OSU CS496 – Fall, 2016: Final Project
James Mills | 932704566

## Introduction:

For this project, I've set up an API to track users and the cities they've visited. If you're interacting with the API directly, e.g. making your own HTTP requests, you can start by submitting a POST request to /user with fname, lname, email, and passwd fields filled out. From there, you can reserve an authentication token by sending a POST request to /session with your email and passwd. Using the token/cookie from the previous step, you can modify and delete your account, add, modify, and delete cities. However, you cannot view another user's data because the cities are stored locally to your account and require that cookie/token to pull it up.

Additionally, the Android native application does all of this and includes a feature for adding your current location. This uses Google Play Services and your phones location functionality to retrieve your coordinates, feed them through a geocoder, and then submit them to the API.

## URL structure:

The API is currently live and available at http://35.160.5.169 with the following routes and methods:

- /
  - GET: retrieve the list of possible routes and methods
- /session
  - POST: takes json object with email and password and returns a response with a set cookie command with the user's new token
  - DELETE: deletes the user's cookie and returns a response with an order to remove the cookie from future headers
- /user
  - POST: takes json object with fname, lname, email, and passwd and creates a new user. Email must be unique to each account.
- /user/profile
  - GET: retrieves that user's full profile data based on the cookie supplied by the client
  - DELETE: deletes that user, provided the cookie matches the one on the user account
  - PUT: updates user data based on JSON data supplied in the request, i.e. fname, lname, email, and passwd.
    Behavior/restrictions: key-field pairs that are omitted will remain unmodified. And again, email must be unique to the account.
- /user/city
  - GET: retrieves the full list of cities attached to that user account based on the cookie supplied by the client host
  - POST: adds a new city with JSON payload of the form
    {"name": …, "population": …, "category": …, "country": …}
    Behavior/restrictions: city must already exist; no fields can be blank

- /user/city/<id>
    - GET: checks the user's list of cities based on client-supplied cookie for city with matching id, returning in JSON format in the response body
    - DELETE: deletes that country and removes it from its user's city list
    - PUT: updates city data based on user supplied JSON data of the form
      {"name": …, "population": …, "category": …}
      Behavior/restrictions: key-field pairs that are omitted will remain unmodified

## Account functionality:

I chose to implement my own user account system by adding a route for creating and destroying sessions. If a client host makes a POST request to /session, the API responds with a set-cookie order and a randomly-generated 64-character alphanumeric token. All other API requests require this token for authentication.

This is additionally how I handle authorization. Since all data is attached to a user in some way (i.e. user data is in a user object and cities are attached only to user objects), the only way to access a user's data is to have that user's login credentials and/or access token.

To implement all of this, I added a cookie field to the user object as well as a login-required function decorator. Everything else is just pulling up data that corresponds with the user's authentication token.

## Mobile Feature:

The Android native application includes a feature for adding your current location. This uses Google Play Services and your phone's location functionality to retrieve your coordinates, feed them through a geocoder, and then submit the city and country details to the API. This creates a new city in your user account, setting population and category with dummy values (i.e. 0 and NA).

## Video demonstration:

Due to my computer's unfortunate memory limitations. The video, which shows a single instance of the application, is split into three parts. The first demonstrates how to add new users, modify profile data, add cities, and then that this city data is unavailable to other users. The second video pretty much does the same but in reverse, showing how a second user's data is not available to the first, and the dives into the mobile feature, which allows the user to add his/her current location to the cities list. Finally, in the third video, I demonstrate how both cities and users can be deleted.

1. https://media.oregonstate.edu/media/t/1_43vla85o
2. https://media.oregonstate.edu/media/t/1_51wwu1fe/
3. https://media.oregonstate.edu/media/t/1_ebqjt31i/