Greg Grube
4/16/22
Geography 777: Project 2

Introduction and Objective:

The park that I chose is officially two parks, but run as one park administratively: Sequoia National Park and Kings Canyon National Park (abbrevated within the Park Service as SEKI). Sequoia National Park is the second oldest National Park in the U.S., founded in September of 1890. Kings Canyon National Park was designated in 1940, but traces its roots back to the smaller General Grant National Park which was formed shortly after Sequoia National Park in October of 1890. Kings Canyon National Park absorbed the land of the former General Grant National Park and was expanded in area. Because both parks are adjacent to each other, they are operated together.

One of the main attractions of the two parks is the Giant Sequoia, the most massive type of tree in the world. These trees occur naturally only in groves on the western slopes of the Sierra Nevada mountain range of California and within SEKI there are about 40 different groves of Sequoias. The trees are uniquely adapted to the specific climate of the area, including having bark that is unusually fire resistant and in natural conditions the fire helps to open their cones. While the Giant Sequoia is well adapted to fire conditions, because of changing climate conditions and fire supression methods, they are at increasing great risk of more destructive fires.

As a result, the park has decided to task both its own staff and volunteers to inventory the Giant Sequoias within the known groves in the park. I am tasked with creating an interactive, mobile friendly map that can be used to collect tree data for the Giant Sequoias, documenting their attributes and condition for future analysis. This map should also allow inventory takers to navigate around the park and to specific areas, find points of interest and alert them to services that they may need near the area they are exploring.

For park staff and interns with the park service, they would be gathering data with either laptops or tablets, so the application needs to be set up to work well with those interfaces. For volunteers, they would mostly be collecting data from their mobile device, so the application needs to be able to function under those conditions as well.

Data Collection and Database Design:

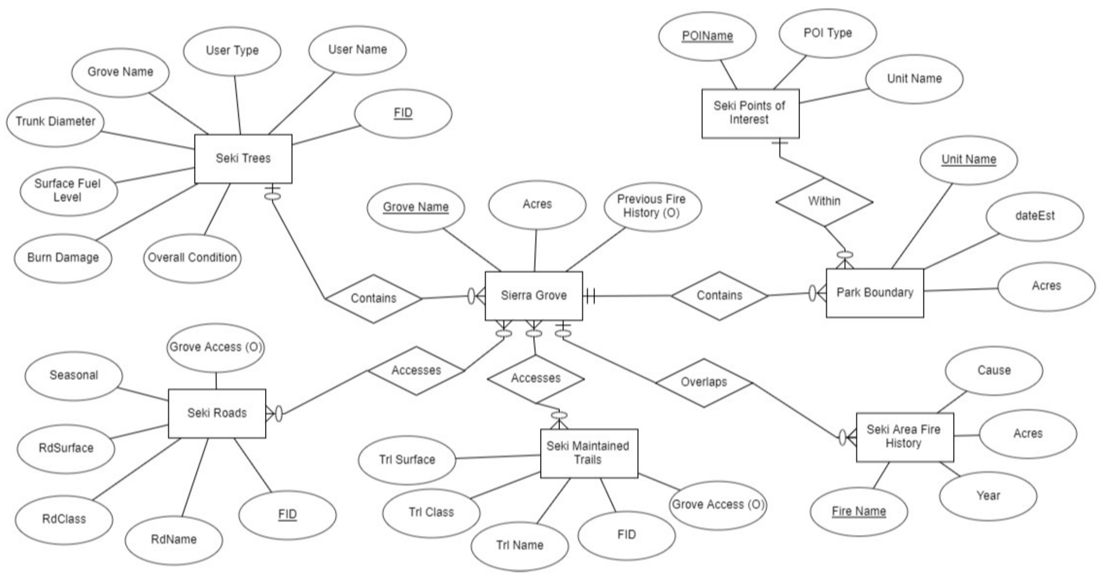
The first step in working on this project was to gather the data that would be used for the project. Fortunately for me, Sequoia and Kings Canyon National Parks had a lot of the reference data already available. I was able to find general reference data like a park boundary layer, trail and road data. There was also a Points of Interest dataset that included different categories of features included. Finally, I was able to find some datasets that could be specifically useful to the individuals gathering the Sequoia data, which were Sequoia grove polygons and a dataset showing fire history in the area.

I was not able to find an existing tree inventory layer, so that would have to be created from scratch and edited by the users of the application. I referenced a helpful document from the U.S. Forest Service, which gave me an idea of the types of items that an inventory-taker might need to record (https://www.fs.fed.us/r5/sequoia/gsnm/feis/AppI.pdf). I tried to reach a balance to not include everything included in that document so that the eventual data entry would not be too cumbersome, but it included items like fire damage, tree diameter and overall condition.

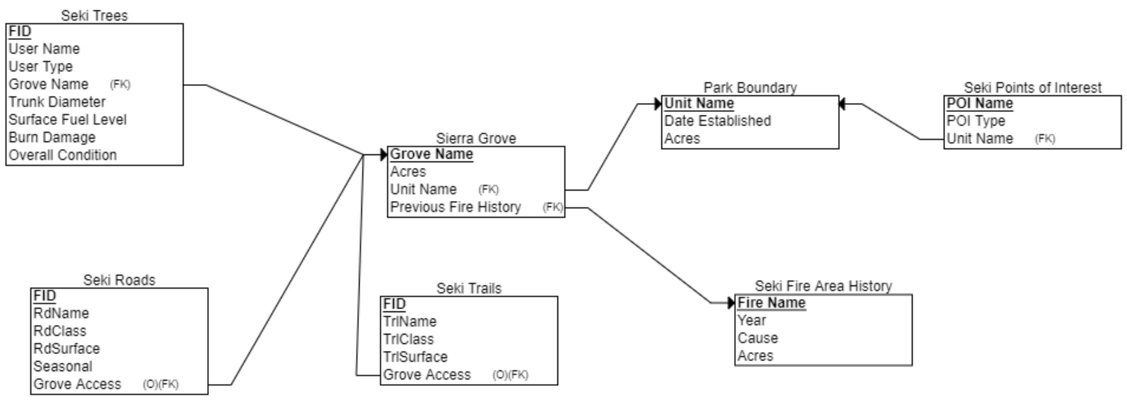
Having the data collected, I then had to prepare it for the database. Specifically, I had to decide which fields to use and if additional fields were needed. Many of the datasets from the National Park Service were pretty comprehensive, but included way more data than was needed for this project. So, I decided which fields were worth maintaining and deleted the remainder. This process was done while creating my E-R Diagram and Logical Schema diagram. That helped to design the database and get it ready for use in the application.

Finally, once all of the data was collected and the tree dataset was set up, I needed to decide where to set up my database so that the data would be available for the application. Initially, I was looking at using Geoserver to host my data and to use Leaflet to display. However, I really was struggling to get the editing functionality to work within Geoserver, specifically when it came to editing an existing layer. I then decided to pivot to using ArcGIS Online to store my data and when coupled with the ArcGIS Javascript API, the editing functionality seemed to work more seemlessly. ArcGIS Online is also fairly intuitive when it comes to uploading data and integrating using Feature Layers, which can be added to the application and styled easily.

E-R Diagram:



Logical Schema:



Functionality:

Once the database was designed and uploaded to ArcGIS Online, I was able to focus on the functionality of the application. Because the application is predicated on collecting tree information, I first wanted to focus on getting the editor tool to work properly. The user should be able to place a pin on the map at the location of the tree, fill in the survey results after analyzing the tree's condition and then display that on the map. So, I needed to tie the tree data layer to the editor tool in the Javascript API and then allow new features to be added on the backend.

ArcGIS Online has options to make a data layer editable and then I called that dataset with the help of the Editor widget. This widget allows for existing data to be edited and for new data to be created. I had to customize some of the settings to work with my data and I also had to design an appropriate logo for the tree data.

By default, the editor used a simple point for all the trees, but I classified the different points by their 'overall condition' field and displayed them in different colors based on that condition using a SVG image of a Sequoia tree. Once the point was placed on the map, I styled a pop-up that would display the various data fields the user entered. I also placed domains on the editor form so that the user can select options instead of free typing, with the exception of the user name. Finally, there is a filter tool that can help to sort through many datasets within the editor, but because this application only has one dataset being edited, I removed the filter option to make things cleaner.

One of tools that I realized I needed early on was the ability to zoom to the location of the user. If the user is going to be adding tree locations, it's very difficult to get that information in the right place only using an air photo and reference points on the map. However, if the user is using a mobile device or tablet to collect that data, we can grab that location and zoom the map to place the point in the correct place. ArcGIS Javascript API has a Locate widget designed for this purpose, which zooms to the location of the user and places a point on the map. I had to hard code a specific location on the map to act as a test location when working on the design, but when used in the field it should zoom to the user's coordinates automatically.

Another tool that was essential for navigation was a search function. Again, the Javascript API has a Search widget that allows the developer to choose the layers that the user can search on. It also contains an autofill function once the user enters 3 characters so that they only have to pick from the list. I configured the tool to search the points of interest, trails and groves layers; in my opinion, those are the most likely features that a user would search for by name.

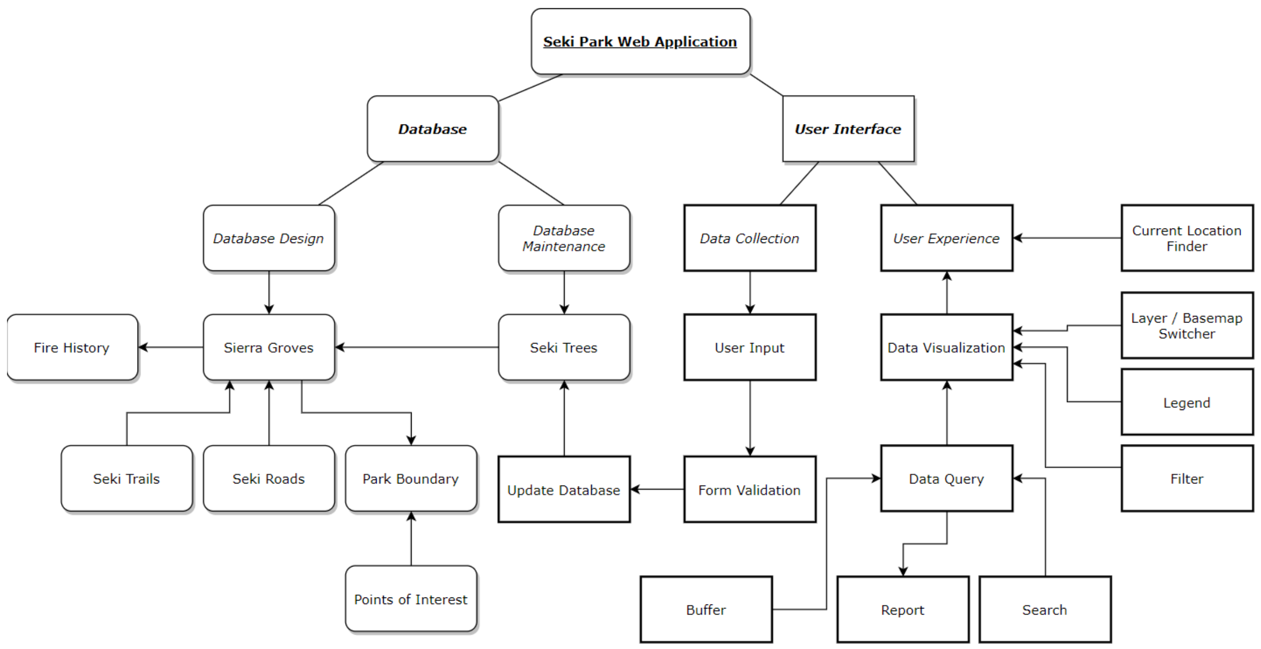
There were also a group of data visualization tools that I added to the map. The Legend widget was another obvious choice to add because it shows the symbology of all of the currently displayed layers in a single list. As was the Layer List widget, which allows for the user to turn on and off the various data layers to suit their needs. By default, the only layer that is turned off is the Fire History, mostly because it is a large polygon dataset that quickly overwhelms the map when combined with the others. Finally, I added a basemap switcher to the application that allows the user to switch from the default topographical basemap to a satellite view if they prefer that view.

Most of the previous widgets required only minor customizing, but there were a few widgets where I had to do much more customization to make them work because the Javascript API didn't have what I was looking for. One was a filter tool designed to work with the Points of Interest layer. Because there are so many different categories within the layer, I wanted the user to be able to select from them for easier visibility depending on what they were trying to find.

Another widget that was customized was a buffer/select tool that allows the user to see features that are within the certain distance of a given location. The user selects a distance and then places a point on the map, which shows a distance from that location and selects and zooms to all of the points of interest within that circle. It also shows a report of all of the points in the corner of the map. This would allow a user to find important resources that are near their location, such as restrooms, telephones or shelter when they are in the field.

The final tool that I put together was a simple informational panel that displays as the map opens. This tells about the purpose of the map and lets the user know how to interact with the rest of the tools. It also includes a logo for both parks.

Implementation Diagram:



<u>Layout/Design:</u>

Once I was able to create all of the needed functions for the map, my next step was to bring all of the elements together and make them work well together. The first issue that I had was getting all of the different tools to display without overwhelming the user. Especially if the user has a mobile device when they are in the field collecting data, they are going to need the ability to make tools larger or smaller depending on what they are trying to accomplish. Luckily, there is a tool called Expand that creates a button that can be clicked to make the window appear when clicked or contracted when clicked again. This allowed me to create a tile for each feature along with a logo and place them in logical locations on the application. I also configured the Expand tool to automatically close when a different tool is picked so that they screen stays uncluttered.

Another design item that I added to the application was a tooltip that informs the user of what each tool does while open. When the application opens, it is set to a default message, but once a widget is opened, the message changes to tell the user about what the tool does and how to interact with it. Once closed the tooltip resets to the default message until the next widget is clicked. This tooltip box can also be closed using the Expand buttons like with the other tools.

Once that was complete, another decision was deciding where to place each of the different widgets. On the upper left, I placed most of the basic navigation tools (zoom in/out, home button, location finder) as well as the tooltip so that it could be open at the same time as the tools on the right. The upper right contains most of the application functions as well as the informational panel. Only one of these functions can be open at once. I ordered them in the order that the user would likely use them, though I also considered the size of their data because some of the displays were pretty large and I didn't want them to fall off the page. Finally, I placed the basemap switcher in the lower left because it is larger and helped to balance out the overall look of the application.

After the overall layout of the main application was complete, I took some additional time to consider the mobile layout of the application. As previously discussed, having the expand buttons allowed the user to have greater control over how many items were open. I also made sure that the different elements displayed okay when using most of the major devices. By default, the Javascript API opens widgets in a panel for the mobile version, but I turned that off because some of the tools only work while open, making it nearly impossible to use under that configuration.

I also took some time to configure the pop-ups for the different data layers. Some items, like the tree layer, I wanted to display all of the available fields when that element was clicked on. Others like the trails and roads, there was really only the name and surface type that was needed. For the larger polygon layers like the groves and fire history, I used labeling instead of pop-ups because otherwise they always would get selected when a user was trying to select the lines or points within them. I also customized the pop-up to have the ability to show in a static, "docked" position and placed that in the lower right to be out of the way of other functionality.

The final design element that I considered was the symbol choices for the different datasets. As mentioned earlier, the tree symbols were broken up into the different categories by overall condition. The 'Excellent' category has a nice healthy green color the subsequent lower colors reflect less healthy orange, red and in the case of fallen trees, black. I used a white sequoia tree outline as the picture displayed so that it would stick out compared to the colored background and decided on a circular symbol for icons.

The Points of Interest symbols also have differences based on their type, derived from the options made available from the National Park Service. Because there are 10 different types, I also used colors to differentiate them from each other. The colors I used were to try and have as much contrast from each other as possible. I also made the symbols colored with a white background to contrast them from the tree icons and used a square background to make a further distinction between the tree and point of interest symbol types.

I then styled some of the more basic layers. The park boundary was a subtle green background, while the groves are a more yellowish-green so that they stick out from the background. The fire history is categorized into colors for cause: human, natural and other. The trails are a brown color and the roads are a dark black to make them stick out from each other.

Finally, I assessed the overall scheme and layout. Because it is a National Park-themed application, I chose to have the background color of the buttons and views to be black with white writing to mimic the look of the National Park Service maps. I also made sure that none of the colors of the layers clashed too much and that all of the tool symbols were common sense for the potential users. The basemaps also worked well with the overlaid layers.