| teamId | Yes | path | Number | Team Id |
| matchId | Yes | path | Number | Match Id |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning squad details | {  "squadId": squadId,  "teamId": teamId,  "matchId": matchId,  "players": [List of player ids]  } |
| 404 | json | Squad not found | {  “message”: “Squad not found in the records”  } |

#### Implementation Details

##### Method Name: TeamController.fetchSquadDetails()

1. Authentication – Not required
2. Validation - Required

### Update team statistic

Update a team’s statistic.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/teams/statistics | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| teamName | Yes | body | string | Team Id |
| captain | Yes | Body | string | Captain name |
| vcaptain | Yes | Body | string | Vice-captain name |
| iccTitles | Yes | Body | number | ICC titles |
| totalMatches | Yes | Body | json | Total team matches with respect to format |
| mom | Yes | Body | json | Total man of the match received by players |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating team’s statistic | {  “message”: “Team statistics updated successfully”  } |
| 401 | json | Unauthorized access | {  “message”: “Unauthorized access”  } |
| 404 | json | Error occurred while updating team statistic | {  “message”: “Team statistics not found in the records”  } |

#### Implementation Details

##### Method Name: TeamController.updateStatistics ()

1. Authentication - Required
2. Validation - Required

### Get team statistic

Fetches a team statistic.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/teams/statistics/{teamName} | No |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | Type | Description |
| teamName | yes | path | String | Team name |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning team statistic | {  "id": 1,  "teamName": "TeamA",  "captain": "PlayerA",  "vcaptain": "PlayerB",  "iccTitles": 4,  "totalMatches": {  "format": "t20",  "opponentTeam": "TeamM",  "count": 30  },  "mom": {  "playerId": 1,  "playerName": "Player1",  "highestMoM": 13  }  } |
| 404 | json | Error occurred while fetching team statistic | {  “message”: “Team statistics not found in the records”  } |

#### Implementation Details

##### Method Name: TeamController.fetchStatistic()

1. Authentication – Not required
2. Validation – Required

## Dependencies

This service is loosely depending upon,

1. User Management Service
2. Player Statistic Service

# Player Statistics Service

## Overview

* Stores and manages player performance data.
* Provides APIs for player profiles, career statistic, and in-match performance.
* Calculates real-time statistic during ongoing matches.
* Handles historical data import and updates.

## Data model specification

Player

{

“playerId”: “<UUID>”,

"playerName": "<String>",

"teamName": "<String> ",

"birthDate": "<Date>",

"age": "<Number>",

“role”: “<String>”

}

Performance

{

“playerId”: “<UUID>”,

“matchId”: “<UUID>”,

“runs”: “<Number>”,

“wickets”: “<Number>”,

“fours”: “<Number>”,

“sixes”: “<Number>”

}

Statistics

{

“playerId”: “<UUID>”,

“totalMatches”: [{“format”: “<String>”, “matches”: “<Number>”}, {“format”: “<String>”, “matches”: “<Number>”}],

“runs”: [{“format”: “<String>”, “runs”: “<Number>”}, {“format”: “<String>”, “runs”: “<Number>”}],

“highestRuns”: [{“format”: “<String>”, “highestRuns”: “<Number>”}, {“format”: “<String>”, “highestRuns”: “<Number>”}],

“fifties”: [{“format”: “<String>”, “fifties”: “<Number>”}, {“format”: “<String>”, “fifties”: “<Number>”}],

“hundreds”: [{“format”: “<String>”, “hundreds”: “<Number>”}, {“format”: “<String>”, “hundreds”: “<Number>”}],

“wickets”: [{“format”: “<String>”, “wickets”:“<Number>”}, {“format”: “<String>”, “wickets”:“<Number>”}]

}

## API Endpoints

### Create player profile

Create a player profile.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| POST | /api/v1/players/ | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| playerName | Yes | body | string | Player name |
| teamName | Yes | body | string | Player’s team name |
| birthDate | Yes | body | date | Player’s birthdate |
| age | Yes | body | int | Player age |
| role | Yes | body | string | Role in team |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Creating profile data | {  “error”: false,  “playerId”: playerId,  “message”: “Created player’s profile”  } |
| 400 | json | Invalid input | {  “message”: “Invalid input parameters”  } |
| 401 | json | Unauthorized access | {  “message”: “Unauthorized access”  } |

#### Implementation Details

##### Method Name: PlayerController.createPlayerProfile()

1. Authentication - Required
2. Validation - Required

### Update player profile

Update player profile.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/players/{playerId} | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| playerId | Yes | body | UUID | Player Id |
| playerName | Yes | body | string | Player name |
| teamName | Yes | body | string | Player’s team name |
| birthDate | Yes | body | date | Player’s birthdate |
| age | Yes | body | int | Player age |
| role | Yes | body | string | Role in team |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating player’s profile | {  “error”: false,  “message”: “Updated team’s profile”  } |
| 404 | json | Error occurred while updating player’s profile | {  “message”: “Player profile not found”  } |

#### Implementation Details

##### Method Name: PlayerController.updatePlayerDetails()

1. Authentication - Required
2. Validation - Required

### Get player profile

Fetches a player’s information.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/players/{playerId} | No |

##### Input Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | in | Description |
| playerId | Yes | path | Player Id |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning player’s profile data | {  “playerId”: playerId,  “playerName”: playerName,  “teamName”: teamName,  “birthDate”: birthDate,  “age”: age,  “role”: role  } |
| 404 | json | Error occurred while fetching player’s profile data | {  “message”: “Player profile not found”  } |

#### Implementation Details

##### Method Name: TeamController.fetchTeamProfile()

1. Authentication – Not required
2. Validation – Required

### Update match performance details

Update match performance details.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/players/{playerId}/matches | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| playerId | Yes | path | string | Player Id |
| matchId | Yes | body | string | Match Id |
| runs | Yes | body | Int | Total runs scored by the player in particular match |
| wickets | Yes | body | Int | Total wickets taken by the player in particular match |
| fours | Yes | body | Int | Fours |
| sixes | Yes | body | Int | Sixes |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating match performance details | {  “error”: false,  “message”: “Updated match performance details.”  } |
| 404 | json | Error occurred while updating data | {  “message”: “Player performance details not found”  } |

#### Implementation Details

##### Method Name: PlayerController.updateMatchDetails()

1. Authentication - Required
2. Validation - Required

### Get match performance details

Fetches a match information.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/players/{playerId}/matches | No |

##### Input Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | in | Description |
| playerId | Yes | path | Player Id |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning match performance details | {  “playerId”: playerId,  “matchId”: matchId,  “runs”: runs,  “wickets”: wickets,  “fours”: fours,  “sixes”: sixes  } |
| 404 | json | Error occurred while fetching match performance details | {  “message”: “Player performance details not found”  } |

#### Implementation Details

##### Method Name: PlayerController.fetchMatchDetails()

1. Authentication – Not required
2. Validation - Required

### Update player statistic

Update player statistic.

#### Access Details

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Method | Path | Auth | Role |
| PUT | /api/v1/players/{playerId}/statistic | Yes | User |

##### Input Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Required | In | type | Description |
| playerId | Yes | path | UUID | Player Id |
| totalMatches | Yes | body | json | Total matches played by player |
| highestRuns | Yes | body | json | Highest runs scored by player |
| fifties | Yes | body | json | Number of fifties |
| hundreds | Yes | body | json | Number of hundreds |
| wickets | Yes | body | json | Total wickets |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Updating player’s statistic | {  “error”: false,  “message”: “Updated statistic  } |
| 404 | json | Error occurred while updating player statistic | {  “message”: “Player statistic not found”  } |

#### Implementation Details

##### Method Name: TeamController.updateStatistics ()

1. Authentication - Required
2. Validation - Required

### Get player statistic

Fetches a player statistic.

#### Access Details

|  |  |  |
| --- | --- | --- |
| HTTP Method | Path | Auth |
| GET | /api/v1/players/{playerId}/statistic | No |

##### Input Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | in | Description |
| playerId | yes | path | Player Id |

##### HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| Status Code | Type | Description | Body |
| 200 | json | Returning player statistic | {  “playerId”: “<UUID>”,  “totalMatches”: [{“format”: “<String>”, “matches”: “<Number>”}, {“format”: “<String>”, “matches”: “<Number>”}],  “runs”: [{“format”: “<String>”, “runs”: “<Number>”}, {“format”: “<String>”, “runs”: “<Number>”}],  “highestRuns”: [{“format”: “<String>”, “highestRuns”: “<Number>”}, {“format”: “<String>”, “highestRuns”: “<Number>”}],  “fifties”: [{“format”: “<String>”, “fifties”: “<Number>”}, {“format”: “<String>”, “fifties”: “<Number>”}],  “hundreds”: [{“format”: “<String>”, “hundreds”:“<Number>”}, {“format”: “<String>”, “hundreds”:“<Number>”}],  “wickets”: [{“format”: “<String>”, “wickets”:“<Number>”}, {“format”: “<String>”, “wickets”:“<Number>”}]  } |
| 404 | json | Error occurred while fetching player statistic | {  “message”: “Player statistic not found”  } |

#### Implementation Details

##### Method Name: TeamController.fetchStatistic()

1. Authentication – Not required
2. Validation – Required

## Dependencies

This service is loosely depending upon,

1. User Management Service
2. Live Score Service

# Scalability

To maintain load balancing, high performance, and reliability, it is important to set up a scalable microservice network. Depending on the load, microservices should increase or decrease the instance count (Horizontal scaling) as well as resources like CPU, memory, etc. (Vertical scaling). Kubernetes and Docker Swarm can achieve this scalability by setting up usage limits.

# Monitoring and Observability

Monitoring and observability are crucial aspects of managing microservices architectures. They enable teams to track system health, detect issues, and gain insights into application performance. While monitoring typically focuses on collecting predefined metrics, observability goes further by allowing deeper exploration of system behavior through logs, traces, and events.

We can use following tools to achieve Microservices monitoring and observability,

1. Prometheus: Time-series database and monitoring system

2. Grafana: Visualization and dashboarding platform

3. ELK Stack (Elasticsearch, Logstash, Kibana): Log management and analysis

4. AWS CloudWatch: AWS metrics repository