

# Events Frontend

[What is an event?](#)

[Event Location](#)

[Dashboard](#)

[Event Create](#)

---

[Events Get - Dashboard](#)

[Event View - Get Single Event](#)

[Event View - Own and Joined Events](#)

[Event Update:](#)

[Join Event](#)

[Leave Event](#)

[Delete Event](#)

## What is an event? [🔗](#)

In Sportify, every user can create or join a sport “event”. Each user will be able to create an event with the following details:

1. Title
2. Description of the event
3. Sport
4. Date and time of the event
5. Location name
6. Number of players who can join the event

## Event Location [🔗](#)

In the create event form, the user should be able to enter the location in a search field, and be able to select the location from the google maps api results. As the user is typing, google maps api will be called to fetch results, and the user should be able to select one.

This is then sent to the backend where it'll be stored as location name, latitude and longitude. when a GET call is made to the backend, it will return this latitude and longitude and frontend has to reverse geocode using gmaps api to get the location from the latitude and longitude and display to the user.

### References:

[🔗 Place Autocomplete](#) | [🔗 Maps JavaScript API](#) | [🔗 Google for Developers](#)

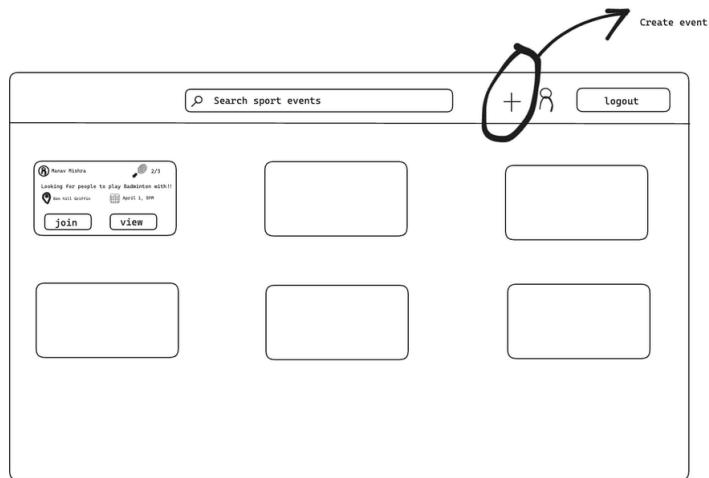
[🔗 How to add Google Maps Autocomplete search box?](#)

## Dashboard [🔗](#)

In the dashboard, the user will be able to see all the events, which will later on be optimised to show events closer to the user. For now, the user will see all the events, in a small viewing box, with the title, sport logo, date, time and number of spots filled for the event.

Rough UI:

# Dashboard



## Event Create [🔗](#)

Beside the profile button, a "+" (plus) symbol signifies the create event button. The user can click it to create their own sport event. Once the button is clicked, the user will be shown a form, with the details of the sport event they are creating. The user can add the details and click "Create".

Rough UI:

## Create event

The 'Create event' form is displayed within the dashboard's header area. It contains the following fields and controls: a text input for 'Title of the event', a larger text area for 'Description', a dropdown menu for 'Sport' with a 'Sport Dropdown' label, a text input for 'No. of players', a location search input with a magnifying glass icon and a dropdown menu showing suggestions like 'Ben Hill Griffin Stadium - University of Fla...', a date input labeled 'Date mm/dd/yyyy', a time input labeled 'Time', and a 'Create' button at the bottom right.

Once the user clicks "create", the frontend makes a POST /event call to the backend with the following headers and parameters:

**URL:** /events

**Method:** POST

**headers:**

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

**parameters:**

```
1 {
2   "sport": "Football",
3   "event_date": "2025-03-25T15:00:00",
4   "max_players": 20,
5   "location_name": "Central Park, NY",
6   "latitude": 40.785091,
7   "longitude": -73.968285,
8   "description": "Friendly football match",
9   "title": "Need players for football match"
```

```
10 }
```

If the backend responds with 201, it means that the event creation is successful.

## Events Get - Dashboard [🔗](#)

**URL:** `/events`

**Method:** `GET`

```
1 {  
2   sport: "Football" #(optional)  
3   event_date: "2025-03-25T15:00:00" #(optional)  
4   max_players: 3 #(optional)  
5 }
```

For the dashboard, no filters (parameters) are necessary. Make a call to `GET /events` to fetch all events and display it on the dashboard.

## Event View - Get Single Event [🔗](#)

**URL:** `/events/:id`

**Method:** `GET`

```
1 GET /events/1
```

When a user wants to view an event from the dashboard or a user profile, once they click the “view” button, frontend must make a call to the backend to fetch the details of the event with the event id.

Rough UI:

## View Event



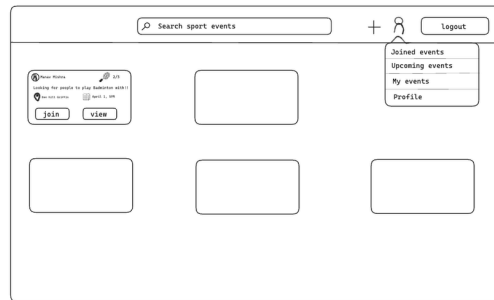
The image shows a rough UI for a 'View Event' page. At the top, there is a search bar with the placeholder text 'Search sport events', a plus icon, a user profile icon, and a 'Logout' button. Below this, the main content area displays event details for a user named 'Manav Mishra' (2/3). The details include a title 'Looking for people to play Badminton with!!', a description 'I'm planning on playing badminton but I'm lonely and I need 3 more players. Let's play doubles!', and a form with fields for 'Badminton' (selected), '3' (max\_players), 'Southwest Recsports' (location), '04/02/2025' (date), and '5:00 PM' (time). A 'Join' button is at the bottom right.

## Event View - Own and Joined Events [🔗](#)

A user can click on a profile, or their own profile to view all events a user (or themselves) are a part of. For the owner, when the profile button is clicked, 4 options are shown

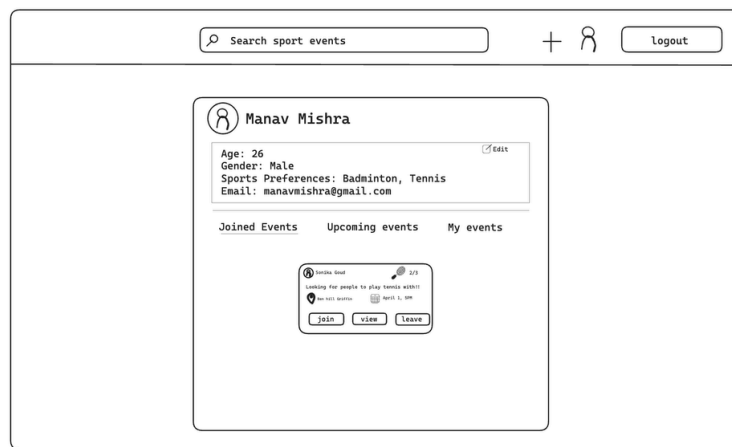
1. Joined events - All events that they have joined so far
2. Upcoming events - Events that have not yet taken place, but have joined
3. My events - Events that the user created
4. Profile - User details

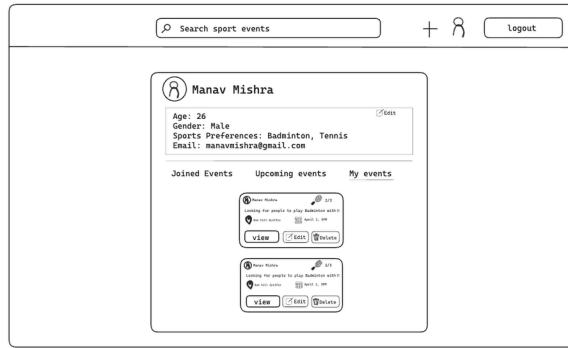
## User Profile and Events



If a user clicks their own profile logo (as shown above) and selects any of the 4 options.

Rough UI:





If a user X clicks on user Y's Profile, they will be shown the Y's name, profile details and the events associated with Y. To get joined events and created events of a user, this is the call:

**URL:** /events

**Method:** GET

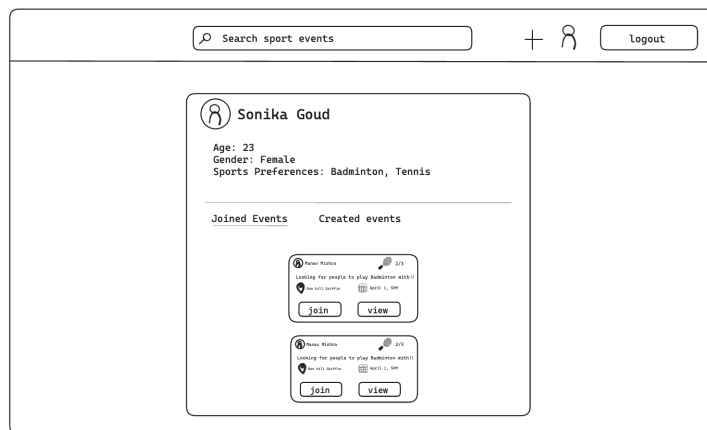
**GET created events:** [Events Frontend | Events Get](#) [Dashboard](#)

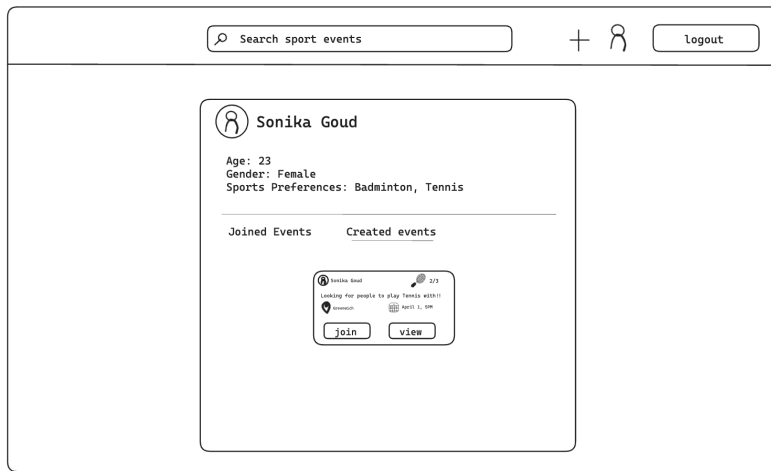
```
1 {
2   event_owner_id: 1
3 }
```

**GET Joined events:**

**TBD**

Rough UI:





## Event Update:

A user can update an event they created. Frontend makes a PUT call to the backend. API spec is:

**URL:** `/events/:id`

**Method:** PUT

```
1 {  
2   "sport": "Basketball",  
3   "event_date": "2025-03-26T15:00:00",  
4   "max_players": 25,  
5   "location_name": "Madison Square Garden",  
6   "latitude": 40.748817,  
7   "longitude": -73.985428,  
8   "description": "Basketball tournament"  
9 }
```

Rough UI:

## Update event

A form for updating an event. At the top is a search bar labeled "Search sport events" with a magnifying glass icon, followed by a "+" icon, a user icon, and a "logout" button. The form contains several input fields: "Title of the event", "Description", "Sport Dropdown" (with a downward arrow), "No. of players", "Location Search" (with a magnifying glass icon and a dropdown menu showing "Ben Hill Griffin Stadium - University of Flo ..." and "Ben Hill Grifforth Auditorium - Ocala"), "Date mm/dd/yyyy", and "Time". An "Update" button is at the bottom right.

## Join Event [↗](#)

When the user clicks "Join" button on the event, frontend makes a POST call to the backend. API spec:

**URL:** `/events/:id/join`

**Method:** `POST`

## Leave Event [↗](#)

When the event owner clicks "delete" button on the event, frontend makes a DELETE call to the backend. API spec:

**URL:** `/events/:id/`

**Method:** `DELETE`

## Delete Event [↗](#)

When the user wants to delete an event, frontend makes a DELETE call to the backend. API spec:

**URL:** `/events/:id/delete`

**Method:** `DELETE`