# Building an Interactive Park Map

**Project 2 Report**

2020-04-08

Cory Leigh Rahman, University of Wisconsin-Madison, Master's in GIS & Web Map Programming, GEOG 777: Capstone in GIS Development

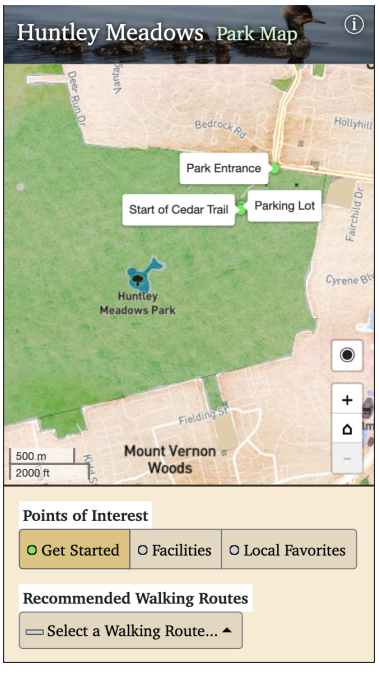
# Introduction

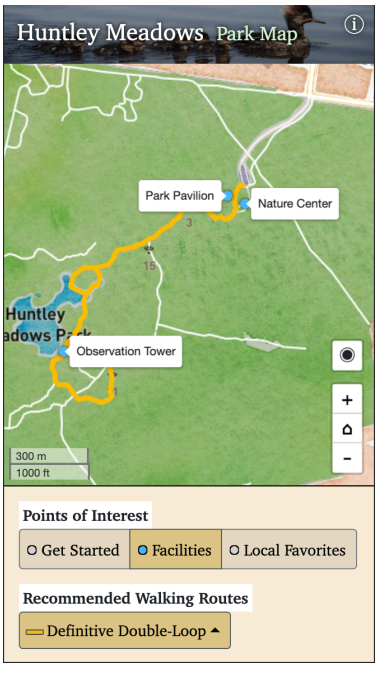
An interactive park map application was created for Huntley Meadows Park in Fairfax County, Virginia. The map application primarily helps users navigate the park and discover places to visit within the park.

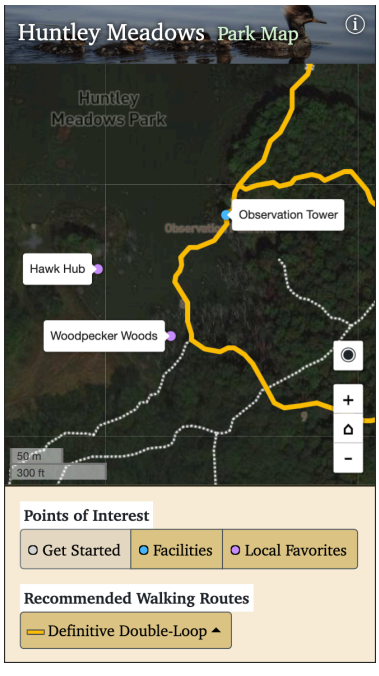
## About Huntley Meadows Park

Huntley Meadows Park is a 1452-acre park located just south of Washington D.C. in Fairfax County, Virginia. The land was originally purchased by George Mason in 1757 but is now owned by Fairfax County.The park is now a vital, biodiverse forest and wetland, home to over 30 species of mammals, 50 species of reptiles, amphibians and dragonflies, 200 species of birds and 600 species of plants. The park is frequented by over 200,000 visitors per year.

# User-Centered Design



User Stories:

- As an interested local, I want easy guidance on how to get started
- As a visitor, I want to know where I can explore in the park
- As a visitor, I recommendations for things to see within the park
- As a visitor, I want to know where the restrooms and other facilities are
- As a frequent visitor, I want to point out places of interest to others

All design decisions stem from the purposes outlined in these User Stories.

## Responsive Design

The Park Map was designed with a mobile-first approach so that users can use the app on their phones while physically exploring the park. Active development was done using the iPhone 6/7/8 screen size to ensure a mobile-first approach. In addition, nearly all controls were kept towards the bottom of the screen so that mobile users have an easier time interacting with them using their thumbs.

## Map Layers

Most map layers are hidden on the initial view in order to avoid overwhelming the user; however, all these map layers are available and help fulfill the app purposes outlined in the User Stories.

- Geting Started Points of Interest (Initial View)
- Facilities
- Local Favorites Points of Interest (User-added)
- Definitive Double-Loop Walking Route
- Boardwalk Bend Walking Route
- Forest Finish Walking Route
- Mapbox Basic Edited/Transparent (TileLayer Overlay, Initial View)
- Stamen Watercolor (Basemap, Initial View)
- Esri World Imagery (Basemap)

The basemap is actually three different TileLayers working together to create a powerful and dynamic backbone for the application.

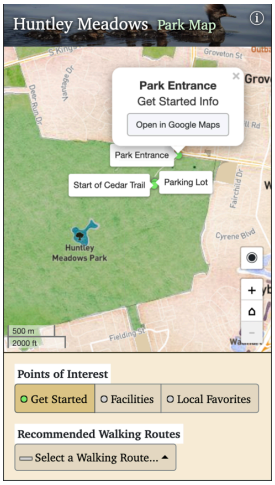
| **Basemap** | **Zoom Levels** | **Characteristics** |
|---|---|---|
| 1. MapBox Basic Edited Overlay | 14-18 | Labels, paths, landmarks |
| 2. Stamen Watercolor | 14-16 | Beautiful watercolor-style art |
| 3. Esri Imagery | 17-18 | Detailed park imagery |

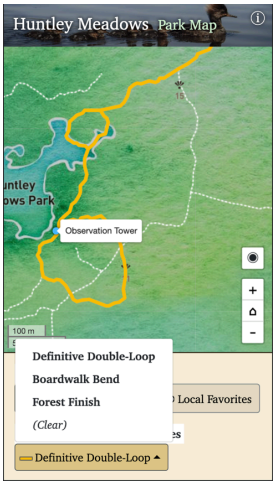
The user is limited to zoom levels 14-18 so that they do not get lost. Each zoom level available to the user has been carefully thought-out and purposefully included.

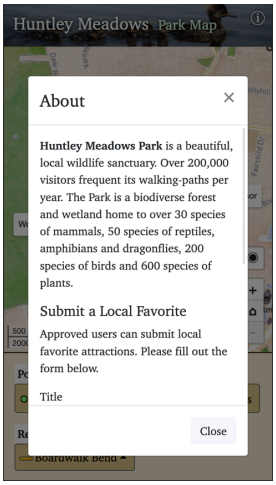
| Zoom Level | Basemap Information |
|---|---|
| 14 | Overview of the park Stamen Watercolor's park outline |
| 15 | Paths appear throughout the park map |
| 16 | A new watercolor texture appears with more crisp paths |
| 17 | Imagery and path labels appear |
| 18 | Imagery is easier to see |

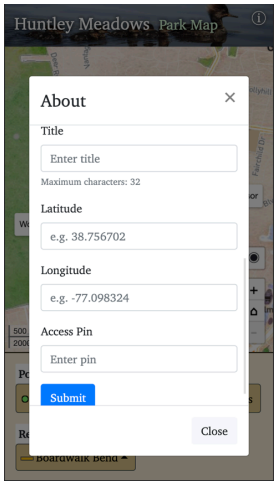
## Application Functions

- **Checkbox-button style layer toggles for Points of Interest (POI) layers**
    - These toggles not only make it easy for mobile users to quickly toggle POI layers, but they also act as a legend, dynamically highlighting and coloring to clearly show which layers are active.
- **Open in Google Maps**
    - Selecting any in-map tooltip layer allows the user to open the coordinates for that point in Google Maps, allowing the location to be easily shared and navigated-to. For example, a user could select "Park Entrance" to easily get driving directions from Google.
- **Drop-up style layer toggle for Recommended Walking Route layers**
    - This toggle style prevents the user from getting confused by overlapping routes, and also acts as a legend when enabled.
- **Return Home**
    - Selecting the Home button in-between the zoom buttons returns the user back to the initial map view if they get lost
- **Show Me Where I Am**
    - The Geolocation button above the home/zoom controls allows the user to see where they are on the map indicated by a blue, pulsing circle-marker. This feature is crucial as it allows users to see if they're on the right track, or near points of interest.
- **User Local Favorite Submissions**
    - The app allows frequent visitors to submit their own "Local Favorites" for others to discover.
- **About Modal**
    - The About modal (information window) provides the user with additional information about the park, as well as providing space for the User Submission form.










# Implementation

Tech Stack:

| Technology | Description |
|---|---|
| Leaflet | Interactive mapping JavaScript Library |
| Bootstrap | Front-end UI Component Library |
| Angular | Front-end JavaScript framework |
| NPM | Front-end package manager |
| Node Express Server | Server for API for database interaction |
| PostgreSQL | Database system |
| Heroku | Platform & host |
| VS Code | IDE / code editor |
| QGIS | Desktop GIS software |

## Front-End

Angular, Leaflet, and Bootstrap were the clear picks for front-end technologies. Angular provided the framework for ease of development, Leaflet is a world-renowned mapping tool, and Bootstrap provided quick and customizable UI components. In addition, Leaflet's rich selection of plugins were particularly useful for features such as the Return Home and Locate Me functions.

```
import { Component, OnInit } from '@angular/core';
```

```
import * as L from 'leaflet';
import 'leaflet.zoomhome/dist/leaflet.zoomhome.min.js';
import 'leaflet.locatecontrol/dist/L.Control.Locate.min.js';
import { HttpClient } from '@angular/common/http';
import { FormBuilder, FormGroup } from '@angular/forms';
import { NgbModal } from '@ng-bootstrap/ng-bootstrap';
```

These are the import statements from the `ParkMapComponent` TypeScript file. Features were used heavily from Angular, Leaflet and Bootstrap.

## Back-End

The back-end consists of a Node Express server, a PostgreSQL database, and Heroku as the host and platform.

### Server Code

The Node Express server handles routing and serving of the build. First, API routes are caught:

```
/* Set non-angular routes (server / api calls) */
app.get('/getroutes', async function (req, res) {
  queryPrimaryDatabase(`select * from recommended_routes;`, functio
n (err, dataresponse) {
    if (dataresponse && dataresponse.rows) {
      res.status(200).json(dataresponse.rows);
    } else {
      res.status(400).end();
    }
  })
});
...
```

Finally, all other routes are forwarded to Angular:

```
...
/* Send all other routes to Angular app */
app.get('*', function (req, res) {
  res.sendFile(path.join(__dirname, '../dist/huntley-meadows-park-m
ap/index.html'));
});
```

In order to support continuous development, I used both `ng build --watch` and `heroku local`; this combination allowed the Node Express server build started by heroku to be

automatically updated by Angular as I make changes.

As for database connections, the one and only function needed was:

```
/**
 * Asynchronous function which queries the database and returns the r
 * @param {string} queryString - SQL Query String
 * @param {function} callBackFunction - requires parameters (err, res
 */
async function queryPrimaryDatabase(queryString, callBackFunction =
 (err, res) => {}) {
  const pgPsqlClient = new Client({
    connectionString: process.env.DATABASE_URL,
    ssl: true,
  });
  await pgPsqlClient.connect();
  return pgPsqlClient.query(queryString, async (err, res) => {
    callBackFunction(err, res);
    pgPsqlClient.end();
  });
}
```

## Database

Here are the real SQL commands used to architect the database:

```
SET timezone = 'America/New_York';

create table recommended_routes (
  route_name text,
  geojson text
);

create table poi_get_started (
  poi_name text primary key,
  latitude decimal(9, 6),
  longitude decimal(9, 6)
);

create table poi_facilities (
  poi_name text primary key,
  latitude decimal(9, 6),
  longitude decimal(9, 6)
);

create table poi_local_favorites (
  poi_name text primary key,
  latitude decimal(9, 6),
```

```
  longitude decimal(9, 6),
  created_time_stamp timestamptz
);

ALTER TABLE poi_local_favorites ALTER COLUMN created_time_stamp SET
 DEFAULT now();
```

An ER Diagram was not developed because no relationships were necessary between the tables.

Here is an example of SQL being constructed in the server and used to update the database:

```
queryPrimaryDatabase(`
    insert into poi_local_favorites (
      poi_name, latitude, longitude
    ) values (
      '${inputTitle}', ${inputLatitude}, ${inputLongitude}
    );
  `, function (err, response) {
  if (response) {
    res.status(200).json(response);
  } else {
    res.status(400).end();
  }
})
```

# Conclusion

The Huntley Meadows Park Map application managed to fulfill all the purposes outlined by the User Stories. It allows users to easily figure out how to get to the park, how to navigate it, and what to see while there. It also provides locals the ability to submit their favorite locations for others to see.

The app manages to convey a large amount of information without overwhelming them. It will be an asset to any prospective Park visitor.