CS 492, Fall 2016, Final Project
Alicia Broederdorf

Runners Aid App

Overview

Runners Aid is the virtual aid station for runners. The app provides the location and status of water fountains and bathrooms that have been added by users. Users can search by type (bathroom or water fountain), status (open or closed), and location given a radius, to find places. They can also filter based on their favorite places. Users can then use this information when planning running routes to ensure their basic needs are met along their course, much as aid stations do during races.

Video Demo: http://web.engr.oregonstate.edu/~broedera/CS496/final_Broederdorf_RunnersAid.mp4

Cloud Backend

Google App Engine is the provider for the cloud backend, implemented using an API written in Java that interfaces with the Google Cloud Datastore NoSQL database. The database contains two main entities, a place and user. A place is created by one user while each user may have a list of zero to many places in her/his favorites. The following is a list of URIs used from the API, which are hyperlinked to more comprehensive documentation online:

http://runnersaidapi.appspot.com/api/v1/places

GET

Returns an array of places in the database. If an id field is included in the query string, only the details for that specified place are returned. Each place object returned includes id, type, name, createdDate, createdUserId, latitude, longitude, status, statusDate, and vote. This endpoint is used with the Place page when displaying the details of a selected place.

POST

Returns a place object for the newly created place. This endpoint is used with the Add Place page within the app to add new places to the database. The user provides the name, type, status, and location through a form on the Add Place page, and submits it which calls this endpoint with the required information in a url-encoded format.

PUT

Returns a place object with the newly modified details of the place. This endpoint is used when modifying the information for a specific place by users voting for a place on the Place page, as well as for modifying the information on the Edit Place page. The data is sent in a url-encoded format. If the call is made as part of the action for one of the voting buttons, the request body contains a vote field with the respective value, either up or

down. From the Edit Place page, the form data consisting of name, type, status, and location is included in the request body.

DELETE

Returns a message that the place was successfully deleted. This endpoint is not currently used by the mobile front end as it is not desired to allow users to delete places. The app is intended to be used to add places and modify their data, but not to delete places by the end users.

http://runnersaidapi.appspot.com/api/v1/users

GET

Returns an array of user objects. If an id field is included in the query string, only the details for that specified user are returned. If the query string contains both an id and fields parameter with value favorite, then the favorites list of place objects is returned for that specified user. If the favType parameter with a value of ids is included with the fields parameter with a value of favorite and id parameter, a list of favorite place ids is returned. If the query string contains a username and password, the associated user id is returned. This endpoint is used when showing place details for a specified user to display the favorite indicator on either the View Places list page or the detailed view Place page. It is also used for logging the user to store the user id locally in the app for use in subsequent API calls.

POST

Returns a user object for the newly created user. This endpoint is used to create a new user account from the Create Account page to add a new user to the database. The user provides a username and password through a form on the Create Account page, and submits it which calls this endpoint with the required information in a url-encoded format.

PUT

Returns a user object with the newly modified details for the specified user. This endpoint is used when the user is adding a place to her/his favorite list . The id for the place is sent in the request body using the url-encoded format. The user can add a place to her/his favorite list from the View Places list page or the detailed Place page by toggling on the favorite indicator (star button). This endpoint can also be used to update the data for a user, but that is not used in the front end.

DELETE

Returns a message that the place was successfully deleted from the user's favorite list. This endpoint is used to remove a place from a user's favorite list, and is called from the View Places list page or the detailed Place page when the user toggles off the favorite indicator (star button).

http://runnersaidapi.appspot.com/api/v1/search

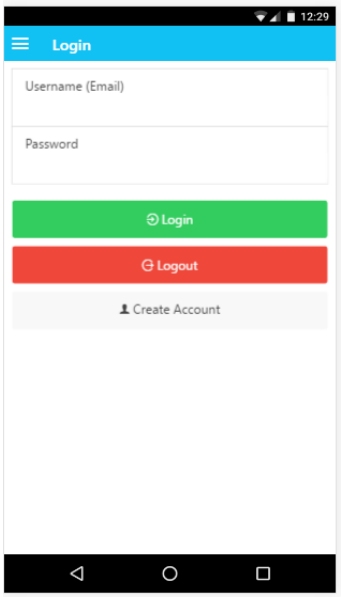
GET

Returns an array of place objects matching the provided search criteria. Optional query parameters include status, type, radius (with latitude and longitude), and user id. If no parameters are provided, an array of all place objects is returned, otherwise only those objects that match the provided query are returned. This endpoint is used for the View Places page for both the list and map views. The Filter page contains a form that provides the search criteria for the call.

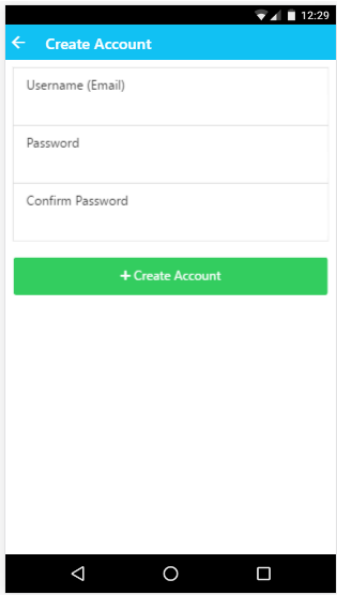
Mobile Front End

The Ionic Framework was used to create an Android app front end. Ionic leverages HTML5, CSS, and AngularJS to create a single code base for a mobile app that can then be built for Android or iOS. The mobile feature used for Runners Aid is geolocation. The user's location is used throughout the app to center the map of places, for use in the filter when searching a specified radius for places, and for adding or editing a place's location. There is also the complimentary option to specify a different address other than the user's current location for all of these functions.

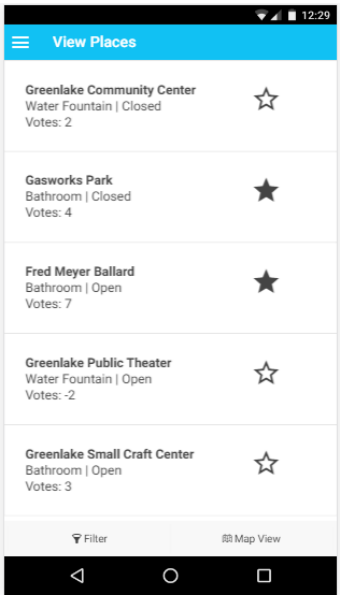
Screenshots:



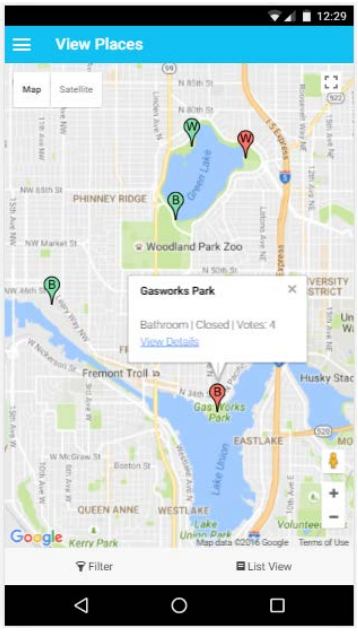
Login



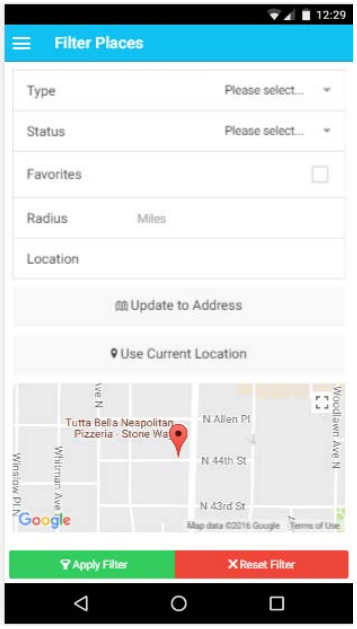
Create Account



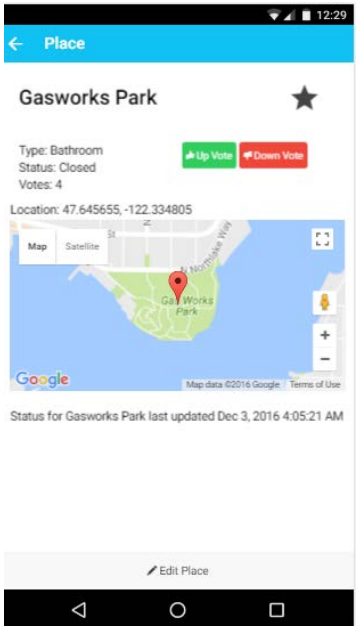
View Places – List



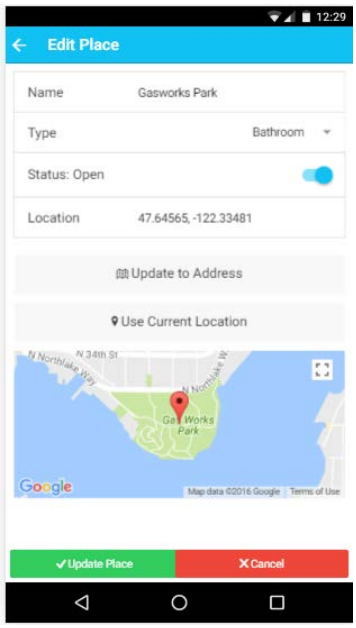
View Places – Map



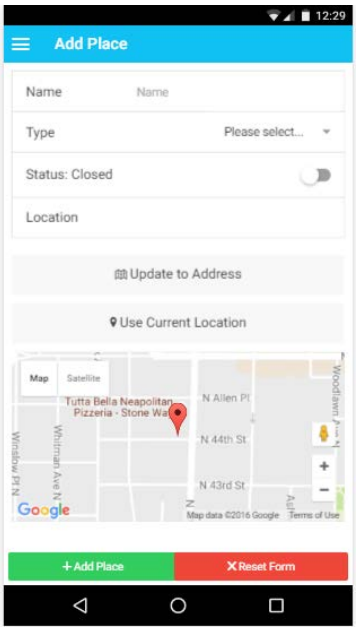
Filter



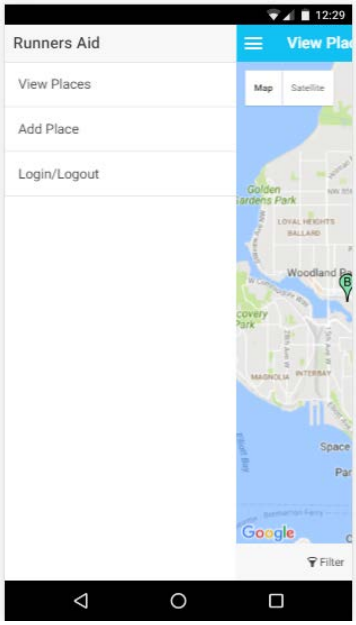
Place – Detailed View



Edit Place



Add Place



Side Menu

User Accounts

After trying to use the Google OAuth services but running into many obstacles, a simple user authentication system was created instead which does not adhere to the OAuth requirements. The UserRA entity was updated in the database to have an indexed username field and password field. From the Create Account page, a user enters a username and password. The username is specified to be an email address. The users endpoint POST method is used to send the form information in a request to

create the new user entity. Once the user has created an account, upon logging in another time she/he enters the registered username and password on the login page. If both parameters match a user entity in the database, the user id is returned from the users GET endpoint when supplying the username and password as query string parameters.

The returned user id is then saved locally in the app in the currentUserId global variable in the controllers.js file. This id is used when creating a place, showing place details to reflect if it is a favorite of the user, and when filtering the places using the favorites option. The favorite list can be used to filter the places the user sees on the View Places page, either the list or map implementation. The user manages this data from the View Places page list or from the individual Place pages by toggling the star button for the place. When the user logs out from the login page, the currentUserId is reset to -1. The app can be used by people without user accounts to view the information and vote in support of a specific status. However, to add a place or take advantage of the favorites feature a user must have an account and be signed in.