Andrew Pittman
Spatial Analysis Interactive Application
UW GIS WMP Master's Capstone Project 2
03/30/2020

Glacier National Park is a national park located in Montana, and is home to many recreational opportunities to those willing to travel there. It has incredible views, an extensive network of trails, and support in various forms from the United States National Park Service. While the park has so much to offer, information about the park is largely isolated to paper products within the park and via resources that require internet access. Should an application be made available to visitors that isolated requisite information relevant to the park into a portable format that did not require constant internet access or took the form of fragile paper products, this would theoretically assist visitors in engaging more deeply with park elements.

This is the motivation behind the development of Glacier Social- a mobile application designed to render information about the park in a way that is convenient and engaging for visitors. The application's purpose is to ultimately provide another medium via which visitors can enjoy Glacier National Park, and to streamline interaction with existing park elements.

The application was built utilizing React Native, which is a mobile application development framework that streamlines development to both Apple iOS and Google Android devices, minimizing operating system specific coding to maximize development of features. In addition, ExpressJS was used to set up an API locally to access the backend database. Mongo DB Atlas was chosen to facilitate backend transactions and requests. Atlas is a freely hosted instance of Mongo DB available for public use up to certain size limits, which are well below the requirements for this project.

In designing the features of this application, it was necessary to balance the above stated goals with a handful of important restrictions inherent in designing a technological application for use in a low data service area (a vast national park with few cell towers). In the eyes of a user, it was important to facilitate app functionality despite a highly likely potential lack of data coverage to a device. The application needed to load and function regardless of data availability, and still provide essential feature as needed by the user.

In order to accommodate this, almost all data needed by the application is served to the app upon initial login, which is assumed to be when a user is most likely to be in range of a cell tower or wifi network. This could be on the way into the park, or at a visitor center, or at home prior to leaving for the park. Additionally, once the user logs in to the application, the application stores a token verifying the user's account so that repeated logins are not necessary as long as the app is not stopped manually by the user. Future considerations and features that could accommodate the low data environment problem/solution set could include: pre-caching all required map tiles need within the park at all zoom levels, storing requests for transmission once a network connection has been detected, and further minimization of dependencies on data networks.

Thinking about how each user might enjoy the park, and engage with it in a more meaningful and social way, the concept of location sharing in a variety of formats was considered. To maximize individual unique enjoyment of the park, capturing a fully recorded track was decided upon- not just an isolated location. This would enable greater detail to be shared between users, and enable greater visualization capabilities and future potential within the application.

This feature came to life as the “Blaze a Trail” feature located on the main map screen within the application. This functionality allows the user to record a trail at a start and stop time of their choosing, as well as saving that trail for later review on their personal account as well as immediate sharing with the broader Glacier community as a whole via a map layer on the main map screen. Functionality is simple and intuitive, with minimal inputs required for operation, as well as prominent placement on the map interface.

When interacting with map applications, especially ones designed to orient you to a variety of opportunities, users have come to expect layer toggling as a feature. This is accomplished on the account screen via a series of check boxes. This feature allows users to tailor their view of the main map screen to their current needs, whether they be hiking for a nearby peak, parking near a lake, or simply driving to the amazing hike they did last summer. The layers currently available in the app include: a default park boundary, park trails, user generated trails, roads, points of interest, parking lots, and nearby mountain peaks. Users have also come to expect tooltips with information about map data as well, and this is provided for each of the layers within the main map interface.

To facilitate user traversal of the map, various quick toggle camera features were added to streamline common requests in the app. A zoom to home button was added to quickly orient the map and the user to an overhead view of the full extent of the park boundaries at a high level. Also, a zoom to user location was added to quickly show the user their own location within the map’s interface. Finally, a zoom and lock on user location was added- which does exactly as is described and works well in tandem with recording a trail as the user is likely to be looking around at the park and not at the application while recording a trail. This feature will always ensure the user’s location is front and center in the map so that when they do eventually look back down to orient themselves or see nearby interesting park elements, they won’t have to find themselves again and again.

In an attempt to provide some privacy and ownership to the application, user accounts and authentication were folded in as well. This allows tracks shown to the user to be correctly correlated when each user utilizes the application and wants to review previously recorded trails. As previously mentioned, a token is logged on the phone so that repeated logins are unnecessary as long as the user doesn’t manually kill the application. Manual sign out is available on the account page, as well.

The schema chosen for the backend MongoDB Atlas database is fairly simple and straight forward, as Mongo has a set schema for its geospatial objects. The entity relationship diagram (Figure 1) and logical schema (Figure 2) are shown to the right and below. Due to constraints with the logical schema drawing program, sub elements in the Mongo db documents (in JSON format) are shown a separate tables, when in

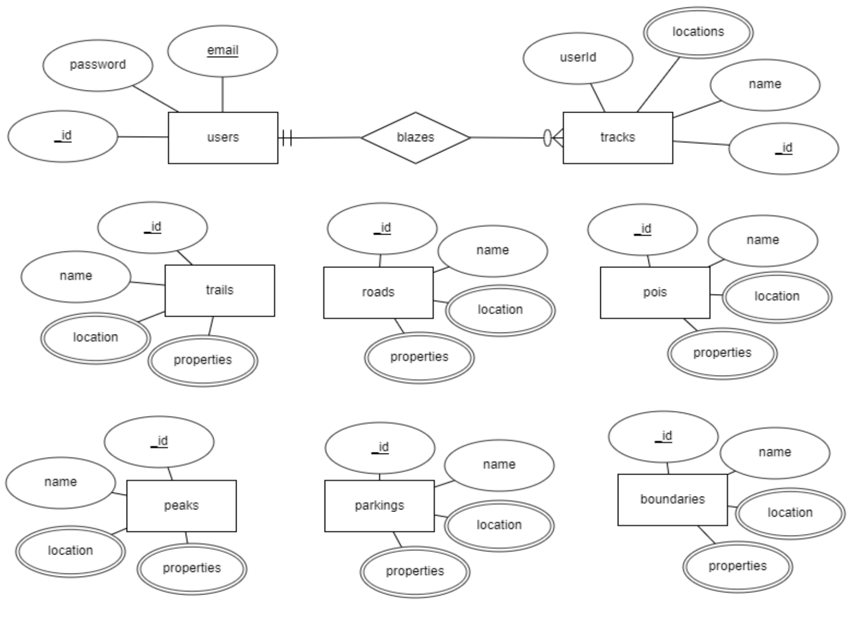

*Figure 1: Entity Relationship Diagram for Glacier Social Mobile Application*

reality they are the same table but sub objects or sub arrays of one another. An example of this is roads, roads_location and roads_properties: they are all a part of a singular document schema but broken apart on the diagram for clarity.

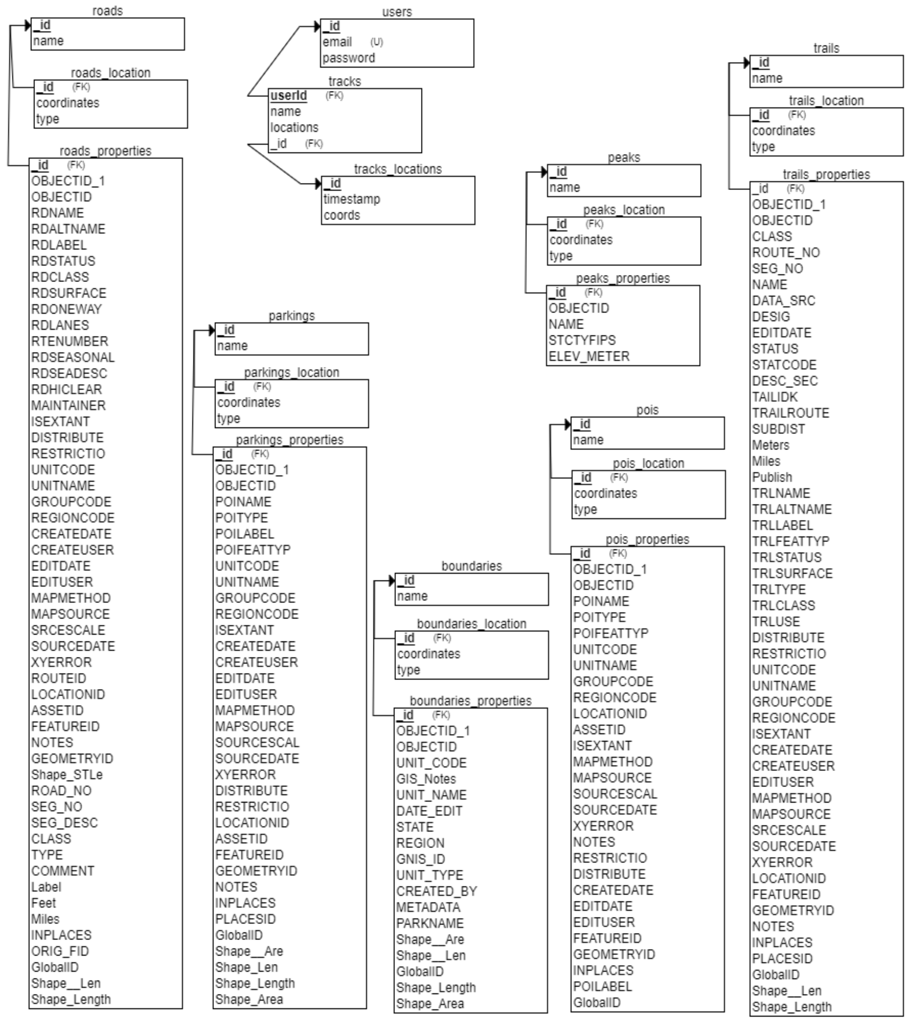

*Figure 2: Logical Schema for Glacier Social Mobile Application*

The user data portion of the backend database is actually a very small part of the data handling piece. This portion is represented by the users table and its relationship to the tracks table (the database terminology for the user saved trails). In large part, the database design and structure is focused correctly storing and serving the geospatial data gained from the National Park Service. This data needed to set up with correct geospatial schemas defined by Mongo DB, and once the data had been ingested, verifications of its completeness and correctness were conducted. Then geospatial indexing was layered on top of these tables should any complex querying of the geospatial data be desired in the future.

Features included in the Glacier Social app include: (1) custom trail recording, creation and sharing, (2) location and park orientation buttons, (3) layer element callouts to extract specific information, and (4) layer toggling. Additional features that could be added include layer element searching, filtering of layer elements based on a user's geographic proximity, further robust design implementation enabling low data rich environment usage, and more.

The application's code has been open sourced here (https://github.com/dro0o/777p2), and a video overview of the application in use is here (https://www.youtube.com/watch?v=JSRohgUvyhY ).