Exercise BACKEND

REST API to List Prizes by Catalog

Goal: Assess your skills in REST API development, using object-oriented concepts, exception handling, and good programming practices.

Estimated Time: 2 hours

Description:

Implement a REST API in Python that lists prizes by catalog, meeting the following requirements:

Functionality:

- **List prizes:**
  - Receives the following parameters:
    - `catalog_id` (number): Catalog identifier.
    - `filter` (optional): Dictionary with the fields:
      - `id` (optional): Prize identifier.
      - `description` (optional): Prize description (substring search).
    - `pagination` (optional): Dictionary with the fields:
      - `page` (number): Page number to be returned (starts at 1).
      - `per_page` (number): Number of prizes per page.
  - Returns a JSON object with:
    - `total` (number): Total number of prizes found.
    - `prizes` (list): List of objects with the prize data:
      - `id` (number): Prize identifier.
      - `title` (string): Prize title.
      - `description` (string): Prize description.
      - `image` (string): URL of the prize image.

Requirements:

- **API Development:**

    - Create an `API` class that defines the API routes and methods.
    - Create a `list_prizes` method that receives the parameters and returns the list of prizes.
    - Validate input parameters and return error messages in case of errors.
    - Handle exceptions and return appropriate error messages.
    - (optional but a plus) Use the `flask` module to create the REST API.
        - Define the `/api/catalogs/<catalog_id>/prizes` route.
- **Data Simulation:**
    - Create a `Prize` class that represents a prize.
    - Create a class that simulates the database query. Mock the data instead of connect and execute and query.
    - Create a `get_prizes` method in that class, that returns a mock list of prizes.
    - The `get_prizes` method should receive the `catalog_id`, `filter`, and `pagination` parameters.
    - The `get_prizes` method should filter the list of prizes according to the `filter` and `pagination` parameters.
- **Testing:**
    - (optional but a plus) Write unit tests for the `list_prizes` method using the `pytest` module.
    - Test different input scenarios and parameter validation.
    - Test the API behavior with different filters and pagination.
- **Code versioning:**
    - Upload the code in a public git repository, we will use that to valutate the code.
- **Documentation:**
    - Provide the necessary documentation to correctly use and test the API method.

Examples:

Example 1: List all prizes from catalog 1

Input:

`GET /api/catalogs/1/prizes`

Output:

JSON

```
{
  "total": 10,
  "prizes": [
    {
      "id": 1,
      "title": "Prize 1",
      "description": "Description of prize 1",
      "image": "https://example.com/image1.png"
    },
    {
      "id": 2,
      "title": "Prize 2",
      "description": "Description of prize 2",
      "image": "https://example.com/image2.png"
    },
    ...
  ]
}
```