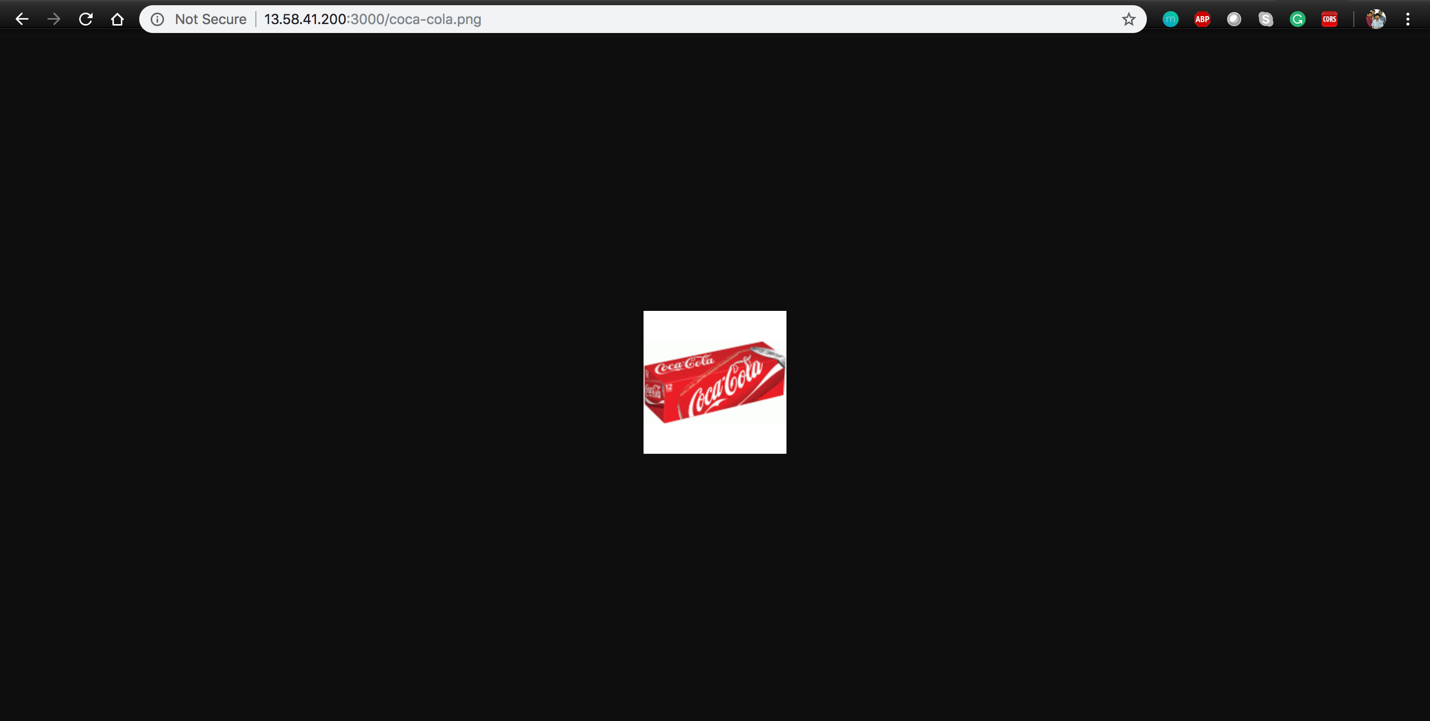
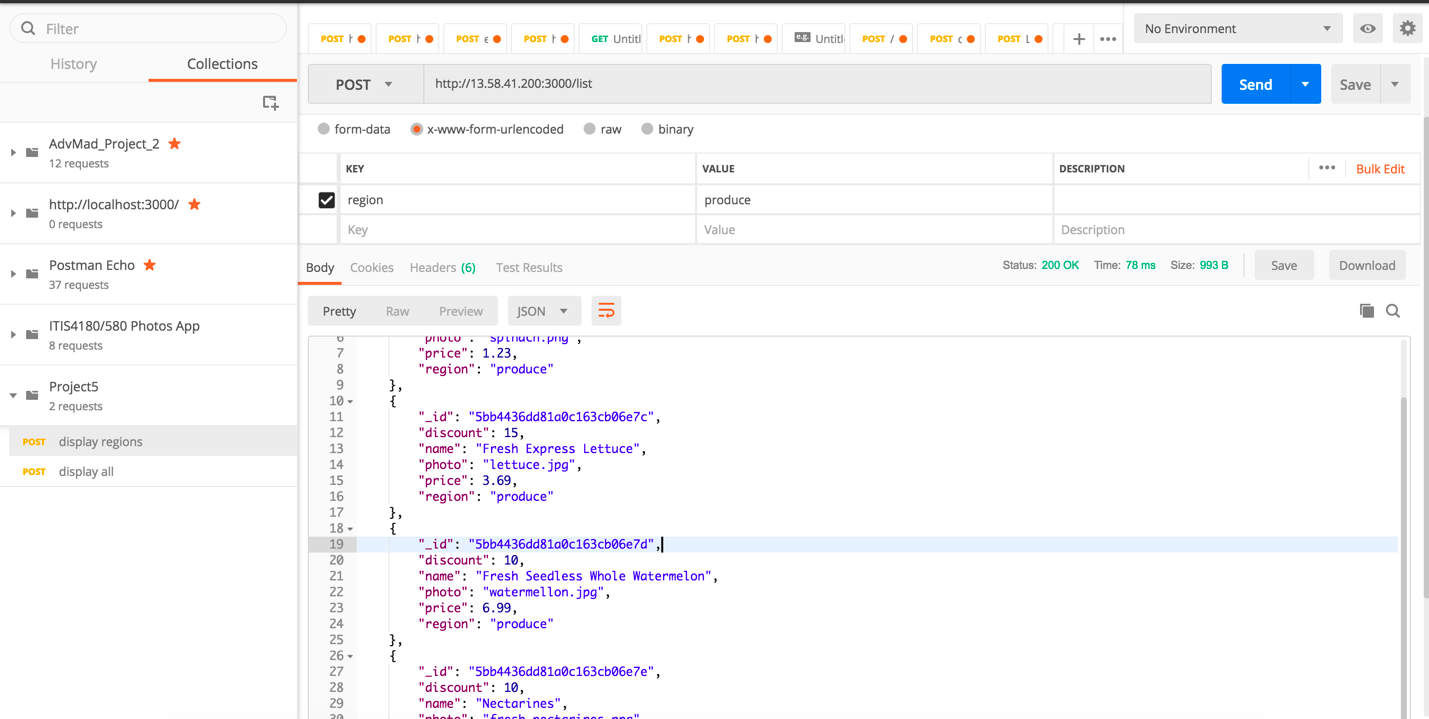
API Design:

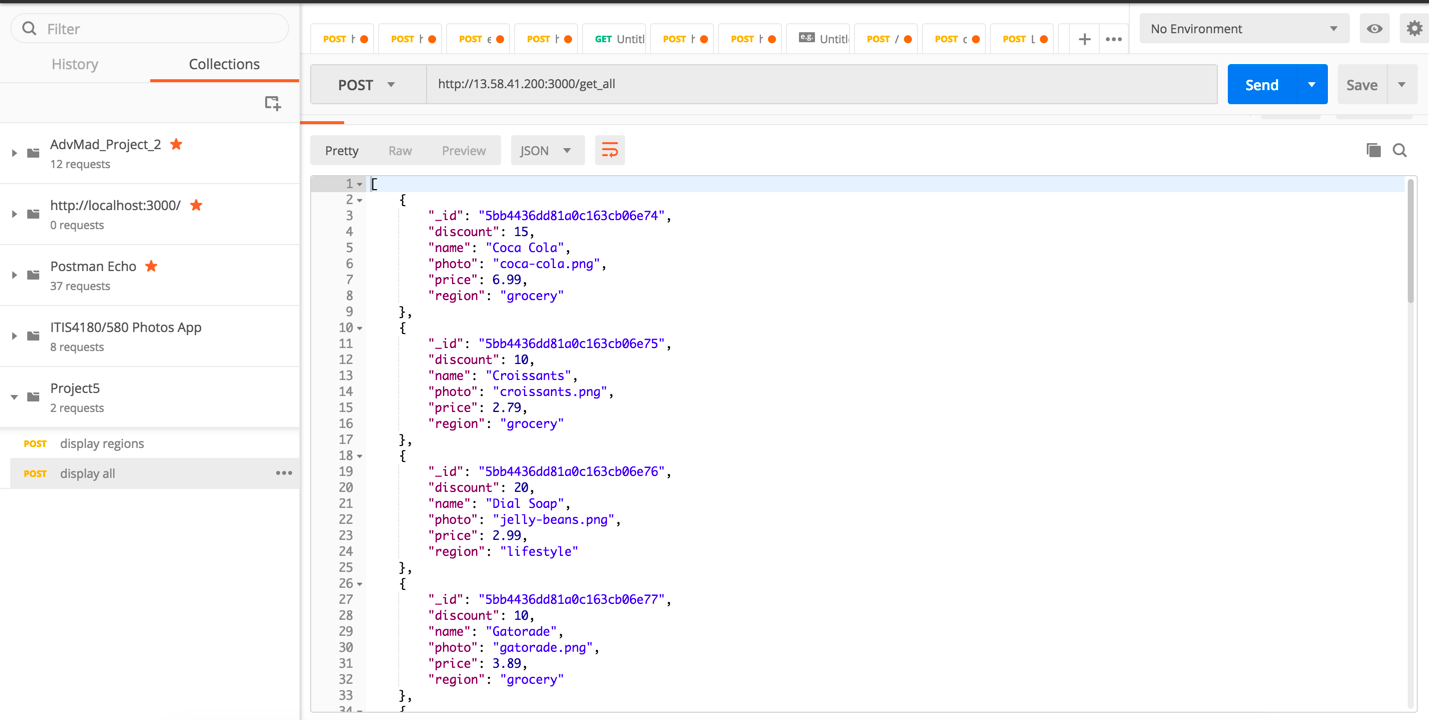
We have created two API routes (NODE JS, Express, Mongoose – for DB connection)  
The API connections are:

1. list: This route taken in a keyword region in the body, which states the kind of list to be displayed. This queries the entire database, stores on a MongoDB Atlas and returns the filtered JSON
2. get\_all: This just selects all the entries in the DB and returns the JSON object array; used for displaying when none of the beacons are encountered.

For displaying the images via server, a publicly hosted page (images) has been pmade, which contains all the images in the images folder.

Public IP for the application on Amazon EC2 instance - **13.58.41.200:3000**

Please refer to the screenshots below:



DB Schema- (How Data is stored on MongoDB Atlas)

We have used MongoDB Atlas for importing the given ‘discounts.json’ file and have stored into a collection named “results”.

Each document has fields –

1) Name- Name of the product.

2) Photo - Filename of the image of the product.

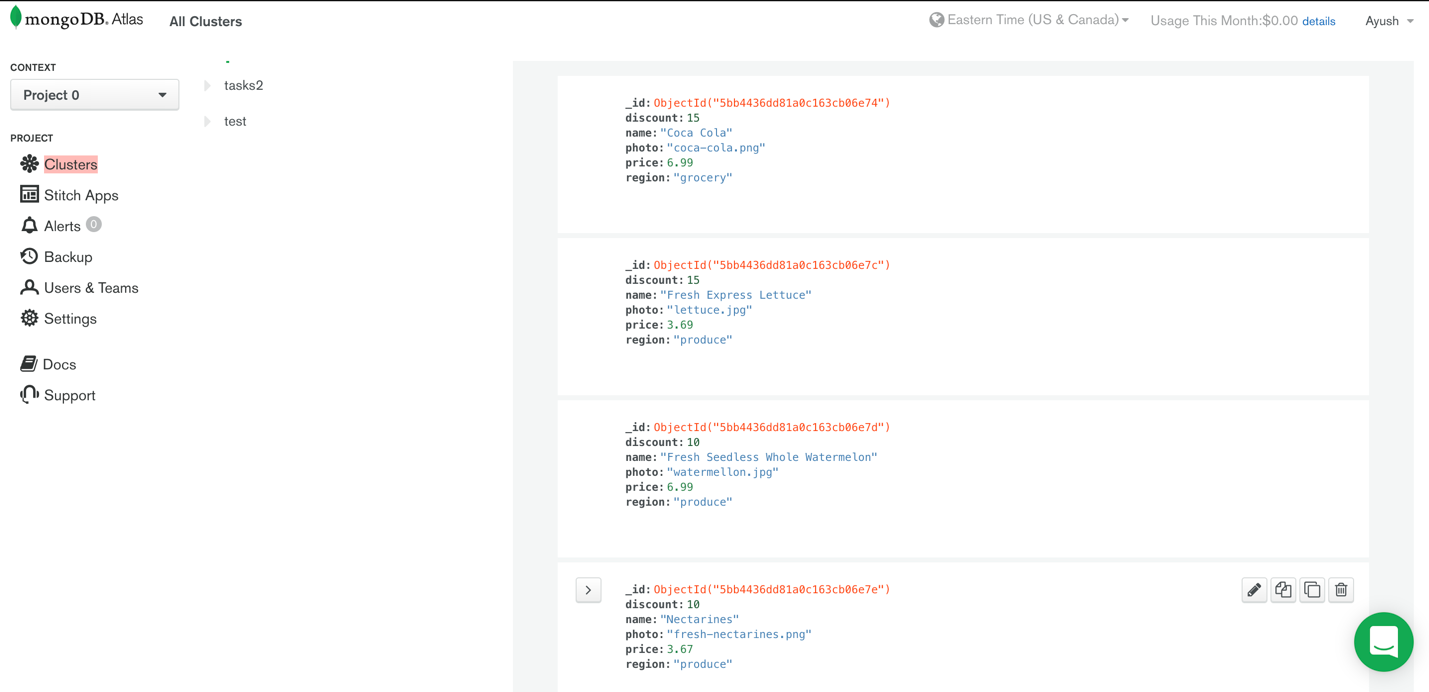
3) Price – Price of the product.

4) Discount – Discount in the product.

5) Region – Category of the product.

Commad used:

mongoimport --host Cluster0-shard-0/cluster0-shard-00-00-mnw6b.mongodb.net:27017,cluster0-shard-00-01-mnw6b.mongodb.net:27017,cluster0-shard-00-02-mnw6b.mongodb.net:27017 --ssl --username amad123 --password <PASSWORD> --authenticationDatabase admin --db <DATABASE> --collection <COLLECTION> --type <FILETYPE> --file <FILENAME>



Mobile Application Design:

1. For Proximity:

* We have used haspmap set for storing major and minor along with the region name, so we have a created a arraylist of maps containing the regions in a minimum distance to determine which beacons are likely closer, and which are probably further away from the device, so that it fixes fluctuating problem.

Screenshots of the Mobile Application: