React Native 1 α

New Attempt

Due Nov 5 by 11:59pm **Points** 5 **Submitting** a text entry box
Available after Oct 26 at 4pm

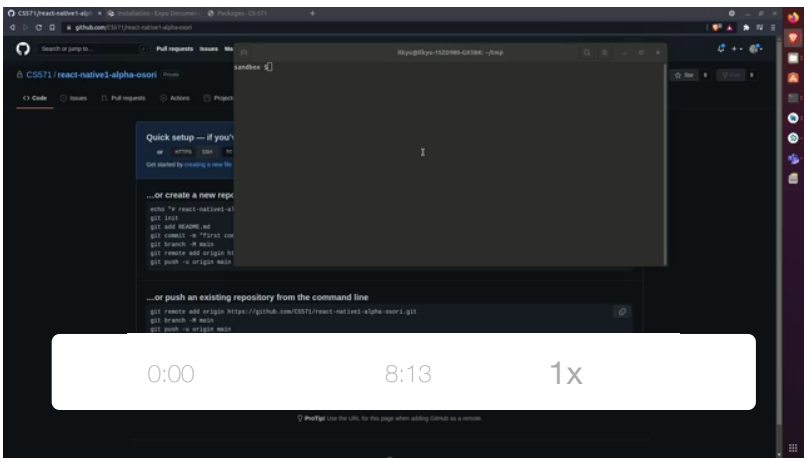
Homework: React Native 1 α (5 Points Total)

Important: The starter code for this assignment is a blank repository. See the setup video below for instructions. You must create your own expo project.

This assignment is meant to introduce you to React-Native, Expo, and the basics of using RESTful APIs. Specifically, you will be building a login screen, a signup screen, and a user profile screen. We recommend starting this assignment as soon as possible, and not waiting until the deadline! See the videos below for helpful information on setting up your project as well as what an example walkthrough may look like.

Setup Information

Starter Code: [GitHub Classroom Starter Code (https://classroom.github.com/a/pThP2J6F)](https://classroom.github.com/a/pThP2J6F)



Helpful Commands

Install Expo	Create Starter Code	Install base-64	Start the expo interface
npm install --global expo-cli	expo init react-native1-alpha-username	npm install base-64	expo start

Submission Details

When pushing your work to the repository, don't forget to include the .gitignore file automatically generated from Expo. React Native is intended to build applications that work across all platforms, though we realize that you may only be able to test on Android or iOS due to your system constraints. **Please specify in your Canvas submission which operating system you have tested on.**

When your assignment is ready for grading, please **submit your tested operating system, repository name, and latest commit hash** from GitHub Classroom. e.g. `android, react-native1-alpha-osori, 81d0462`.

Program Requirements

Problem 1 (1.5 Points)

Create a **Login View** with username and password input fields. It should:

- Use the `/login` endpoint of the API
- Provide the user with feedback if there is an error returned from the endpoint, such as "Username or password is incorrect!"
- Provide the user with functionality to switch between the login and signup pages (see problem 2).

*Hint: Provide a **Basic authentication header** (https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization#basic_authentication) in your **fetch()** (https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch). Make sure you have spent enough time playing around with the API with Postman or the **interactive docs** (http://cs571.cs.wisc.edu:5000).*

*Helpful link: **Moving between screens** (https://reactnavigation.org/docs/navigating)*

Problem 2 (1.5 Points)

Create a **Signup View** with username and password fields. It should:

- Use the `/users` endpoint of the API
- Provide the user with feedback if there is an error, such as "Username is already taken!" or "Password is too short!"
- Provide the user with functionality to switch between the login and signup pages (see problem 1).

*Helpful link: **Uploading JSON Data** (https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch#uploading_json_data)*

Problem 3 (2 Points)

Create a **Profile View** that allows the logged-in user to **view** and **edit** their first and last names and goals. It should:

- Use the `/users/<username>` endpoint of the API
- When the user edits their profile, it should be saved in the server (use the API!)
- For the time being, make this the landing page after logging in to the application. We will be expanding the application in React Native 2 **α**.

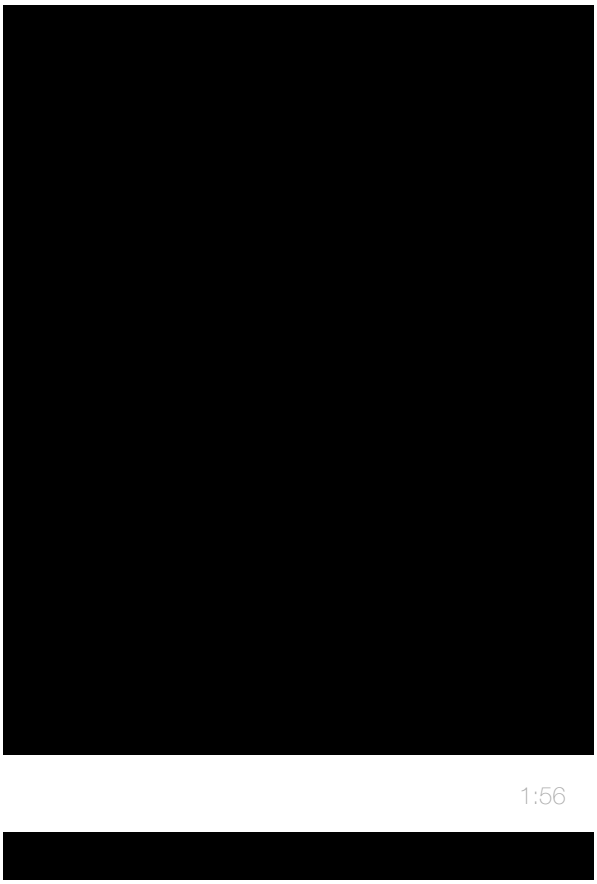
Hint: x-access-token

Helpful link: ***Authentication flows*** ***(https://reactnavigation.org/docs/auth-flow/)***

Example Walkthrough

The video below walks through a scenario where a user performs the following tasks:

1. Attempt to login with an existing user with a wrong password.
2. Attempt to login with a non-existing user.
3. Switch to sign up page.
4. Attempt to create a user that already exists.
5. Attempt to create a user with too short of a password.
6. Create a new user.
7. Login with the new user.
8. Edit profile for the new user.
9. Save changes to the profile.
10. Exit to the login screen.
11. Login with the same user.
12. View saved changes.
13. Exit to the login screen.
14. Switch to the signup screen.
15. Switch back to the login screen.



API Information

Note: You will only be using *some* of these endpoints for React Native 1 α. React Native 2 α will explore the other endpoints, such as meals, foods, and activities.

We highly recommend exploring this API using a tool such as **Postman (https://www.postman.com/)** . You may also use the interactive docs provided at **http://cs571.cs.wisc.edu:5000 (http://cs571.cs.wisc.edu:5000)** .

The following API can be accessed at `http://cs571.cs.wisc.edu:5000`

Route	Auth Required	Token Required	Get	Post	Put	Delet
/login/	✔		✔			
/users/				✔		
/users/`<username>`		✔	✔	✔	✔	✔
/meals/		✔	✔	✔		
/meals/`<meal_id>`		✔	✔		✔	✔
/meals/`<meal_id>`/foods/		✔	✔	✔		
/meals/`<meal_id>`/foods/`<food_id>`		✔	✔		✔	✔
/activities/		✔	✔	✔		

Route	Auth Required	Token Required	Get	Post	Put	Delete
/activities/`<activity_id>`			✔		✔	✔
/foods/			✔			
/foods/`food_id`			✔			

Auth and Tokens

For this API, users need to provide credentials in order to access information specific to themselves. They get these credentials by requesting tokens, which are short-lived codes which tell the server that you are who you are saying you are, without having to provide a username and password each time. For endpoints that require a token you should provide it in your requester header as `x-access-token`. The steps to get these tokens are outlined below.

Signup

This can be done with a `POST` request to the `/users` route. You will need to tell the API a bit about the user. You should provide this data in the message body (stringified) in the following form:

```
{username:<str>,                 // Required
 password:<str>,                 // Required
 firstName:<str>,                // Optional
 lastName:<str>,                 // Optional
 goalDailyCalories:<float>,      // Optional
 goalDailyProtein:<float>,       // Optional
 goalDailyCarbohydrates:<float>, // Optional
 goalDailyFat:<float>,           // Optional
 goalDailyActivity:<float>       // Optional
}
```

Only the `username` and `password` fields are required. Don't worry about the other ones for creating a user, as they can be updated later with `PUT` requests.

If the user is successfully created, you will receive a positive message back from the server. You will then need to log in with that user.

Login

You can do this via the `/login` route with a `GET` request. You will need to send the username and password in the authorization header using the header `Authorization` with value 'Basic ' + base64.encode(username:password). e.g. 'Basic bXl1c2VyOnBhc3MxMjM0' for a hypothetical user 'myuser' with password 'pass1234'.

You will receive back a token that you can use to access information from the API. The token you receive can then be added in the `x-access-token` header.

User

Users cannot query `/users`, since that would involve exposing all the other users' data. Instead, they must get/modify the information for their user separately. They do this by using the `/users/<username>` route, where `<username>` is filled in with their actual username. Suppose your username is Fred571, then you could fetch (`GET`) `/users/Fred571` to get the information about

yourself, but only if you provide the right token. Likewise, you can `PUT` to `/users/Fred571` to modify your goals, by providing the changes in the form of a `json`. You can modify the following fields:

- `pasword`
- `firstName`
- `lastName`
- `goalDailyCalories`
- `goalDailyProtein`
- `goalDailyCarbohydrates`
- `goalDailyFat`
- `goalDailyActivity`

Additionally, you can delete unused users using the `DELETE` method on the `/users/<user_id>` route. As articulated in the Final Notes, please be a good citizen and clean up after yourself.

React Native 2 α API Information

This is provided so you can get a sneak peek about the rest of the API!

User Meals

Each user can track their meals across days. These are interacted with using the `/meals` routes. Despite not including the user in the route, you will only be able to retrieve the information for your current user, since the token tells the API who you are. Unlike `/users`, you can request the full set of meals using a `GET` on `/meals`. This will return an array-like object like this:

```
{meals:[
 {id:<int>,
  name:<str>,
  date:<isostring>
 },
 {id:<int>,
  name:<str>,
  date:<isostring>
 }]
}
```

You can also access an individual meal by `meal_id` (`id`), using `GET` on `/meals/<meal_id>`. This will return the single meal data object:

```
{id:<int>,
 name:<str>,
 date:<isostring>
}
```

Posting to `/meals` creates a new meal. If not specified in ISO format, the `date` will default to the current date and time. You cannot specify the `id`, only the `name` and `date`. Using the `PUT` method is

current date and time. You cannot specify the `id`, only the `name` and `date`. Using the `PUT` method is possible for the `/meals/<meal_id>` route, such that you can modify the `name` and `date`.
Using `DELETE` on an individual meal removes the item (and all associated foods).

User Foods

You may notice that the meal doesn't have any food data. That is because they are accessed with a deeper route, `/meals/<meal_id>/foods` and `/meals/<meal_id>/foods/<food_id>`. These behave similarly to the meals, where using `/meals/<meal_id>/foods` returns the list of all foods associated with that meal:

```
{foods:[
 {id:<int>,
  name:<str>,
  calories:<float>,
  protein:<float>,
  carbohydrates:<float>,
  fat:<float>
 },
 {id:<int>,
  name:<str>,
  calories:<float>,
  protein:<float>,
  carbohydrates:<float>,
  fat:<float>
 }]
}
```

A single food can be accessed by `id`, using `/meals/<meal_id>/foods/<food_id>`, returning the single object:

```
{id:<int>,
 name:<str>,
 calories:<float>,
 protein:<float>,
 carbohydrates:<float>,
 fat:<float>
}
```

Like meals, you can modify all attributes with `PUT`, other than the `id`. Using `DELETE` on an individual food removes the item.

User Activities

By now you should see the pattern. That is the great thing about RESTful APIs! They are consistent. User activities follow the same pattern as meals, but don't have further routes like foods. You can use the routes `/activities` and `/activities/<activity_id>` in conjunction with relevant `GET`, `POST`, `PUT` and `DELETE` methods. The data structure of individual activities is as follows:

```
{id:<int>,
 name:<str>,
 duration:<float>, // Minutes
 date:<isostring>,
 calories:<float>
}
```

Using `GET` returns a list of the above objects, like in meals and foods.

Food Library

As a convenience, we have provided a set of basic foods that you can use to create and add to meals. These readonly data can be accessed with `GET` methods to `/foods` and `/foods/<food_id>`. They appear similar to foods in meal, with an additional attribute of `measure`, which tells you how much of each the nutritional values are associated with, and what the grouping name is called. For example, bread comes in `slices`:

```
{id:2,
 name:"whole wheat bread",
 measure:"slice",
 calories:69.0,
 protein:3.6,
 carbohydrates:12.0,
 fat:0.9
}
```

Final Notes

While the server API articulated above does adhere to standard conventions in security, you should not assume that it is completely secure. As such, do not use sensitive usernames and passwords for your testing. If you need to create a number of users for testing your login/signup, please be a good citizen and `DELETE` your unused users, so that the server does not get bogged down.

React Native 1 α

Criteria	Ratings			Pts
P1: Login	**1.5 pts** **Full Marks** The program reliably allows the user to specify both a username and password to login with or switch to a signup screen. A message is displayed when the user enters an incorrect username or password.	**1.5 to >0.0 pts** **Partial Marks** The program semi-reliably allows the user to specify both a username and password to login with or switch to a signup screen. A message may or may not be displayed when the user enters an incorrect username or password.	**0 pts** **No Marks** There does not exist a login view.	1.5 pts
P2: Signup	**1.5 pts** **Full Marks** The program reliably allows the user to sign up with both a username and password or switch back to the login screen. A proper message is displayed when the user enters a taken username or when the password is too short.	**1.5 to >0.0 pts** **Partial Marks** The program semi-reliably allows the user to sign up with both a username and password and may be able to switch back to the login screen. A proper message might be displayed when the user enters a taken username or when the password is too short.	**0 pts** **No Marks** There does not exist a signup view.	1.5 pts
P3: Profile	**2 pts** **Full Marks** The user is able to access their profile in which they are able to view and edit their first name, last name, and goals. The program reliably saves this data, displaying any changes when logging out and logging back in.	**2 to >0.0 pts** **Partial Marks** The user is able to access their profile in which they are able to view and possibly edit their first name, last name, and goals. The program may or may not save this data, possibly displaying changes when logging out and logging back in.	**0 pts** **No Marks** There does not exist a profile view.	2 pts

Criteria	Ratings	Pts
		Total Points: 5