I have decided to use Mongodb for my database with mongoose to establish the schemas. I am choosing Mongodb because it is scalable, has high performance and flexible. Mongodb has become of the most preferred NoSQL databases and can handle large amounts of data very quickly in an agile manner. The application code within the database is mapped to the objects by using a document model which makes handling the database very easy. Also, an important part is that the fields can be made to vary within documents to enable data structure changes over time.

I do not have an established website yet so I need a database that can expand and allow me to easily change and adapt and be able to scale easily. Once I am established and I know for sure what criteria I will be using for all aspects of the website, at that time I could look at possibly transitioning to SQL. However, Mongodb works really good with apps and allows me the freedom to easily adjust my database. Upgrades or changes to my application database can be turned around and implemented much quicker because of the flexibility within Mongodb.

Along with Mongodb I will be using Mongoose. Mongoose is an object data mapper or object document mapper (ODM) which will help communicate between NodeJS and Mongo. Mongoose provides ways for us to model out our application data and define a schema. It offers easy ways to validate data and build complex queries from the comfort of JavaScript. I also plan on using Express as a framework to build my web application on top of Node.js. This simplifies the server creation process that is already available in Node.

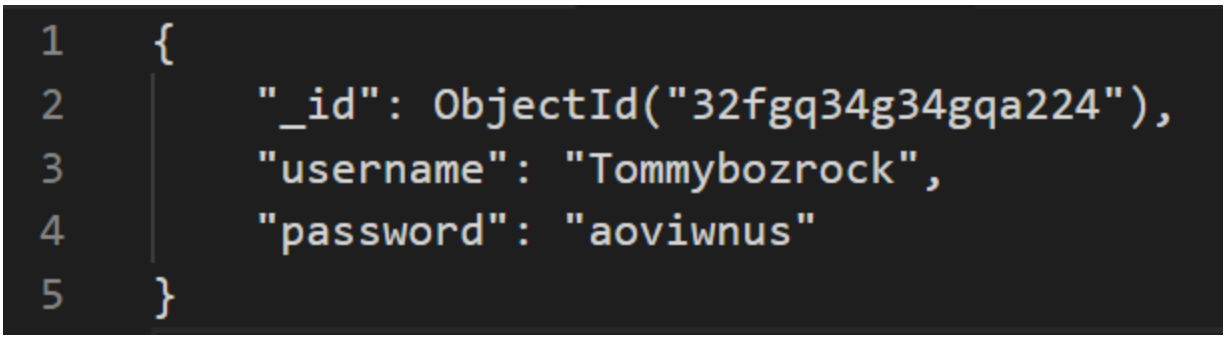
To upload pictures, I will be using Cloudinary to store images for me that allow me to send files (the data) to Cloudinary. Cloudinary will send back URL's and I can use those to send to Mongoose. I am still in the process of learning how all of this works but that is something I am in the process of researching.

Below is how I was thinking about setting up my database. I wasn't sure if I should just create a data structure of destinations and then do separate categories for places gone, places projected to go within the next 5 years, and places to go before I kick the bucket. The reason I separated them out into their own is because I wanted each section to have their own page to click on so I thought it might be beneficial to have them all separated out.

The User Account Information would be used to associate the user with their username and password.

User account information

Keyname	Data Type	Usage
_id	ObjectID	Primary document key
username	String	Login username
password	String	Login password



The following use the same criteria for the next three tables but one is for places gone, one is for places projected to go within the next 5 years, and the last one is for places I would like to go before I pass away (kick the bucket).

Destination information for places Gone

Keyname	Data Type	Usage
userId	User_ObjectID	To match up with user
destinationId	Integer	To assign each destination their own ID
place	String	To show where the destination is
date	Date	To show when the travel happened
cost	Number	To keep track of how much each trip cost
PictureId	Number	To give the picture their own ID
pictureLocation	String	To show where the picture is being retrieved from
Comment box (Stretch feature)	String	A comment box to write notes about why a user gave a 1-5 star rating

```
{
    "userId": User_ObjectId("32fgq34g34gqa224"),
    "destinationID": "25246115",
    "Place": "Palm Springs",
    "date": "December 21,2014",
    "cost": "2436",
    "PictureId": "21",
    "pictureLocation": "cloudinary URL"
}
```

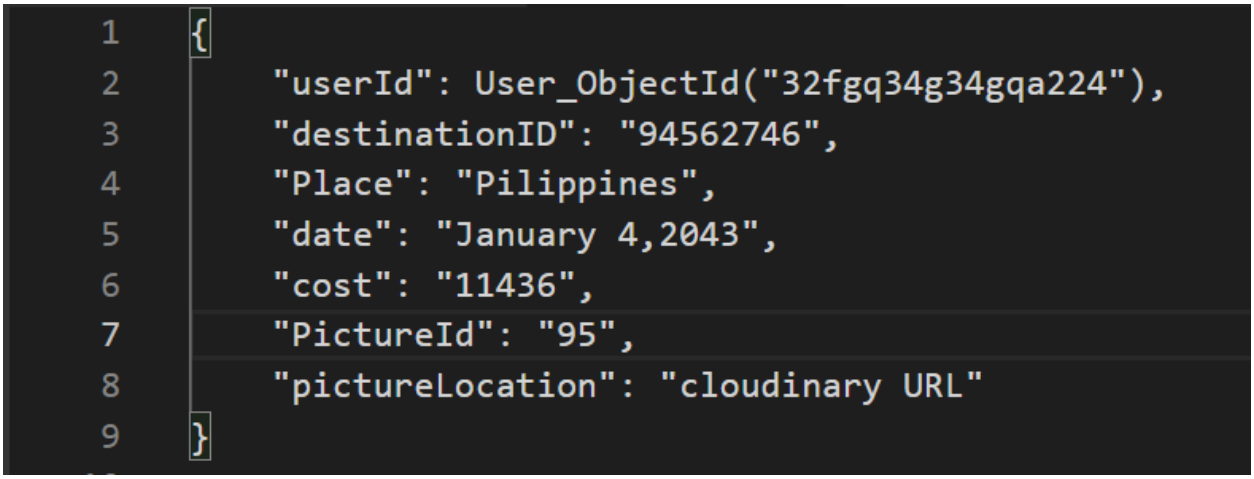
Destination information for places projected to go within 5 years

Keyname	Data Type	Usage
userId	User_ObjectID	To match up with user
destinationId	Integer	To assign each destination their own ID
place	String	To show where the destination is
date	Date	To show when the projected travel date
cost	Number	To keep track of how projected cost will be
PictureId	Number	To give the picture their own ID
pictureLocation	String	To show where the picture is being retrieved from

```
{
    "userId": User_ObjectId("32fgq34g34gqa224"),
    "destinationID": "52398146",
    "Place": "Hawaii",
    "date": "May 11,2023",
    "cost": "13436",
    "PictureId": "42",
    "pictureLocation": "cloudinary URL"
}
```

Destination information for places projected to go before kicking the bucket

Keyname	Data Type	Usage
userId	User_ObjectID	To match up with user
destinationId	Integer	To assign each destination their own ID
place	String	To show where the destination is
date	Date	To show when the projected travel date
cost	Number	To keep track of how projected cost will be
PictureId	Number	To give the picture their own ID
pictureLocation	String	To show where the picture is being retrieved from



In my peer review I was asked about how I would store my images as well as how I would create my user ID's. As already stated above I plan on using Cloudinary to allow me to store pictures. I also plan on letting MongoDB to create my user ID's. For my stretch feature I will add a star rating with comment section on my places gone section. This would allow users to see how satisfied other users were with destinations gone. This will help in determining if a user wants to go to a certain location.