To create an event:

`POST /events`

Input

Name	Type	Description
name	string	**Required.** Name of event.
date	string	**Required.** Date of event (YYYY-MM-DD).
time	string	Time of event (HH:MM)
all_day	boolean	**True.** Indicates event is an all day event.
description	string	Description of event
account_id	string	**Required.** User's account ID or user's Facebook account ID

Name, date and account_id is required when creating an event. An account_id can represent the ID of a user entity or ID of a user Facebook account. Other inputs, time and description, if not specified will set to NULL while all_day will set to FALSE. If all_day is set to true, any time value specified will be ignored.

When a call is made to POST/events, a new event will be created. Before adding the new event to data store, the account_id provided in the request body is checked if the id is associated with a user entity. If a valid user entity is present, the newly created event will be added to the user's events list.

An example of user entity:

```
{
      "first_name": "John",
      "last_name": "Doe",
      "username": "johnDoe",
      "email": "johndoe@gmail.com"
      "password": "password",
      "id": "user1234",
      "events": []
}
```

A sample request with above account id is shown below:

```
{
      "name": "Event 1",
      "date": "2017-09-01",
      "account_id": "user1234",
      "time": "12:30"
      "description": "Sample event #1",
```

```
        "all_day": false
    }
```

Response:
Status: 200 OK

```
    {
        "name": "Event 1",
        "date": "2017-09-01",
        "account_id": "user1234",
        "time": "12:30"
        "description": "Sample event #1",
        "all_day": false,
        "id": "event1234"
        "self": "/events/event1234"
    }
```

Checking John Doe's account will show the following:

```
    {
        "first_name": "John",
        "last_name": "Doe",
        "username": "johnDoe",
        "email": "johndoe@gmail.com"
        "password": "password",
        "id": "user1234",
        "events": [{
                    "id": "event1234"
                    "self": "/events/event1234"
        }]
    }
```

Delete all event entities:

`DELETE /events`

Delete all event entities from the data store. Before deleting an event entity, if an event is associated with a user entity, the reference to the event will be removed from the user entity.

In addition, the event history entity is created with the same event information and added to data store.

Response:
Status: 200 OK

Retrieve all event entities:

`GET /events`

Display a list of all event entities currently in data store.

Response:
Status: 200 OK

```
{“events”: [
  {
     “name”: “Event 1”,
     “date”: “2017-09-01”,
     “account_id”: “user1234”,
     “time”: “12:30”
     “description”: “Sample event #1”,
     “all_day”: false,
     “id”: “event1234”
     “self”: “/events/event1234”
  },
  {
     “name”: “Event 2”,
     “date”: “2017-09-019”,
     “account_id”: “facebookUser1234”,
     “time”: “12:30”
     “description”: “Sample event #1”,
     “all_day”: false,
     “id”: “event5678”
     “self”: “/events/event5678”
  },
  ...]}
```

Retrieve an event entity:

`GET /events/{event id}`

Display an event entity. An example request:

```
GET /events/event1234
```

Response:
Status: 200 OK

```
{
    “name”: “Event 1”,
    “date”: “2017-09-01”,
    “account_id”: “user1234”,
    “time”: “12:30”
    “description”: “Sample event #1”,
    “all_day”: false,
    “id”: “event1234”
    “self”: “/events/event1234”
}
```

Delete an event entity:

```
DELETE /events/{event id}}
```

Delete an event entity. Before deleting, remove event reference from user entity if event associated to a user entity, and create an event history entity for this event.

Response:
Status: 200 OK

Update an event entity:

```
PUT /events/{event id}
```

Edit the information of an event with new information sent from the request body. All properties of an event entity can be updated except account_id.

Input

Name	Type	Description
name	string	New event of event. (optional)
date	string	New date of event (YYYY-MM-DD). (optional)
time	string	New time of event (HH:MM). (optional)

all_day	boolean	New value for all_day. (optional)
description	string	New description of event. (optional)

An example of request:

```
{
    “name”: “Modified Event 1”,
    “date”: “2017-09-18”,
    “description”: “Updated event #1”,
    “all_day”: true
}
```

Response:

Status: 200 OK

```
{
    “name”: “Modified Event 1”,
    “date”: “2017-09-18”,
    “account_id”: “user1234”,
    “time”: null,
    “description”: “Updated event #1”,
    “all_day”: true,
    “id”: “event1234”
    “self”: “/events/event1234”
}
```

Create a user entity:

`POST /users`

Input

Name	Type	Description

first_name	string	**Required.** First name of user.
last_name	string	**Required.** Last name of user.
username	string	**Required.** Username
email	string	**Required.** Email of user.
password	string	**Required.** Password of user

Create a new user entity with information provided in request body. Check that all required properties provided. Also check that username and email is not associated with another user entity. If both username and email are unique, and all properties are provided, a new user entity will be created. By default, all user entity starts with an empty events list.

Response:
Status: 200 OK

```
{
      "first_name": "John",
      "last_name": "Doe",
      "username": "johnDoe",
      "email": "johndoe@gmail.com"
      "password": "password",
      "id": "user1234",
      "events": []
}
```

Delete all user entities:

`DELETE /users`

Delete all user entities in data store. Before deleting a user entity, check that no events are associated with this user entity. Delete any events associated with this user before deleting the user entity.

Response:
Status: 200 OK

Get all user entities:

`GET /users`

Display a list of all user entities in the data store.

Response:
Status: 200 OK

Get a user entity:

`GET /users/{user ID}`

Display a single user entity.

Response:
Status: 200 OK

```
{
     "first_name": "John",
     "last_name": "Doe",
     "username": "johnDoe",
     "email": "johndoe@gmail.com"
     "password": "password",
     "id": "user1234",
     "events": [{
               "id": "event1234",
               "self": "/events/event1234"
     }]
}
```

Delete a user entity:

`DELETE /users/{user ID}`

Delete a user entity from data store. Before deleting, also delete all events associated with this user and add events deleted to event history list by creating event history entity for each deleted event.

Response:
Status: 200 OK

Update a user entity:

`PUT /users/{user ID}`

Edit information of a user entity.

Input

Name	Type	Description
first_name	string	New first name of user. (optional)
last_name	string	New last name of user. (optional)
username	string	New username. (optional)
email	string	New email of user. (optional)

Check if new username provided is unique. Check if new email provided is not associated with another user entity. If either username or email is used by another user entity, return an error. If password property or events property is provided in request body, return an error. For example:

```
PUT /users/user1234
{
     “username”: “johnDoe123”
}
```

Response:
Status: 200 OK

```
{
     “first_name”: “John”,
     “last_name”: “Doe”,
     “username”: “johnDoe123”,
     “email”: “johndoe@gmail.com”,
     “password”: “password”,
     “id”: “user1234”,
     “events”: [{
               “id”: “event1234”,
               “self”: “/events/event1234”
     }]
}
```

Change password of a user entity:

`PATCH /users/{user ID}`

Change the password of a user entity.

Input

Name	Type	Description
password	string	**Required.** New password.

If request body has other properties such as first_name or username, return an error as this method is only for changing of password. If new password provided is same as old password, return an error. Example:

```
PATCH /users/user1234
{
      “password”: “newpassword”
}
```

Response:
Status: 200 OK

```
{
      “first_name”: “John”,
      “last_name”: “Doe”,
      “username”: “johnDoe123”,
      “email”: “johndoe@gmail.com”,
      “password”: “newpassword”,
      “id”: “user1234”,
      “events”: [{
                  “id”: “event1234”,
                  “self”: “/events/event1234”
      }]
}
```

Get events associated with a user entity:

`GET /users/{user ID}/events`

Display a list of event entities associated with a user entity.

Response:

Status: 200 OK

```
{“events”: [{
          “name”: “Event 1”,
          “date”: “2017-09-01”,
          “account_id”: “user1234”,
          “time”: “12:30”
          “description”: “Sample event #1”,
          “all_day”: false,
          “id”: “event1234”
          “self”: “/events/event1234”
   },
   ...
   ]
}
```

Delete events associated with a user entity:

`DELETE /users/{user ID}/events`

Delete all events associated with a user entity. Before deleting an event, create an event history entity with same properties of the event to be deleted.

Response:
Status: 200 OK

Get previous events associated with a user entity:

`GET /users/{user ID}/events/history`

Display a list of event history entities associated with a user entity.

Response:
Status: 200 OK

```
{“history”: [{
          “name”: “Event 10”,
          “date”: “2016-09-01”,
          “account_id”: “user1234”,
          “time”: “12:30”
          “description”: “Deleted event #10”,
          “all_day”: false,
          “event_id”: “event123456”,
```

```
            "id": "history1234"
        },
        ...
        ]
    }
```

Delete previous events associated with a user entity:

`DELETE /users/{user ID}/events/history`

Delete event history entities associated with a user entity.

Response:

Status: 200 OK

Get events associated with a Facebook user:

`GET /events/users/{user ID}`

Display a list of event entities associated with a user's Facebook ID.

Response:

Status: 200 OK

```
    {"events": [{
            "name": "Event 2",
            "date": "2017-09-19",
            "account_id": "facebookUser1234",
            "time": "12:30"
            "description": "Sample event #2",
            "all_day": false,
            "id": "event5678"
            "self": "/events/event5678"
        },
        ...
        ]}
```

Delete events associated with a Facebook user

`DELETE /events/users/{user ID}`

Delete all event entities associated with a user's Facebook account ID.

Response:
Status: 200 OK

Get previous events associated with a Facebook user

`GET /history/{user ID}`

Display a list of event history entities associated with a user's Facebook account ID.

Response:
Status: 200 OK

Delete previous events associated with a Facebook user

`DELETE /history/{user ID}`

Delete all event history entities associated with a user's Facebook account ID.

Response:
Status: 200 OK

Delete all event's in history:

`DELETE /history`

Delete all event history entities in data store.

Response:
Status: 200 OK

Get all event history entities:

`GET /history`

Display a list of all event history entities in data store.

Response:
Status: 200 OK

```
{“history”: [{
          “name”: “Event 10”,
          “date”: “2016-09-01”,
          “account_id”: “user1234”,
          “time”: “12:30”
          “description”: “Deleted event #10”,
          “all_day”: false,
          “event_id”: “event123456”,
          “id”: “history1234”
   },
   ...
   ]
}
```

Check user login credentials

`POST /user`

Input

Name	Type	Description
username	string	**Required.** Username
password	string	**Required.** New password.

Check if user entered login credentials match with user’s entity. If no user entity with username provided found, return error. If password provided does not match with password in data store, return an error.

Response:
Status: 200 OK

```
{
     “first_name”: “John”,
     “last_name”: “Doe”,
     “username”: “johnDoe”,
     “email”: “johndoe@gmail.com”
     “password”: “password”,
     “id”: “user1234”,
```

```
        “events”: [{
                    “id”: “event1234”,
                    “self”: “/events/event1234”
        }]
}
```