Kyle Anderson
SWDV-691: Software Development Capstone
PoolPal Design: Database

My application will make use of MongoDB, a document-based NoSQL database. My application will be built using the Ionic framework, and thus will be able to utilize Mongoose (https://mongoosejs.com/) to provide object modeling of the collection data stored in MongoDB. Because my application is able to store a majority of its information in a single collection (by using embedded documents), it is more efficient to use a document-based database because a single query can be performed to retrieve all of the data needed for a given operation, rather than needing to perform multiple query operations across tables as would be needed in a traditional relational database.

Below are the collections and fields that will be used in my database:

```
{
   "_id": UUID,
   "user_name": String,
   "email": String,
   "password": String (hashed/salted),
   "name": String,
   "pool_gallons": Number,
   "pool_type": String,
   "chemicals": [
       // embedded chemicals document
       {
           "_id": UUID,
           "chlorine": String,
           "ph_up": String,
           "ph_down": String,
           "alkalinity_up": String,
           "alkalinity_down": String,
           "calcium_up": String,
           "calcium_down": String
       }
   ],
   "readings": [
       // embedded readings document(s)
       {
           "_id": UUID,
           "reading_date": Date,
           "ph": Number,
           "free_chlorine": Number,
           "combined_chlorine": Number,
           "alkalinity": Number,
           "cyanuric_acid": Number
       }
   ]
}
```

This collection will store all of the user-specific information used by the application. Starting at the top, information specific to the user's account will be stored. This data is mutable, but will only ever have a single instance. The user-specific information is used by the application for user authentication. The "pool_gallons" and "pool_type" fields are also used by the application to make recommendations on the amount of chemicals needed to treat the pool water.

Moving further down the collection, information about the user's chemical inventory (i.e. which chemicals the user prefers to use in their pool) will be stored in an embedded document. This information is used by the application when making recommendations to the user as to which chemicals are needed to treat their pool. Again, the information contained here is mutable, but will only ever have a single instance.

Finally, chemical reading data will be stored in the embedded "readings" document(s). This data is meant to represent the chemical levels recorded by the user when testing their pool water. This is stored as an embedded document within the user document and can (and likely will) have multiple occurrences for each user. This information is used by the application to make recommendations of how to treat the pool water. It is also stored and presented in graph form for the user to view their water chemistry readings over time.

A second collection will be used by the application to provide ratios/quantities of each chemical needed to affect a given water characteristic by a given amount. For example:

```
{
    "chlorine": [
        "chemical": [
            "amount": Number,
            "units": String
        ],
        "adjustment": [
            "amount": Number,
            "units": String
        ],
        "water_volume": Number
    ]
}
```

As mentioned, this collection will store information about each pool chemical supported by the application. Looking at the collection, it will store 3 basic pieces of information about each chemical - the “chemical” array will be the amount of chemical required to adjust the “water_volume” a given amount (specified by the “adjustment” array). To better illustrate this, below is an example collection:

```
{
    "chlorine": [
        "chemical": [
            "amount": 3.4,
            "units": oz
        ],
        "adjustment": [
            "amount": 1,
            "units": ppm
        ],
        "water_volume": 1000
    ]
}
```

Here, the information stored in this collection tells the application that 3.4oz of chlorine is required to increase the chlorine level in 1000 gallons of water by 1ppm. When the user enters their chlorine information into the application (which will then be stored in the embedded “readings” document within the “user” collection), the application will find the difference between the user’s chlorine reading and the ideal chlorine reading. It will then reference the information in the “chemicals” collection (shown above) to determine how much chlorine is needed to adjust the chlorine level in the user’s pool to the target level.