**Assignment: IPL Players Management API**

**Objective:**

Build an API for managing IPL players with the following functionalities: listing players with pagination and filtering, adding, updating, deleting players, and retrieving detailed descriptions of a player.

**Technologies:**

Node.js

Fastify or Express.js

validation (using libraries like Joi)

Database - MySql or Postgres or MongoDB

**API Endpoints:**

**1. List All Players API (GET /players):**

**Description:** Fetch a paginated list of IPL players. Support filtering by team and pagination of the results.

**Query Parameters:**

**page:** Page number for pagination (default: 1)

**limit:** Number of results per page (default: 10)

**team:** Filter players by team (optional)

**Response:** A paginated list of IPL players with the following fields:

**name:** Name of the player

**image:** URL of the player's image

**role:** Role of the player (Batsman, Bowler, All-rounder)

**team:** Team the player belongs to

**Example Response:**

json

{

"page": 1,

"limit": 10,

"total": 50,

"players": [

{

"id": <UUID>

"name": "Virat Kohli",

"image": "https://linktoimage.com/kohli.jpg",

"role": "Batsman",

"team": "RCB"

},

{

"id": <UUID>

"name": "AB de Villiers",

"image": "https://linktoimage.com/ab.jpg",

"role": "Batsman",

"team": "RCB"

}

]

}

**2. Update Player API (PATCH /players/:id):**

**Description:** Update the details of an existing IPL player.

**Request Body:**

json

{

"name": "Updated Player Name",

"team": "Updated Team Name",

"country": "Updated Country",

"runs": 3000,

"image": "https://linktoimage.com/updatedplayer.jpg",

"role": "Batsman",

"salary": 100000000

}

**Response:**

json

{

"message": "Player updated successfully"

}

**3. Create Player API (POST /players):**

**Description:** Add a new IPL player to the list.

**Request Body:**

json

{

"name": "New Player Name",

"team": "New Team",

"country": "New Country",

"runs": 5000,

"image": "https://linktoimage.com/newplayer.jpg",

"role": "All-rounder",

"salary": 200000000

}

**Response:**

json

{

"message": "Player created successfully"

}

**4. Delete Player API (DELETE /players/:id):**

**Description:** Delete an IPL player by their ID.

**Response:**

json

{

"message": "Player deleted successfully"

}

**5. Description About Player API (GET /players/:id/description):**

**Description:** Get detailed information about an IPL player, including their name, team, country, total runs, image, role, and salary.

**Response:**

json

{

"name": "Virat Kohli",

"team": "RCB",

"country": "India",

"runs": 6000,

"image": "https://linktoimage.com/kohli.jpg",

"role": "Batsman",

"salary": 150000000

}

* **Validation:**

For each API that handles input (such as adding or updating a player), ensure that the data is validated before processing. You can use the following validation rules:

**name:** Required, must be a non-empty string.

**team:** Required, must be a non-empty string.

**country:** Required, must be a non-empty string.runs: Required, must be an integer.

**image:** binary

**role:** Required, must be one of Batsman, Bowler, All-rounder.

**salary:** Required, must be a positive number.

Use a validation library like Joi.

* **Pagination:**

For the List All Players API, support pagination by using the page and limit query parameters:

If no page is provided, default to page = 1.

If no limit is provided, default to limit = 10.

The API should also return the total number of players along with the current page of results.

* **Sorting, Searching**

**Sorting:** Add functionality to sort players by runs or salary.

**Search:** Implement a search option that allows the user to search for players by their name (optional).

**Error Handling:** Ensure proper error handling for cases like invalid inputs, missing query parameters, or player not found.