Calendar Backend Documentation

Our backend is written using node.js and the express web framework. The application is currently deployed using openstack at 130.233.42.230 behind Nginx.

Directory Structure:

- calendar-backend // root repository
    - test-data // test data conforming to the api for testing
        - event
        - calendar
        - login
        - user
        - search
    - calendar // contains the engine
        - bin
            - www // application entry point to start the server
        - components // generic modules
            - utils.js
        - routes // contains definitions of engine endpoints
            - events.js
            - calendars.js
            - users.js
        - node_modules // external dependencies
        - app.js

Installation and Running:

To install the backend clone the git repository using:
git clone https://github.com/m0hamed/calendar-backend.git

install dependencies using npm:
cd calendar-backend/calendar/
npm install

run the backend server:
node bin/www
or
npm start

API Specification:

The api uses an authentication token to authenticate users and authorise access to resources. All endpoints taking auth_token as a query strings require the presence of a valid auth_token retrieved using the login endpoint and only allow access for users to their owned resources. All endpoints respond with a json. On error the json will contain a field error with the error message. Our api specifies the following end points:

1. *post /users*

Create a new user. Request body should contain
```
{
        “username”:
        “password”:
}
```

2. post /users/login

Login using username and password. Request body should contain
```
{
        “username”:
        “password”:
}
```
The response is an authentication token that should be used for all subsequent communications.

3. get /calendars?auth_token=....

Lists all calendars belonging to user with authorization token auth_token

4. post /calendars?auth_token=....

Creates new calendar for user with authorization token auth_token. Request body should contain:
```
{
        “name”:
}
```

5. post /calendars/:id?auth_token=....

Updates a the calendar with id = :id. Request body should contain:
```
{
        “name”:
}
```

6. delete /calendars/:id?auth_token=....

Deletes calendar with id = :id

7. get /calendars/:id/events?auth_token=....

Lists all events inside the calendar with id = :id

8. post /calendars/:id/events?auth_token=....

Creates a new event in the calendar with id = :id. Request body should contain:
```
{
        “name”:
        “place”:
        “starts_at”:
        “ends_at”:
}
```

9. post /calendars/:cal_id/events/:id?auth_token=....

Updates the event with id = :id in the calendar with id = :cal_id. Request body should contain:

```
{
        “name”:
        “place”:
        “start_at”:
        “ends_at”:
}
```

10. delete /calendars/:cal_id/events/:id?auth_token=....

Deletes the event with id = :id from calendar with id = :cal_id.

11. post /calendars/:cal_id/events/search?auth_token=....

Retrieves a list of events matching the search query from the calendar with id = cal_id. Request body should contain:

```
{
        “name”:
        “place”:
        “starts_at”: {
                “from”:
                “to”:
        }
}
```