**Algorithms and Data Structures**

We use many Algorithms and Data Structures but the most prominent ones will be described here.

**9.1 Algorithms**

* Array Lists or Lists
* These lists store our reservations in multiple different array lists that are organized by ‘Status.’ Each array is very important for admin view so they can easily manage this data such as: viewing, approving, and denying these reservations. After these reservations have been approved, they will be stored and sorted accordingly based on the users own account.
* Sets
* We have many users, reservations, facilities, and equipment that needs to be managed. A set is a collection of items which are unique i.e no element can be repeated. Sets are ordered: elements of the set can be iterated in the insertion order. Set can store any types of values whether primitive or objects, which is very useful for our use cases. We are able to return iterator objects, which contain arrays have entries in a specified set. For example, making a reservation requires you to select a facility, date and time, equipment or equipment bundles (if necessary), and give information about your event and sets allow us to make good use of the already used hash maps, iteration functions, and arrays.
* Iterations
* These iterations loop through all of our reservation data and organize them into our lists so they can be displayed to the user or admins. They are mostly used to iterate over our hashes to organize this data.

**9.2 Data Structures**

* Hash Maps
* Our most used and customized data structure which inherit from Object(s) and keeps all of our data in key value pairs and has pretty good performance overall.

Objects are similar to Maps in that both let you set keys to values, retrieve those values, delete keys, and detect whether something is stored at a key. Because of this (and because there were no built-in alternatives); however, these are the most important decisions we had made for using a Map:

* The keys of an Object are Strings and Symbols, whereas they can be any value for a Map, including functions, objects, and any primitive types.
* The keys in Map are ordered while keys added to object are not. Thus, when iterating over it, a Map object returns keys in order of insertion.
* You can get the size of a Map easily with the size property, while the number of properties in an Object must be determined manually.
* A Map is iterable and can thus be directly iterated, whereas iterating over an Object requires obtaining its keys in some fashion and iterating over them.
* An Object has a prototype, so there are default keys in the map that could collide with your keys if you're not careful.
* A Map may perform better in scenarios involving frequent addition and removal of key pairs.