Mike Mann

Final Project

Cloud Only Implementation

/user

- POST – creates a new user. A json object with the username and email fields must be included. The newly created user will be assigned an id and this new user object will be returned. Creating a new user for sake of the assignment is available by anyone since we are not authorizing each user in this assignment and we only have two pre-authorized users. This simply shows the post operation for this entity. The newly created user will be returned as a json object in the body of the request with the id, username, email, and empty watch/wish list as attributes.
- GET – returns a list of all the users. Each user will be returned in the body as an object with their username, id, email, and current watch/wish list of their desired or planned trip destinations

/user/{user_id}

- GET – returns the user account specified by the user id. Anyone can view a user, but they cannot modify a particular user entity without authorization. There will be a json object returned with the username, email, id, and watch/wish list of the user's trip destinations.
- PATCH – edits the specified user object with the corresponding user id. A json object with the username must be passed in the body of the request. Only the username may be changed via a patch request.
- DELETE – deletes the user entity with the matching user id.

/location/{token}

- POST – creates a new location entity. A json object must be passed in the body of the request with the location city name, country code ("us" or "uk" etc.), nickname (user desired note or vacation nickname). The api will then gather the current, minimum, and max temperatures of this desired location and store it with the newly created location as attributes. Because this API automatically updates the weather info of the location you create, the user must have and pass an authorized token in the path.
- GET – returns a list of all the current location entities. A list of each location will be returned in the body of the request and each will represent its own json object with the id, cityName, country, nickname, currentTemp, minTemp, and maxTemp attributes.

/location/{token}/{location_id}

- GET – returns a specified location entity matching the location_id. A json object will be returned in the body with the location id, cityName, country, currentTemp, minTemp, and maxTemp attributes.
- PATCH – updates or edits an existing location entity with the matching location_id. Only the location nickname may be edited to hold the integrity of the location. To change other

attributes of the location a PUT request must be made. A json object with the nickname field must be passed in the body of the request.

- PUT – replaces the location at the matching location_id. A json object with the new cityName, country, and nickname must be sent in the body of the request. If cityName, country, and nickname have been passed in in the request, then the api will gather the currentTemp, minTemp, and maxTemp for the new location replacing the old location. A json object with all of the location attributes, including the same id, will be returned in the body of the request.
- DELETE – deletes the location at the specified location_id. If a location is in any user watch/wish lists, it will be removed from the list.

/user/{token}/watchlist/{location_id}

- POST – adds the specified location with the matching location_id to the authorized user's watch/wish list.
- DELETE – deletes the specified location with the matching location_id from the authorized user's watch/wish list. The location will only be removed from user's list and will not be deleted entirely.

/user/{token}/watchlist

- GET – this will return the watch/wish list of the authorized user. A json object containing a list of location ids will be returned in the body of the request.
- DELETE – deletes the entire watch/wish list of the authorized list. After a delete call to this route the user's watch/wish list will be empty.

3rd Party API

The third party API used in this project is the openWeatherMap API. The openWeatherMap API can be used to gather the weather of a certain location by city name and country code, zip code, city code, or geographical coordinates. The API can return the weather for a single location or multiple locations. The API can return the current weather, 5 day/3 hour, 16 day, historical data, weather alerts, air pollution, and more for specified locations. In terms of my API utilizing the openWeatherMap API, it gathers the current weather of a single location by the city name and country code pair (i.e. New York, us), and stores the current temperature, minimum temperature, and maximum temperature of that location.