Project 3 - Visualizing Surf Location Forecasts

Team Members:

- Tommy Hvidhyld
- Taylor Marino
- Conor Chapman
- Evgeniya Kardashov

Project Scope

We compiled the most popular and known surf locations in California to create an application showing the forecast for each location.

The data we used included Wave Height, Air Temperature, Wind Speed, Water Temperature, Cloud Cover, and Visibility.

Back-End Data

Research and Extraction:

Using SurfSpot CSV file we pulled in all locations in California. In order to find the forecast for the day of each location we used the StormGlass API.

```
url = 'https://api.stormglass.io/v2/weather/point'
for x, y in zip(surfcali1['latitude'], surfcali1['longitude']):
    try:
        lat = (f'{y}')
        lng = (f'{x}')

        params ={
            'lat': lat,
            'lng': lng,
            'params': ','.join(['waveHeight', 'airTemperature', 'windSpeed', '
            'start': start.to('UTC').timestamp(),  # Convert to UTC timestamp
            'end': end.to('UTC').timestamp()  # Convert to UTC timestamp
        }
```

```
surf_cali_df = surf_cali.loc[(surf_cali['latitude']> -123) & (surf_cali['latitude']< -113)]
```

```
surf_cali = surf.loc[(surf['longitude']< 43) & (surf['longitude']>32)]
```

Back-End Data

Data Transformation and Load:

The data pulled from the API was put into a dataframe of the columns we needed. From there it was written to a SQLite Database.

From the SQLite Database the data was rendered through a flask app into JSON format

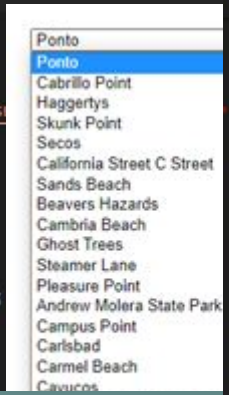
	index	spot	longitude	latitude	spot_id	wind_speed	wind_direction	wave_height	air_temp	water_temp	cloud_cover
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	41	Ponto	33.088	-117.314	1149	1.58	65.06	1.92	13.88	15.0	4.8
2	42	Cabrillo Point	33.707	-118.285	284	2.41	48.54	1.18	13.51	14.54	6.0
3	43	Haggertys	33.803	-118.401	282	2.41	48.54	1.79	11.14	14.5	5.0
4	44	Skunk Point	33.983	-119.984	1251	5.6	338.02	3.36	12.76	13.81	5.0
5	45	Secos	34.038	-118.879	278	3.75	357.42	1.84	10.67	14.19	15.0
6	46	California Street C Street	34.273	-119.303	275	3.25	346.39	1.66	2.07	13.9	100.0
7	47	Sands Beach	34.409	-119.882	268	8.76	323.91	2.33	5.01	13.96	100.0
8	48	Beavers Hazards	34.461	-120.054	267	8.76	323.91	2.33	5.01	13.96	100.0
9	50	Cambria Beach	35.582	-121.121	261	3.53	1.98	2.99	7.13	13.25	26.4
10	51	Ghost Trees	36.596	-121.977	825	8.51	322.74	3.34	11.97	12.5	5.0
11	52	Steamer Lane	36.952	-122.026	163	8.51	322.74	3.34	7.63	12.19	92.9
12	53	Pleasure Point	36.955	-121.972	644	8.51	322.74	3.34	7.63	12.19	92.9

```python
@app.route("/")
def welcome():
    """List all available api routes."""
    return (
        f"Available Routes:<br/>"
        f"/api/v1.0/surf<br/>"
    )

@app.route('/api/v1.0/surf')
def names():
    # Create our session (link) from Python to the DB
    session = Session(engine)

    """Return a list of all surf data"""
```

Front-End: *Normalizing Data in JS for Charts*

Create dropdown

```
app.js M
app.js > chartCreate > then() callback > series > data

// setting up url from our render file
const url = "https://surf-app.onrender.com/api/v1.0/s

// selecting the data for the dropdown
d3.json(url).then(function(data) {
  let selected = d3.select("#selDataset");
  console.log(data)
  for (let i = 0; i < data.length; i++) {
      selected.append('option').text(data[i].spot);
}});
```



Start function using index from selected surf spot then get array

```
// what we want to do: create a chart when new surf spot is selected
function chartCreate(spots) {
  d3.json(url).then(function(data){
    let spotIndex = data.findIndex(data => data.spot === spots);
    let weather = data[spotIndex];
    let weatherData = [];
    let weatherLabels = [];
    let removeValFrom = [2,3, 4, 5, 6, 7,8,11,13];
    for (const key in weather) {
      {
      weatherLabels.push(key)
      }
    };
    weatherLabels= weatherLabels.filter(function(value, index) {
      return removeValFrom.indexOf(index) == -1;
    });
```

```
app.js M
app.js > chartCreate > then() callback > spotIndex > data.findIndex() callback

for (const key in weather) {
  {
  weatherLabels.push(key)
  }
};
weatherLabels= weatherLabels.filter(function(value, index) {
  return removeValFrom.indexOf(index) == -1;
});
for (let [key, value] of Object.entries(weather)) {
  if (key == "air_temp") {
    if (value <= 10.00 || value >= 43.00 )
    {
      weatherData.push(1)
    }
    else if (value >= 23.00 && value <= 27.00){
        weatherData.push(5);
    }
    else if (value <= 22.99 && value >= 18.00) {
      weatherData.push(4)
    }
    else if (value <= 17.99 && value >= 14.00){
      weatherData.push(3)
    }
    else if (value <= 13.99 && value >= 10.01){
      weatherData.push(2)
    }
    else{
      weatherData.push(0)
    }
  }
  if (key == "cloud_cover") {
    if (value >= 80.00 )
    {
      weatherData.push(1)
    }
    else if (value >= 0.00 && value <= 19.99){
        weatherData.push(5);
    }
    else if (value >= 20.00 && value <= 39.99) {
      weatherData.push(4)
    }
    else if (value <= 59.99 && value >= 40.00){
      weatherData.push(3)
    }
    else if (value <= 79.99 && value >= 60.00){
      weatherData.push(2)
    }
    else{
      weatherData.push(0)
    }
  }
```

Normalize data variables on scale of 0-5 with 0 (worst) and 5 (best) based on research from surf guide websites for ideal surfing weather conditions for wind speed, air temp, visibility, wave height, cloud cover, & water temp

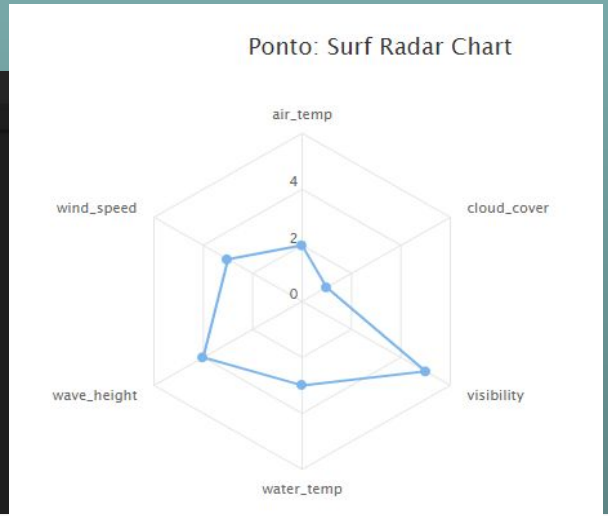
Initialize initial chart when page loads & click function to change chart when dropdown selected

```
app.js > chartCreate > then() callback > series >

// this activates when someone selects a
function optionChanged(spots) {
    chartCreate(spots);
};

//initializing charts right off the bat
function init() {
  d3.json(url).then(function (data) {
      chartCreate(data[0].spot);
  })
};
init();
```

Visualizations: *Location Ratings*

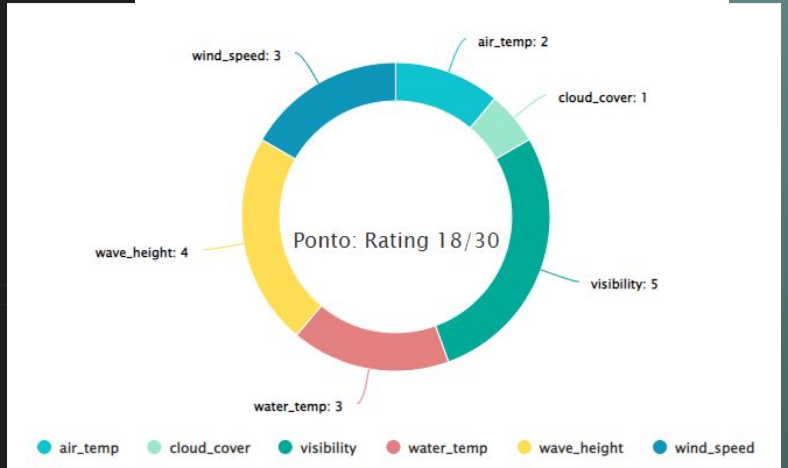
Highcharts.JS: Radar/Spider Chart

```
app.js M
app.js > chartCreate > then() callback > legend > layout
165  Highcharts.chart('container', {
166
167    chart: {
168      polar: true,
169      type: 'line'
170    },
171
172
173    title: {
174      text: `${spots}: Surf Radar Chart`,
175      x: 0
176    },
177
178    pane: {
179      size: '80%'
180    },
181
182    xAxis: {
183      categories: weatherLabels,
184      tickmarkPlacement: 'on',
185      lineWidth: 0
186    },
187
188    yAxis: {
189      gridLineInterpolation: 'polygon',
190      lineWidth: 0,
191      min: 0
192    },
193    legend: {
194      align: 'right',
195      verticalAlign: 'middle',
196      layout: 'vertical'
197    },
198
199    series: [{
200      name: `${spots} Score`,
201      data: weatherData,
202      pointPlacement: 'on'
203    }],
```




```
app.js M
app.js > chartCreate > then() callback
158      let sum = 0;
159      weatherData.forEach(item => {
160          sum += item;
161      });
162    console.log(sum);
```

Sum variable to represent the summed rating for location



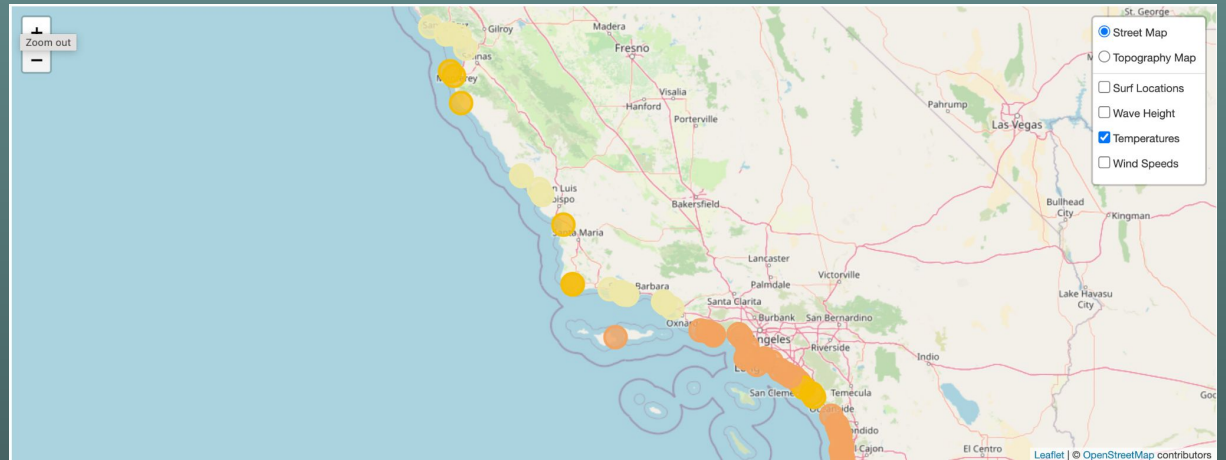

```
app.js M
app.js > chartCreate > then() callback
226
227  Highcharts.chart('container2', {
228    colors: ["#0fc4d0", "#9be7cb", "#00aa99", "#E48080", "#ffdd54", "#0D96BA"],
229    chart: {
230        type: 'pie'
231    },
232    title: {
233        text: `${spots}: Rating
234        ${sum}/30`,
235        align: 'center',
236        verticalAlign: 'middle',
237        floating: true,
238        padding: 5
239    },
240    plotOptions: {
241        pie: {
242            allowPointSelect: true,
243            cursor: 'pointer',
244            dataLabels: {
245                enabled: true,
246                format: '{point.name}: {y}'
247            },
248            showInLegend: true
249        }
250    },
```

Highcharts.JS: Donut Chart with summed rating for how ideal surf spot is

Map

Creation:

Using OpenStreetMap and leaflet we created a map of California with markers that depict different important factors for surfers. This data was pulled using d3 from our Flask API. The goal was to give users a quick indication on the best surf spots. This was done by having four overlays, and the markers depict different information that give the user an indication on the quality of the spot.



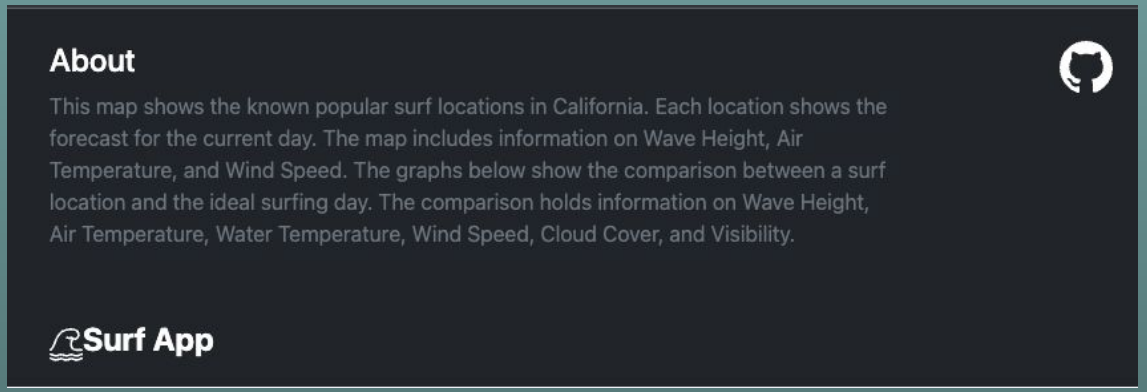
Marker Functions

```
function createMarkers(response) {

  // Pull the "stations" property from response.data.
  // let stations = response.data.stations;

  // Initialize an array to hold markers.
  let surfMarkers = [];
  let waveMarkers = [];
  let tempMarkers = [];
  let windMarkers = [];

  function circleSize (wave) {
      return wave * 5000
  };

  function circleSizeWind(wind) {
      return wind * 2000
  }

  function circleColor (temp) {
      if (temp > 14)
          return "#C34A2C";
      if (temp >= 10)
          return "#F4A460";
      if (temp >= 7)
          return "#F6BE00";
      if (temp >= 3)
          return "#EEE8AA";
      else
          return "#9AFEFF"
  };

  function circleFill (wind) {
      if (wind > 12)
          return 1;
      if (wind >= 8)
          return .75;
      if (wind >= 4)
          return .5;
      if (wind >= 1)
          return .2;
      else
          return .1;
  };
```

Creating Markers based on rating

```
  // Loop through the stations array.
  for (var i = 0; i < response.length; i++) {
    let data = response[i];
    // For each station, create a marker, and bind a popup with the station's name.
    let marker = L.marker([data.longitude, data.latitude])
      .bindPopup("<h2>" + data.spot + "<h2><h4>Wave Height: " + data.wave_height +  " M</h4><h4>Air Temperature: " + data.air_temp + " °C</h4><h4>Wind Speed: " + data.wind_speed + " m/s</h4>");

    // Add the marker to the array.
    surfMarkers.push(marker);

    let waveMarker = L.circle([data.longitude, data.latitude], {
          radius: circleSize(data.wave_height),
          fillColor: "#00AA99",
          color: "#E48080",
          fillOpacity: 1,
          stroke: true
    }).bindPopup("<h2>" + data.spot + "<h2><h4>Wave Height: " + data.wave_height +  " M</h4><h4>Air Temperature: " + data.air_temp + " °C</h4><h4>Wind Speed: " + data.wind_speed + " m/s</h4>");
    waveMarkers.push(waveMarker);

    let tempMarker = L.circle([data.longitude, data.latitude],{
          radius: 12000,
          fillColor: circleColor(data.air_temp),
          color: circleColor(data.air_temp),
          fillOpacity: .8,
          stroke: true
    }).bindPopup("<h2>" + data.spot + "<h2><h4>Wave Height: " + data.wave_height +  " M</h4><h4>Air Temperature: " + data.air_temp + " °C</h4><h4>Wind Speed: " + data.wind_speed + " m/s</h4>");;
    tempMarkers.push(tempMarker);

    let windMarker = L.circle([data.longitude, data.latitude], {
      radius: circleSizeWind(data.wind_speed),
      fillColor: "#E48080",
      color: "#00AA99",
      fillOpacity: 1,
      stroke: true
    }).bindPopup("<h2>" + data.spot + "<h2><h4>Wave Height: " + data.wave_height +  " M</h4><h4>Air Temperature: " + data.air_temp + " °C</h4><h4>Wind Speed: " + data.wind_speed + " m/s</h4>");;
    windMarkers.push(windMarker)
  }

  // Create a layer group that's made from the markers array, and pass it to the createMap function.
  createMap(L.layerGroup(surfMarkers),L.layerGroup(waveMarkers),L.layerGroup(tempMarkers),L.layerGroup(windMarkers));
}

// Perform an API call to get the information. Call createMarkers when it completes.
d3.json("https://surf-app.onrender.com/api/v1.0/surf").then(data => createMarkers(data));
```

Our Dashboard



Project link

Thank You!

Questions?