Reference exercises for knowledge testing and candidate development

The goal is to implement a simple application for searching entities such as competitions, teams, and players using the Livesport Search Service API.

Functional requirements

The application will consist of two screens: a results overview and a detailed view.

Results Overview will contain

- Title: For example "Results."
- Search field with a button for executing the search.
- Results list displaying the retrieved entries.
- Users should be able to filter results based on entity types (as specified in the API parameters):
    - All Types (IDs: 1, 2, 3, 4).
    - Competitions Only (ID: 1).
    - Participants Only (IDs: 2, 3, 4).
- The application should clearly display loading states while data is being fetched.
- In case of any error (e.g., lack of internet connection or server issues), an alert message should appear with relevant information and a button labeled "Retry."
- Each row in the results list must include at least:
    - Entity name, for example: Arsenal FC, Roger Federer.
    - Logo/photo, or a placeholder if the image is missing.
    - Each row must be clickable/navigable to access its detailed view.

Detail View will feature

- A title displaying the entity's name.
- Enlarged logo/photo/placeholder as applicable.
- Country details related to the competition/team/player.
- Additional relevant data as deemed appropriate based on availability.

This is the minimal required and sufficient functionality of the application. Any additional improvements are, of course, possible as desired, but there's no need to overdo it.

Unfunctional requirement

- Implement the **search** field component yourself, but for other UI elements, you can use a component library at your discretion.
- For data handling, make use of **Promises** or **async/await**.
- Cover the logic with **tests** (I would like to see one test for the component that verifies its behavior and content as a UI element, and another test focusing on the asynchronous logic for data fetching).
- Use **Git** and commit your work logically and incrementally throughout the process.

Livesport Search Service API

Endpoint

- https://s.livesport.services/api/v2/search

Parameters (all are mandatory)

- `lang-id`

Value	Description
1	English

- `project-id`

Value	Description
602	Flashscore Portable projekt

- `project-type-id`

Value	Description
1	

- `sport-ids`

Value	Description
1	football
2	tennis
3	basketball
4	hockey
5	american football
6	baseball
7	handball
8	rugby
9	floorball

- `type-ids`

Value	Description
1	Competitions
2	Teams
3	Individual players (e.g., tennis players)
4	Team players (e.g., football players)

- q

Value	Description
Any string	Search words

Requestu and response example

Request

https://s.livesport.services/api/v2/search?type-ids=2,3&project-type-id=1&project-id=602&lang-id=1&q=dj&sport-ids=1,2,3,4,5,6,7,8,9

Response


```
[
  {
    "id": "AZg49Et9",
    "url": "djokovic-novak",
    "gender": {
      "id": 1,
      "name": "Men"
    },
    "name": "Djokovic Novak",
    "type": {
      "id": 3,
      "name": "Player"
    },
    "participantTypes": [
      {
        "id": 2,
        "name": "Player"
      }
    ],
    "sport": {
      "id": 2,
      "name": "Tennis"
    },
    "favouriteKey": {
      "web": "2_AZg49Et9",
      "portable": "2_AZg49Et9"
    },
    "flagId": null,
    "defaultCountry": {
      "id": 167,
      "name": "Serbia"
    },
    "images": [
      {
        "path": "tSfwGCdM-0rY6MEPI.png",
        "usageId": 3,
        "variantTypeId": 15
      }
    ],
    "teams": [],
    "defaultTournament": null,
```

```
      "superTemplate": null
    }
  ]
```

- The name of the entity is located under the “name” property.
- The name of the image (logo of the team/competition) is included in the “images” section, where we select the “variantTypeId” = 15 (image size 100x100px). The image is presented in the format “XXXXXXXX-XXXXXXXX.png”, e.g., “CWPezaCa-UX2TBBm0.png”. Its content can be retrieved from a URL structured as follows: “https://www.livesport.cz/res/image/data/XXXXXXXX-XXXXXXXX.png”.
- For Djokovic’s 100x100 image, it will be accessible via this URL: https://www.livesport.cz/res/image/data/tSfwGCdM-0rY6MEPI.png
- If an image in this specific variant for a competition or participant cannot be found within the response, a placeholder should be displayed instead.
- The properties that are relevant to this task are marked as bold and underlined for emphasis.

Possible response codes

- `200 OK`
- `400` - error 101 One or more values are missing, see array of errors for details.

```
{
  "code": 101,
  "message": "One or more values are missing, see array of errors
for details.",
  "name": "BadRequestError",
  "stack": "BadRequestError: One or more values are missing, see
array of errors for details.\n …)",
  "errors": [
    {
      "message": "\"q\" is required",
      "type": "any.required"
    }
  ]
}
```

- 422 - error 100 One or more values are invalid, see array of errors for details.


```
{
  "code": 100,
  "message": "One or more values are invalid, see array of errors
for details.",
  "name": "UnprocessableEntityError",
  "stack": "UnprocessableEntityError: One or more values are
invalid, see array of errors for details.\n …)",
  "errors": [
    {
      "message": "\"project-type-id\" must be one of [LOCAL, IN,
MOBI, ASS, DEVEL, LIVEINPUT, SCORERADAR, 1, 2, 3, 4, 5, 6, 7]",
      "type": "any.only"
    }
  ]
}
```


- 503 - error 110 Sport API error


```
{
  "code": 110,
  "message": "Sport API error",
  "name": "ServiceUnavailableError",
  "stack": "ServiceUnavailableError: Sport API error\n …)",
  "data": {
    "name": "MissingResponseError"
  }
}
```

Inspiration for UI