

Garcia, Randy

Geog 576

25 April 2019

National Park Trail Reporter:
A Web Servlet Application using Java, HTML, CSS & Javascript

Original Problem Statement:

Gates of the Arctic National Park is the least traveled national park in the United States with less than 10,000 visitors per year. It is often said that you can travel for several days and not come in contact with another human. While the seclusion draws many people to the park, there is still a need for visitors to easily obtain data on their trip and report new information to other park travelers. The goal is to produce an application that be used to report information on the park to include, best camping locations, dangerous animal sightings and resource locations.

Challenges with Initial Goal:

After starting the project it became obvious that there would be some technical limitations in limiting the scope of the project to Gates of the Arctic National Park. Gates of the Arctic is located in the northern region of Alaska far from network connectivity which would prove a web application useless for real time collection unless the application was programed to work offline, which was deemed outside the scope of this release. Secondly the technology that would be used to create the program had to reason to be confined to a single national park as the theory of the application could be applied to any.

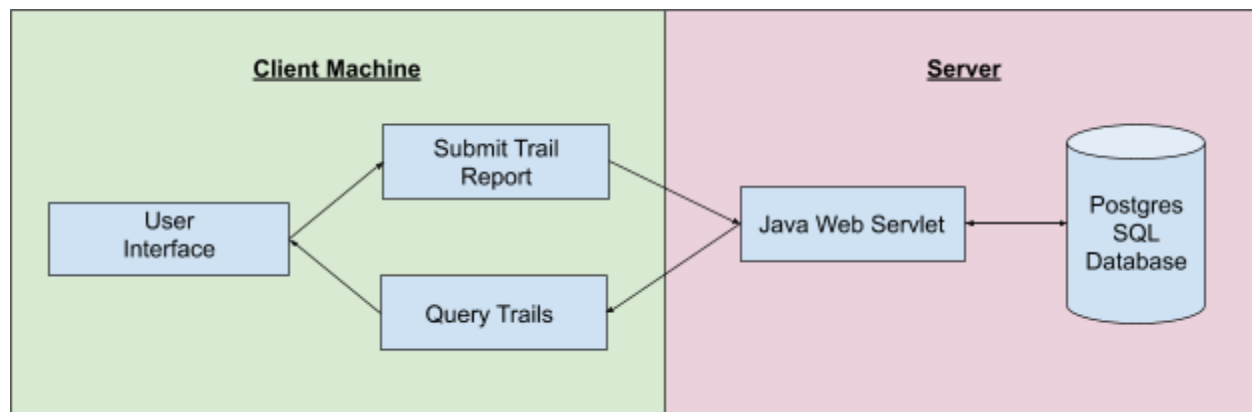
Revised Application Goal:

Create a webservlet application that allows a user to report trails in a National Park Consisting of location, Trail Name, Trail Classification, and Trail Notes. This

application should allow the user to interact directly with the map and report trail geometry with a pointing device. The application will store these user generated reports in a database and then be added to the map after posting the data.

System Design and Implementation

The workflow of the product is as follows



User Interface:

The user interface is using a combination of HTML, CSS and Javascript like most website implementations. The interface (figure 1) displays a satellite view of the world using data from an Esri Map Layer and overlaid on an Esri Javascript API Scene view that allows for 3D manipulation of a web map, called a “scene”.

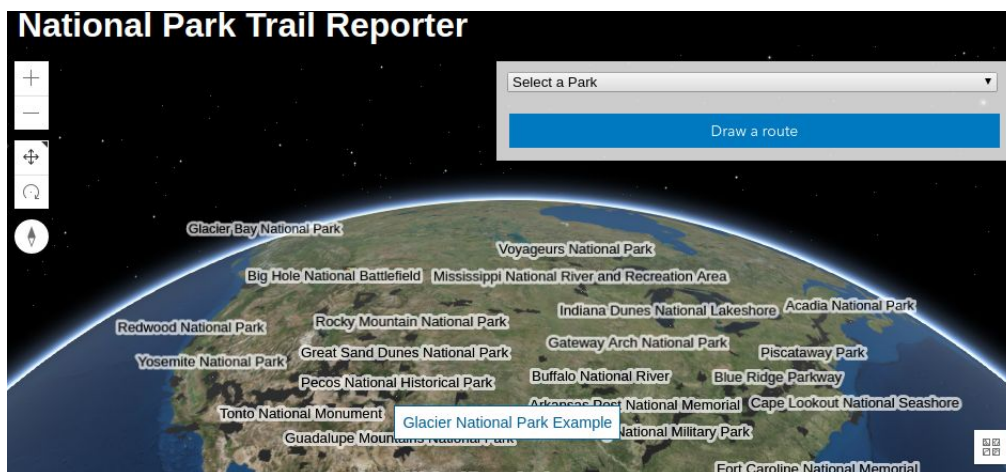


Figure 1: overall user interface

The goal of the user interface was to provide a simple but powerful tool to submit trail reports. The design focused on simplicity while still giving the user quick options to navigate the globe.

One of the main focus of the supplementary UI elements is the 3D navigation. In the top left hand corner of the page is displayed a set of options to zoom in and out changing the perspective of the user on a Z axis. The two following options allow the user to change how their pointing device will interact with the map on the x and y axis. The tools allow a user to change their perspective while maining the z value they set when they zoomed in.

On the top right of the page the webpage builds a list of National Parks contained in a GeoJSON layer provided by the National Park Service. Javascript code loops through the list and appends each park to the HTML Select element until the element contains a record for every geometry in the geoJSON:

```
var query = parkBound.createQuery();
query.outFields = ["UNIT_NAME"];
var parkslist = [];
parkBound.queryFeatures(query)
    .then(function(response) {
        parks = response.features;
        var parkselect = document.getElementById("parkselect");
        var i;
        for (i = 0; i < parks.length; i++) {
            currentPark = response.features[i].attributes.UNIT_NAME;
            parkslist.push(response.features[i].attributes.UNIT_NAME);
            parkvalue = i++;
            $('#parkselect').append('<option
value='+parkvalue+'>'+currentPark+'</option>');
        };
        //console.log(parkslist);
        sortlist();
```

```

        document.getElementById("loading").innerHTML = "Select a
Park";

        //parkselect.remove(0);
    });

```

Once a user selects the park in which they desire to add a hike, the map will zoom to the extent of a the selected park by passing new “view” values to the scene.

Underneath the park selection element is an HTML defined button that allows a user to draw a hike route. An AJAX call that is notified of the button press dispatches a function to initiate the drawing of the geometry. The user can then select points on a map with a a left click of the mouse, these points will be joined as a single path until the user double clicks, a defined function to end the drawing. On double click a separate Javascript function captures the path of the drawn route as coordinates in an array defined as “cpcoordstr”, this will then be used to pass to the database as line coordinates:

```

function crackPaths(){
    var i;
    for (i = 0; i < path.length; i++) {
        var test = webMercatorUtils.xyToLngLat(path[i][lineid],
path[i][lineid]);
        cpcoord.push(test);
        //console.log(test);
    };
    cpcoordstr = cpcoord.toString();
    bracektscoord = cpcoordstr.replace(/([-\\d.]+),([-\\d.]+),?/g, '[$1, $2],
').trim();
    cpcoordstr = cpcoordstr.replace(/([-\\d.]+),([-\\d.]+),?/g, '$1 $2,
').trim();
};

```

Once the user finished their drawing the upper right drawing div is expanded to show three more input fields where they can submit info about their hiking trail report (figure 2).

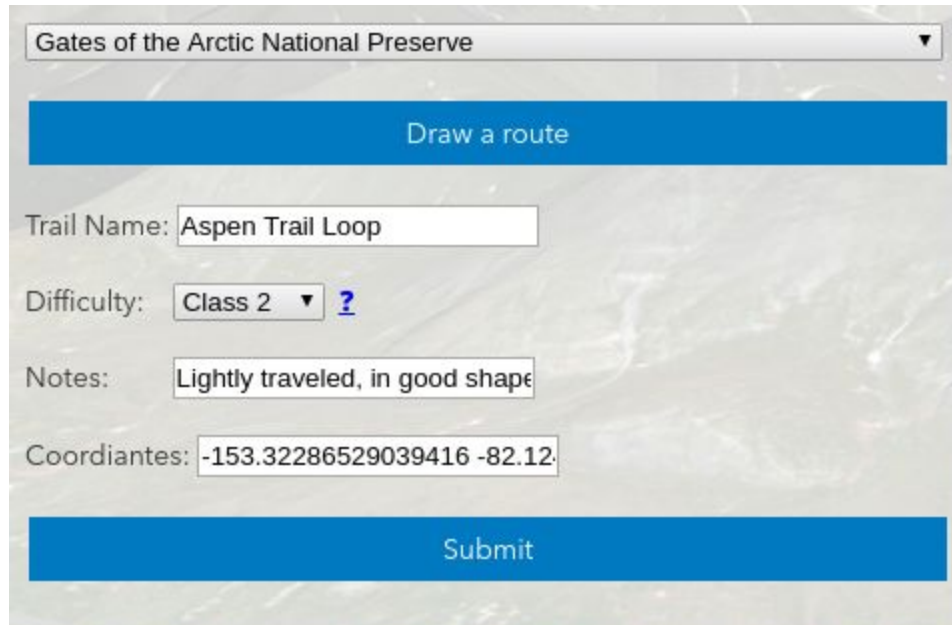
The image shows a web form for reporting a hiking trail. At the top, there is a dropdown menu with the text "Gates of the Arctic National Preserve". Below this is a blue button labeled "Draw a route". Underneath the button, there are four input fields: "Trail Name:" with the text "Aspen Trail Loop", "Difficulty:" with a dropdown menu showing "Class 2" and a question mark icon, "Notes:" with the text "Lightly traveled, in good shape", and "Coordiantes:" (note the typo) with the text "-153.32286529039416 -82.12". At the bottom of the form is another blue button labeled "Submit". The entire form is set against a light gray background with a faint map of a snowy landscape.

Figure 2: Expanded route drawing div showing options for the user report

In order to keep the report simple it was decided to only use Trail Name, Trail Class and Notes. These three provided sufficient info as a survey into trails. By separating Name and Class we are giving the ability to query against these in the future. When the user submits the report the data is then posted to the server using a POST method:

```
function sendcoord(){
    $.ajax({
        url: 'RunQuery.jsp',
        type: 'POST',
        data: { "tab_id": "0", "report_type": report_type, "report_name":
report_name, "report_notes": report_notes,
        "report_object": report_object},
        success: function(data){
            $.each(data, function(i, name) {
                alert("report successfully submitted");
            });
        }
    });
}
```

```

});
document.getElementById("create_report_form").reset();
onPlaceChanged();
},
error: function(xhr, status, error) {
    alert("An AJAX error occured: " + status + "\nError: " + error);
    console.log(error);
}
});
};

```

Trails are queried and placed onto the map using a Esri Javascript Graphics layer that is colored using a unique value renderer based on the class difficulty (figure 3):

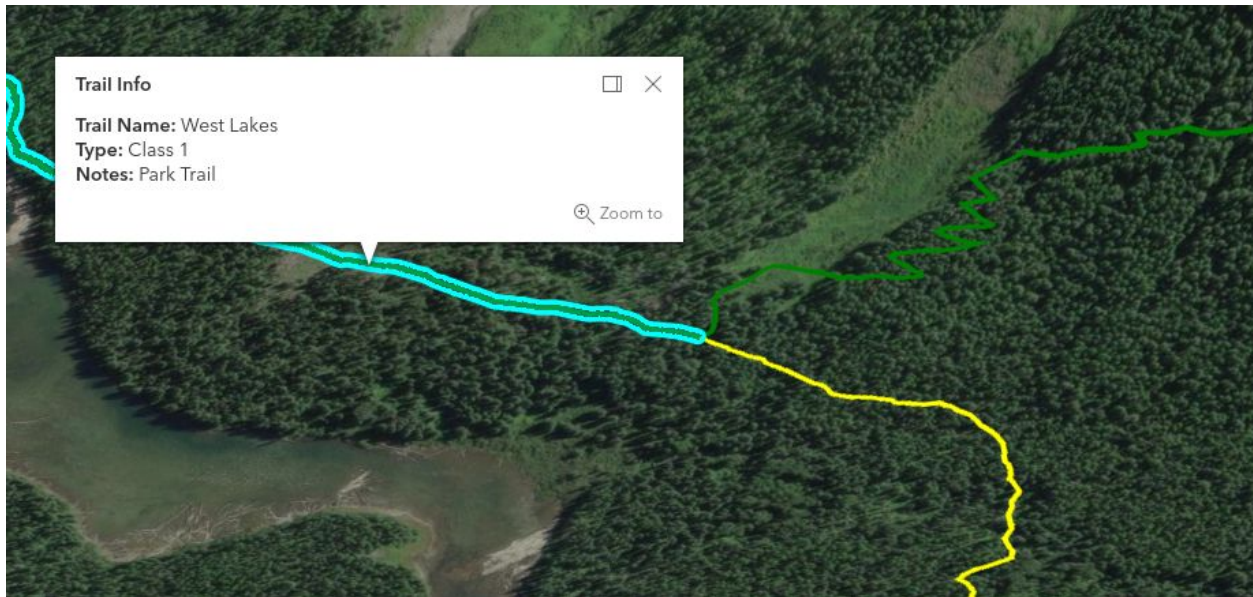


Figure 3: A view of three trails with varying class values

Server Implementation:

As mentioned above, once the user submits the report, the url parameters defined as:

“...?report_name=[name]&report_type=[class]&report_notes=[notes]&report_object=[coordiantes]”

The runQuery.jsp file parses the URL for values as defined below:

```
String report_type = request.getParameter("report_type");
```

```

String report_name = request.getParameter("report_name");
String report_notes = request.getParameter("report_notes");
String report_object = request.getParameter("report_object"); //maybe cast
error here?

System.out.println("A report is submitted!");

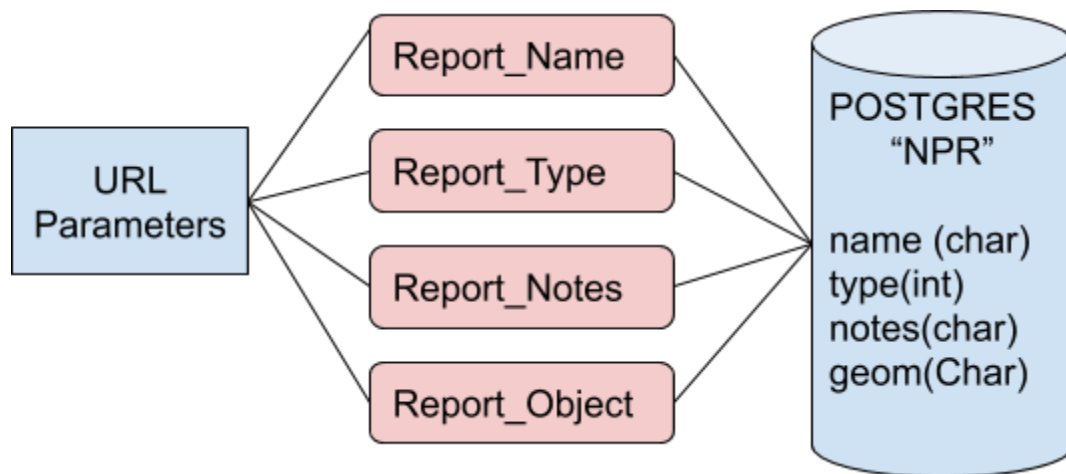
if (report_type != null) {report_type = "'" + report_type + "'";}
if (report_name != null) {report_name = "'" + report_name + "'";}
if (report_notes != null) {report_notes = "'" + report_notes + "'";}
if (report_object != null) {report_object = "'" + report_object + "'";}
if (report_type != null && report_name != null) {
    // create the contact
    sql = "insert into placepoint (type, name, notes, object) " +
        "values (" + report_type + "," + report_name + "," +
report_notes + ","
        + report_object + ")";
    dbutil.modifyDB(sql);
}

```

The above code passes the values into a SQL statement that is then passed to a java method “modifyDB” that is used to add the values into the Postgres Database on the server.

Database Implementation:

The data from the java method is passed to a Postgres database that uses the following schema



Because the application only records one type of object, hiking routes, there is little need for a more extensive database. In the future as the reporter is expanded to support additional features and report types the database schema will need to be updated to reflect that. The largest issue when storing the data are the coordinates of the reported routes. Route coordinates are stored in a string that are xy pairs of each points along the line, based on the length of the route the database may potentially run up against the 1gb limit but that is unlikely. There was also thought in storing the data inside of an Esri Hosted Feature Service or a hosted WFS this would allow the application to not rely on a server and take some of the technology load away from managing the service. The Esri Feature Service / WFS would take POST request that contains the report attributes and coordinates in JSON format. As the current code has a POST function it would be an easy swap to an more reliability hosted geospatial service and would increase reliability and scalability across the application.

Future Features:

Due to the limited time and scope of this release there were a couple features left off in order to make the deadline. These items are on the roadmap for a future release.

1. Add the ability to upload a GPX file
2. Add the ability to edit a route
3. Add the ability to display routes of a certain type
4. Mobile friendly version
5. Migrate to managed geodatabase
6. Add image upload to trail reports

Case Study #1

Who: James Jackson

What: Outdoor enthusiast

Why: James, an avid outdoorsman, is traveling to Rocky Mountain National Park In September of this year. He is hoping to hike as many trails as he can for his 4 day stay in the park. With the current resources available he has been unsuccessful in finding an overview of trails and trail conditions in the park he is visiting. He has in the past hiked many class 2 trails, he knows that he is looking to find some more class 2 and class 3 trails to hike while on his trip.

Results: James uses the National Park Trail Reporter to see the trails in Rocky Mountain National Park, the map takes him to the overall extent of the park where he can see the latest data for user trails he is looking for trails that are orange and yellow knowing those are of the classification he desires. He find a location where he can best spend his four days traveling along these various routes.

Case Study #2

Who: Rachel Maize

What: Trail Conservationist

Why: Rachel spends her time looking after trails and ensuring that they are not degrading overtime. She is looking for a way to add trail routes to maps of national parks in the hopes that she can help others stay on the trail and not degrade the wildlife habitat overtime. She also is looking to report the poor condition of trails she had just inspected in Glacier National Park as a way to document what parts of the part need to be reported for repair.

Results: She uses the National Park Trail Reporter to define the location of trails in the southern part of the park where there have been incidents of people not staying on the trails as they seem to be poorly marked. She uses the geometry editor and the 3D satellite imagery to produce an accurate representation of the trail on that map, she

adds notes to the trails stating that the trails are poorly marked but it would be best for travelers to follow the trees marked with yellow tags as they best follow the route.

Data:

Name	Source	Description
Park Unit Boundaries	https://services1.arcgis.com/fBc8EJBxQRMchlei/arcgis/rest/services/NPS_Park_Boundaries/FeatureServer/0/query?outFields=*&where=1%3D1	Extents of National Park Service Park boundaries, Layer was used to display and build list of National Parks supported in the reporter
Glacier National Park Trails	https://opendata.arcgis.com/datasets/4746b25f893a4e25b94ab571e8c4cf3d_0.geojson	GeoJSON that contains trails located within Glacier National Park, this is used as an example dataset that can be created with the reporter
User Generated Trails	User Generated, stored within Postgres database on NPR server	Geometry and attributes for all user generated hiking trail reports

Libraries and tools:

Name	Source
Esri JS API 4.7	https://developers.arcgis.com/javascript/
jQuery 2.0.2	https://jquery.com/
Postgres database	https://www.postgresql.org/
PostGIS Extension	https://postgis.net/
Tomcat Server	http://tomcat.apache.org/