Events Backend

Please refer to Events Frontend to understand what events are. In short, they define what, where and when a sporting event is going to take place, along with other details.

DB Schema

events table

```
CREATE TABLE events (
    id SERIAL PRIMARY KEY,
    event_owner INT REFERENCES users(id) ON DELETE CASCADE,
    sport TEXT NOT NULL,
    event_datetime TIMESTAMP NOT NULL,
    max_players INT NOT NULL CHECK (max_players > 0),
    location_name TEXT NOT NULL,
    latitude DECIMAL(9,6) NOT NULL,
    longitude DECIMAL(9,6) NOT NULL,
    description TEXT,
    title TEXT,
    is_full BOOLEAN,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Column	Comments
id	primary key, id of the event. can be used to fetch a particular event
event_owner	stores the user id of the user who created the event
sport	text field that shows which sport the event is associated to (in the future, consider having sports as a separate table so that every object has sports taken from that table. easy for scalability)
event_date	datetime field that shows when the event is going to occur. in the frontend, date and time is written by the

	user in separate fields, but frontend sends a datetime converted field to the backend
max_players	int field that shows how many players can join the event
location_name	text field that stores the location name where the event is expected to happen
latitude	decimal field. frontend uses google maps api to get the geocode for the location the user enters. this geocode is stored as latitude and longitude in the backend. the backend sends this to any frontend GET calls, where frontend reverse geocodes and fetches the location again from google maps api.
longitude	decimal field. stores the longitude of the location
title	text field that stores the title of the event
description	text field that stores the short description of the event
created_at	timestamp of when the event is created by a user
updated_at	timestamp of when the event is updated by a user
is_full	boolean that indicates whether the event has been filled up or not

event_participants

```
CREATE TABLE event_participants (
    id SERIAL PRIMARY KEY,
    event_id INT REFERENCES events(id) ON DELETE CASCADE,
    user_id INT REFERENCES users(id) ON DELETE CASCADE,
    joined_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Column	Comment
id	event participant id
event_id	references the event id from the events table
user_id	references the user id from the users table
joined_at	timestamp of when a user joins the event. could be used to show the joined users based on when they joined

---

API Spec

Request Headers for all APIs

```
{
  "Content-Type": "application/json",
  "Authorization": "Bearer ${localStorage.getItem('jwt_token')}"
}
```

POST - `/events` - Create event

Adds event details to the **events** table

Request Body

```
{
  "sport": "Football",
  "event_date": "2025-03-25T15:00:00Z",
  "max_players": 20,
  "location_name": "Central Park, NY",
  "latitude": 40.785091,
  "longitude": -73.968285,
  "description": "Need players for a friendly football match",
  "title": "Lets football!"
}
```

Response Body

200 OK: Event successfully created.

```
{
  "id": 1,
  "event_owner": 1,
  "sport": "Football",
  "event_datetime": "2025-03-25T15:00:00Z",
  "max_players": 20,
  "location_name": "Central Park, NY",
  "latitude": 40.785091,
  "longitude": -73.968285,
  "description": "Friendly football match",
  "title": "Lets football!",
  "created_at": "2025-03-25T10:00:00Z",
  "updated_at": "2025-03-25T10:00:00Z",
  "is_full": false,
  "registered_count": 0
}
```

GET - `/events` - All events with a filter

Gets the details from **events** as well as **event_participants** (to get the registered_count)

Request Body

```
{
  "id": 1 //search based on event id
  "sport": "Football", //filter based on sport
  "max_players": 20, //filter based on max_players
}
```

Response Body

200 OK: List of events based on filters.


```
[
  {
    "id": 1,
    "event_owner": 1,
    "sport": "Football",
    "event_datetime": "2025-03-25T15:00:00Z",
    "max_players": 20,
    "location_name": "Central Park, NY",
    "latitude": 40.785091,
    "longitude": -73.968285,
    "description": "Friendly football match",
    "title": "Lets football!",
    "created_at": "2025-03-25T10:00:00Z",
    "updated_at": "2025-03-25T10:00:00Z",
    "is_full": false,
    "registered_count": 2,
    "event_participants": [
      {
        "id": 2,
        "name": "John Doe",
        "email": "johndoe@gmail.com"
      },
      {
        "id": 3,
        "name": "John Does",
        "email": "johndoes@gmail.com"
      }
    ]
  },
  {
    "id": 2,
    "event_owner": 3,
    "sport": "Football",
    "event_datetime": "2025-03-27T15:00:00Z",
    "max_players": 20,
    "location_name": "Central Park, NY",
    "latitude": 40.785091,
    "longitude": -73.968285,
    "description": "Unfriendly football match",
    "title": "Lets play football!",
    "created_at": "2025-03-25T10:00:00Z",
    "updated_at": "2025-03-25T10:00:00Z",
    "is_full": false,
    "registered_count": 2,
    "event_participants": [
      {
        "id": 2,
        "name": "John Doe",
        "email": "johndoe@gmail.com"
      },
      {
        "id": 3,
        "name": "John Does",
        "email": "johndoes@gmail.com"
      }
```

```
    ]
  }
]

```

GET - `/events/:id` - Get Single Event by ID

Gets the details from **events** as well as **event_participants** (to get the registered_count)

Request Body

No body required

Response Body

200 OK: Event details

```
{
  "id": 1,
  "event_owner": 1,
  "sport": "Football",
  "event_datetime": "2025-03-25T15:00:00Z",
  "max_players": 20,
  "location_name": "Central Park, NY",
  "latitude": 40.785091,
  "longitude": -73.968285,
  "description": "Friendly football match",
  "title": "Lets play football!",
  "created_at": "2025-03-25T10:00:00Z",
  "updated_at": "2025-03-25T10:00:00Z",
  "is_full": false,
  "registered_count": 2,
  "event_participants": [
      {
        "id": 2,
        "name": "John Doe",
        "email": "johndoe@gmail.com"
      },
      {
        "id": 3,
        "name": "John Does",
        "email": "johndoes@gmail.com"
      }
    ]
}
```

PUT - `/events/:id` - Update an Event

Updates event details on the **events** table (what should happen to the users who have already joined this event?)

Request Body

All fields are optional - can update any or all of the fields

```
{
  "sport": "Basketball",
  "event_date": "2025-03-26T15:00:00Z",
  "max_players": 25,
```

```
  "location_name": "Madison Square Garden",
  "latitude": 40.748817,
  "longitude": -73.985428,
  "title": "Lets play Basketball",
  "description": "Basketball tournament",
  "event_datetime": "2025-03-25T15:00:00Z"
}
```

Response Body

200 OK: Event updated successfully.

```
{
  "id": 1,
  "event_owner": 1,
  "sport": "Basketball",
  "event_datetime": "2025-03-26T15:00:00Z",
  "max_players": 25,
  "location_name": "Madison Square Garden",
  "latitude": 40.748817,
  "longitude": -73.985428,
  "description": "Basketball tournament",
  "title": "Lets play Basketball",
  "created_at": "2025-03-25T10:00:00Z",
  "updated_at": "2025-03-25T11:00:00Z"
}
```

POST - `/events/:id/join` - Join an Event

The backend must add the participant information in the **event_participants** table, and also update the **is_full** column on the **events** table in case it's full.

Request Body

No body required

Response Body

200 OK: Event joined successfully.

```
{
  "message": "You have successfully joined the event!"
}
```

DELETE - `/events/:id/leave` - Leave an Event

Used when a user leaves an event. The backend must remove the participant information in the **event_participants** table, and also update the **is_full** column on the **events** table.

Request Body

No body required

Response Body

200 OK: Event left successfully.

```
{
  "message": "You have successfully left the event!"
}
```

DELETE - `/events/:id` - Delete an Event

Used when an event owner wants to delete their event. The backend must remove the event information in the **events** table and **event_participants** table.

Request Body

No body required

Response Body

200 OK: Event deleted successfully.

```
{
  "message": "You have successfully deleted the event!"
}
```