Sports Event Management API Documentation

Table of Contents

Project Overview

The Sports Event Management API is a Flask-based application that allows users to manage sports, events, and selections. The application supports creating, updating, searching, and managing the status of these entities. When all the selections of a particular event are inactive, the event becomes inactive, and similarly, when all events of a sport are inactive, the sport becomes inactive.

Prerequisites

- Docker
- Docker Compose

Installation

1. Clone the repository:

   ```
   git clone https://github.com/your-username/sports-event-management.git
   cd sports-event-management
   ```

2. Ensure Docker is installed and running on your system.

Running the Application

1. Build the Docker image:

   ```
   docker-compose build
   ```

2. Run the Docker containers:

   ```
   docker-compose up
   ```

3. Access the application at `http://localhost:8000`.

API Endpoints

Sports

- **Create Sport**
  - **URL:** `/sports/`
  - **Method:** `POST`
  - **Body:**

    ```
    {
      "name": "Basketball",
      "slug": "basketball",
      "active": true
    }
    ```

  - **Response:**

    ```
    {
      "id": 1,
      "name": "Basketball",
      "slug": "basketball",
      "active": true
    }
    ```

- **Update Sport**
  - **URL:** `/sports/<int:sport_id>`
  - **Method:** `PUT`
  - **Body:**

    ```
    {
      "name": "Updated Basketball",
      "slug": "updated-basketball",
      "active": false
    }
    ```

  - **Response:**

```
{
  "id": 1,
  "name": "Updated Basketball",
  "slug": "updated-basketball",
  "active": false
}
```

- **Search Sports**
  - **URL:** `/sports/search`
  - **Method:** `POST`
  - **Body:**

```
{
  "name_regex": "Basketball",
  "min_active_events": 1
}
```

  - **Response:**

```
[
  {
    "id": 1,
    "name": "Basketball",
    "slug": "basketball",
    "active": true
  }
]
```

Events

- **Create Event**
  - **URL:** `/events/`
  - **Method:** `POST`
  - **Body:**

```
{
  "name": "Internazionale vs. Shakhtar Donetsk",
  "slug": "internazionale-vs-shakhtar-donetsk",
  "active": true,
  "type": "preplay",
  "sport_id": 1,
  "status": "Pending",
  "scheduled_start": "2023-06-10T20:00:00"
}
```

  - **Response:**

```
{
  "id": 1,
  "name": "Internazionale vs. Shakhtar Donetsk",
  "slug": "internazionale-vs-shakhtar-donetsk",
  "active": true,
```

```
  "type": "preplay",
  "sport_id": 1,
  "status": "Pending",
  "scheduled_start": "2023-06-10T20:00:00"
}
```

- **Update Event**
  - **URL:** `/events/<int:event_id>`
  - **Method:** `PUT`
  - **Body:**

```
{
  "name": "Updated Cricket Match",
  "slug": "updated-cricket-match",
  "active": true,
  "type": "preplay",
  "sport_id": 1,
  "status": "Started",
  "scheduled_start": "2023-06-10T20:00:00"
}
```

  - **Response:**

```
{
  "id": 1,
  "name": "Updated Cricket Match",
  "slug": "updated-cricket-match",
  "active": true,
  "type": "preplay",
  "sport_id": 1,
  "status": "Started",
  "scheduled_start": "2023-06-10T20:00:00"
}
```

- **Search Events**
  - **URL:** `/events/search`
  - **Method:** `POST`
  - **Body:**

```
{
  "name_regex": "Cricket",
  "min_active_events": 1,
  "min_active_selections": 1,
  "scheduled_start": ["2023-06-01T00:00:00", "2023-06-
30T23:59:59"]
}
```

  - **Response:**

```
[
  {
    "id": 1,
```

```
    "name": "Cricket Match",
    "slug": "cricket-match",
    "active": true,
    "type": "preplay",
    "sport_id": 1,
    "status": "Pending",
    "scheduled_start": "2023-06-10T20:00:00"
  }
]
```

Selections

- **Create Selection**
  - **URL:** `/selections/`
  - **Method:** `POST`
  - **Body:**

```
{
  "name": "1",
  "event_id": 1,
  "price": 1.63,
  "active": true,
  "outcome": "Unsettled"
}
```

  - **Response:**

```
{
  "id": 1,
  "name": "1",
  "event_id": 1,
  "price": 1.63,
  "active": true,
  "outcome": "Unsettled"
}
```

- **Update Selection**
  - **URL:** `/selections/<int:selection_id>`
  - **Method:** `PUT`
  - **Body:**

```
{
  "name": "Updated Selection",
  "active": false
}
```

  - **Response:**

```
{
  "id": 1,
  "name": "Updated Selection",
  "event_id": 1,
```

```
  "price": 1.63,
  "active": false,
  "outcome": "Unsettled"
}
```

- **Search Selections**
  - **URL:** `/selections/search`
  - **Method:** `POST`
  - **Body:**

    ```
    {
      "name_regex": "1",
      "min_active_events": 1,
      "min_active_selections": 1,
      "scheduled_start": ["2023-06-01T00:00:00", "2023-06-
    30T23:59:59"]
    }
    ```

  - **Response:**

    ```
    [
      {
        "id": 1,
        "name": "1",
        "event_id": 1,
        "price": 1.63,
        "active": true,
        "outcome": "Unsettled"
      }
    ]
    ```

Testing

To run the tests, use the following command:

```
docker-compose run web python -m unittest discover -s tests
```

Project Structure

```
sportsapp/
  ├── __init__.py
  ├── database.py
  ├── models.py
  ├── schemas.py
  ├── crud.py
  ├── routes.py
tests/
  ├── __init__.py
  ├── test_main.py
main.py
requirements.txt
Dockerfile
```

docker-compose.yml

Development Notes

- **Initialization**: Ensure the database is initialized with the correct schema before starting the application.
- **Error Handling**: Use appropriate error handling for database operations to provide meaningful error messages to the API consumers.
- **Testing**: Ensure thorough testing of all API endpoints to verify functionality and performance.

Docker Setup

Dockerfile

```
# Dockerfile
FROM python:3.8-slim

# Set environment variables
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

# Set work directory
WORKDIR /code

# Install dependencies
COPY requirements.txt /code/
RUN pip install --no-cache-dir -r requirements.txt

# Copy project
COPY . /code/

# Run the application
CMD ["flask", "run", "--host=0.0.0.0", "--port=8000"]
```

docker-compose.yml

```
version: '3.8'

services:
  web:
    build: .
    command: flask run --host=0.0.0.0 --port=8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    environment:
      FLASK_APP: "main"
      FLASK_ENV: "development"
```

Steps to Create Docker Setup

1. **Build the Docker image**:

   ```
   docker-compose build
   ```

2. **Run the Docker containers**:

   ```
   docker-compose up
   ```

3. **Verify the application** by navigating to `http://localhost:8000` in your web browser.
