CANINE DATABASE - PARIS

Implementation with MongoDB

Krishna Teja Kancherla
Abhigna Doddiganahalli chandrashekar

Objective

In this project we have developed a web application for use in veterinary clinics, to maintain records of all appointments.

Technology stack

We decided to use MEN stack for our application.

- MongoDB : NoSQL database
- NodeJS : Server side platform , we have chosen this mainly because of its rich feature set and also because our application is not CPU intensive.
- ExpressJS : web application framework for Node.js

Apart from this we have package dependencies such as

- express-handlebars: It is a template engine and used to create client-side applications.
- mongoose: Helps to communicate with MongoDB.
- body-parser: Helps to convert the POST data into the request body.
- nodemon: Helps to automatically restart the server whenever the code changes.

Getting Started

1. Install Node.js (V12.13.1) from https://nodejs.org/en/download/
2. Install Packages

   npm i --s express express-handlebars mongoose body-parser

```
C:\Users\Krishna\Pictures\dogs-mongo-app - final>npm i --s express express-handlebars mongoose body-parser
+ express-handlebars@3.1.0
+ body-parser@1.19.0
+ express@4.17.1
+ mongoose@5.8.1
updated 4 packages and audited 217 packages in 26.756s
found 0 vulnerabilities
```

3. Install nodemon such that it can access any file in the directory, install it with the global command:

   npm i -g nodemon

```
C:\Users\Krishna\Pictures\dogs-mongo-app - final>npm i -g nodemon
C:\Users\Krishna\AppData\Roaming\npm\nodemon -> C:\Users\Krishna\AppData\Roaming\npm\node_modules\nodemon\bin\nodemon.js

> nodemon@2.0.2 postinstall C:\Users\Krishna\AppData\Roaming\npm\node_modules\nodemon
> node bin/postinstall || exit 0

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\nodemon\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.2: wanted {"os":"darwin","arch":"any"} (cur
rent: {"os":"win32","arch":"x64"})

+ nodemon@2.0.2
updated 1 package in 17.563s
```

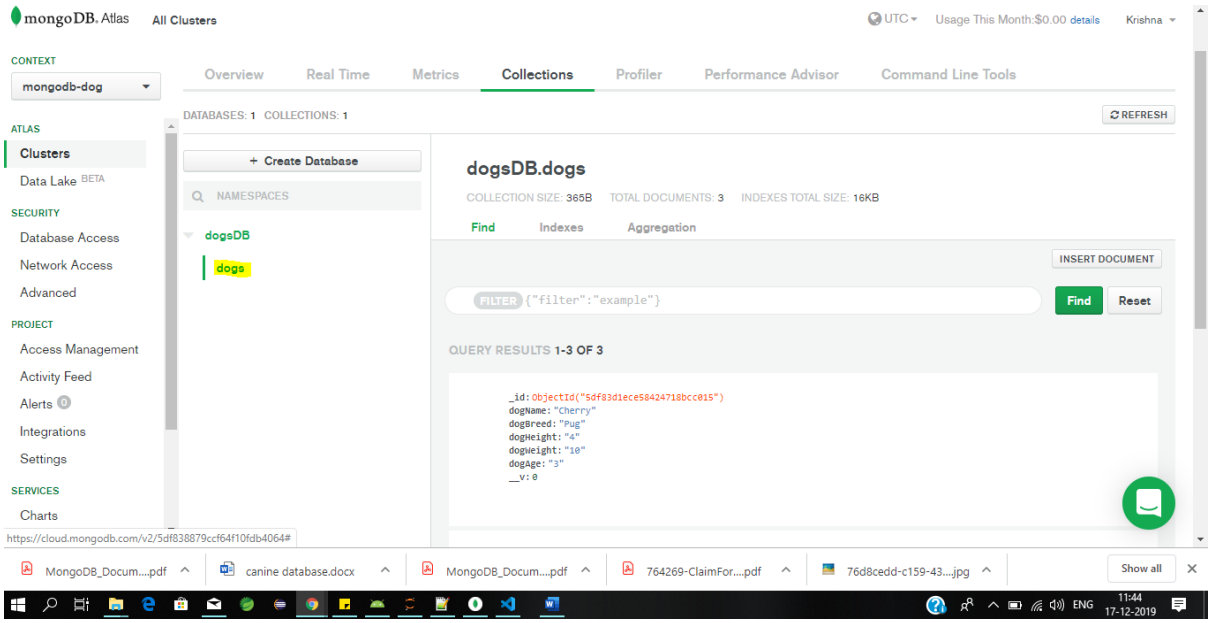
4. npm i -g nodemon
5. Open the Command prompt and navigate to the project folder or if you are using Visual studio code open the terminal and type in the below command to start the server.
6. Nodemon script.js

```
C:\Users\Krishna\Pictures\dogs-mongo-app - final>nodemon script.js
[nodemon] 2.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node script.js`
(node:16216) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future
version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient cons
tructor.
Listening on port 3001..
Connection with MongoDB was success.
```

You can now launch your application in any browser at **http://localhost:3001**

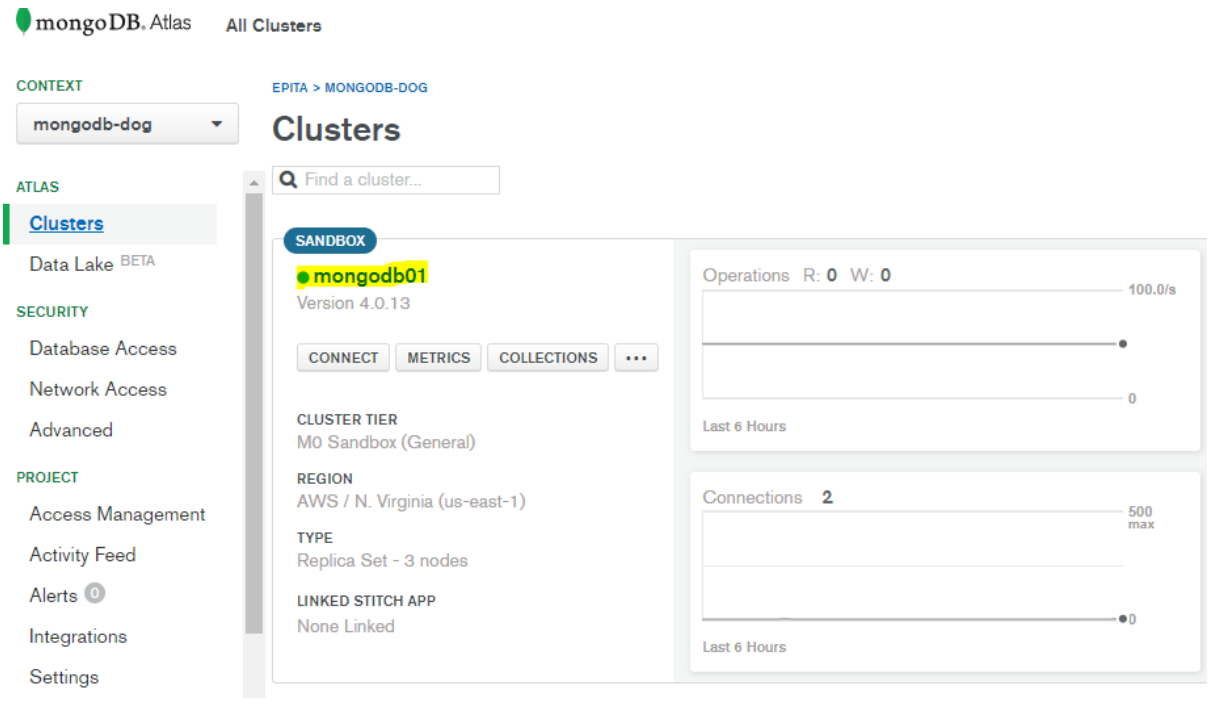
Database collections:

We have implemented one collection : DogsDB as our application as of now does not have complex data

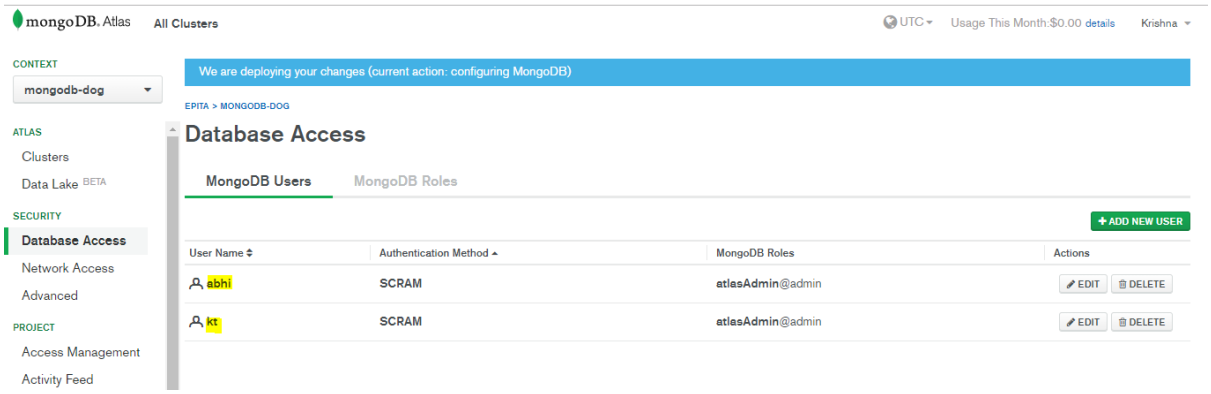


Cluster Implementation:

Created below Cluster in cloud to connect our application so that the database will be handled at cloud MongoDB atlas (not Local)



Created Database Access for below users



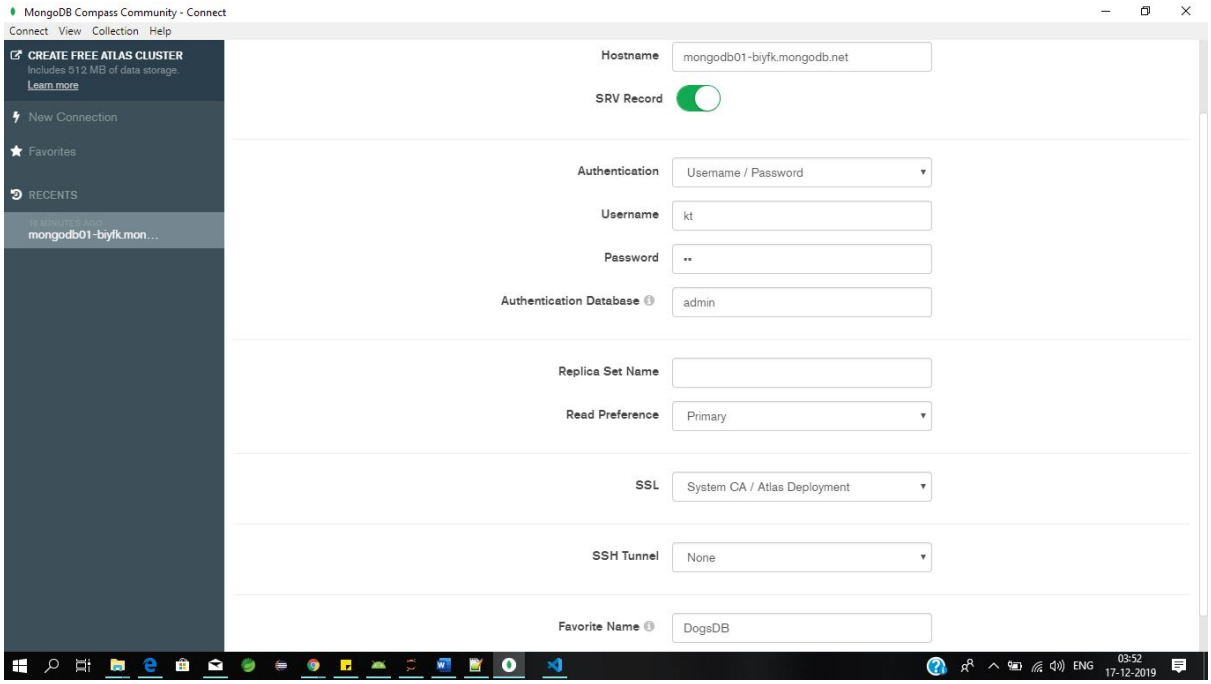
Procedure to connect Cluster to Mongo DB Compass

To connect we use following details and refer below image for more details

Hostname : mongodb01-biyfk.mongodb.net

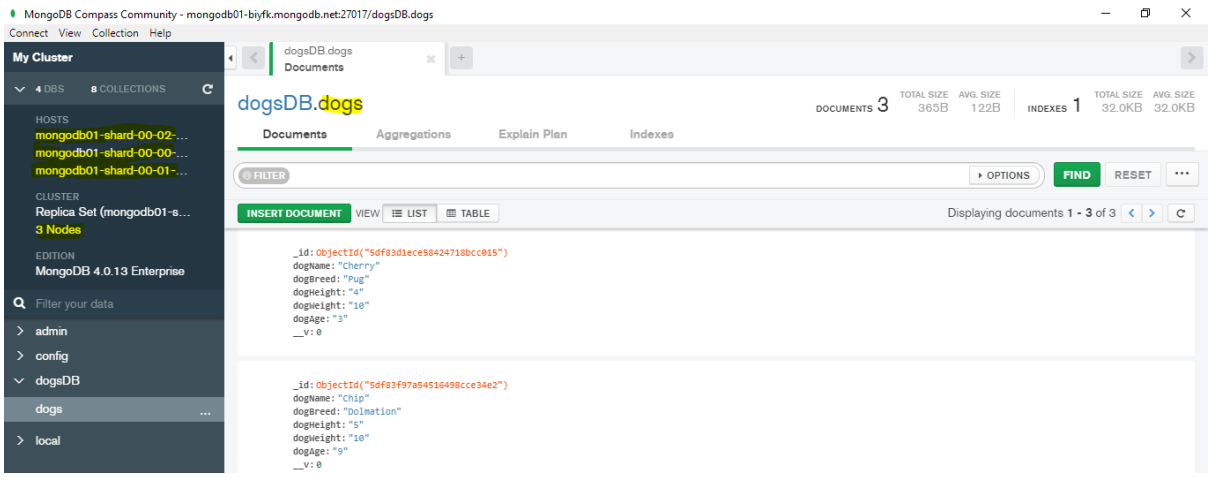
Username : kt or abhi

Password : kt or abhi



Now MongoDB compass is connected to cluster(**mongodb01**)

3 Nodes with replica set after connecting to Mongo DB



Code snippet for connecting cluster in our application

```
 * @author ${Abhigna DC, Krishna Teja}
 * This file is created for MongoDB connection
 */
const mongoose = require('mongoose');

//Cluster Connection details
mongoose.connect('mongodb+srv://kt:kt@mongodb01-biyfk.mongodb.net/dogsDB', {useNewUrlParser: true},
if (!err) {
console.log('Connection with MongoDB was success.')
}
else {
console.log('Failed to Establish Connection with MongoDB with Error: '+ err)
}
});
```

Document structure:

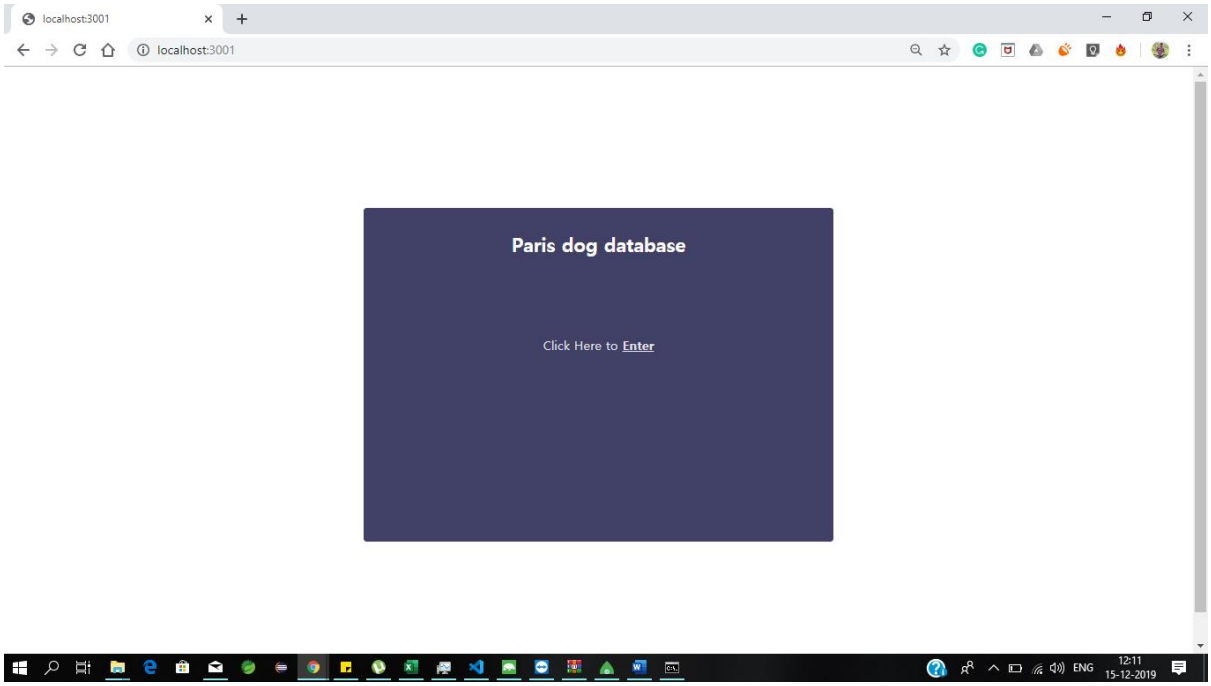
We have 5 fields

- Name
- Height
- Weight
- Breed
- Age

Application structure

Home page

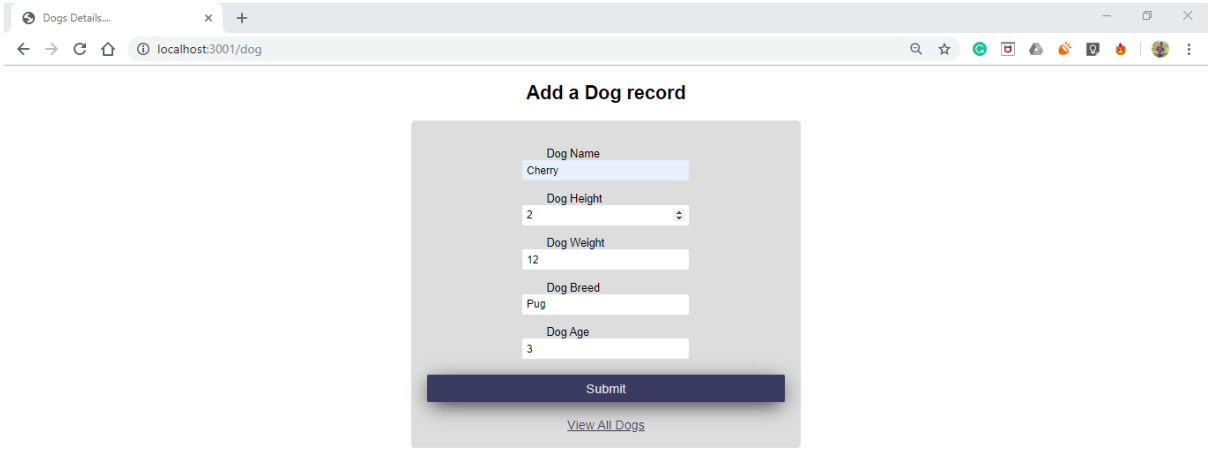
Launch your application in any browser at **http://localhost:3001**



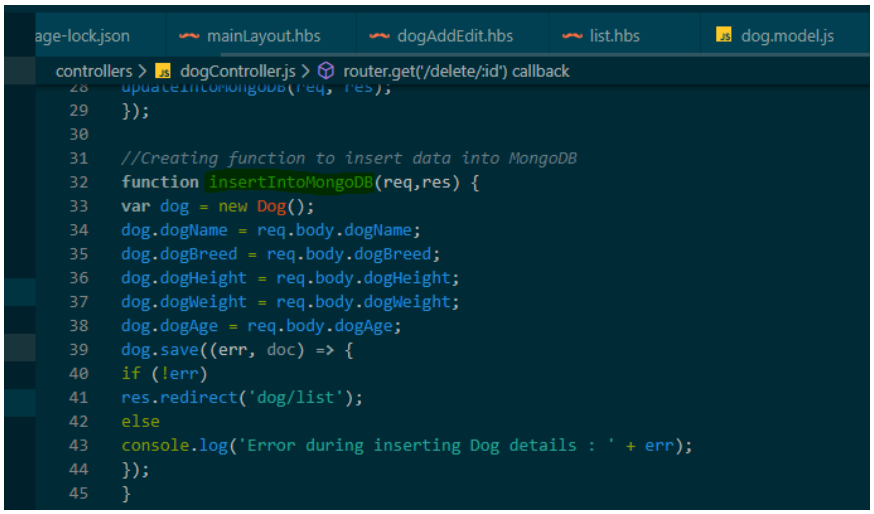
CRUD operations

Create :

Submit button will be submitting details filled in by the user into the mongoDB

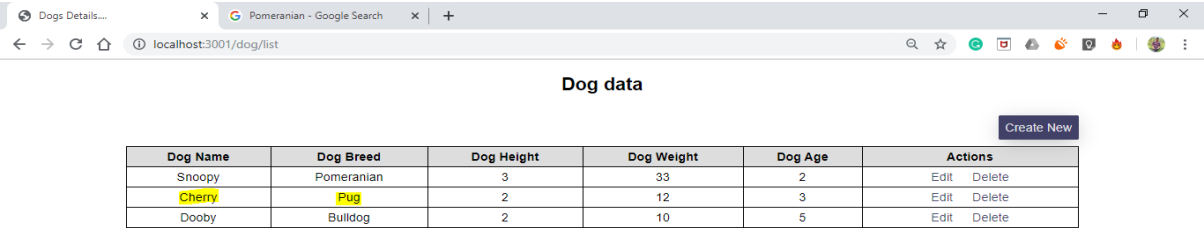


Code Snippet - Create



Read

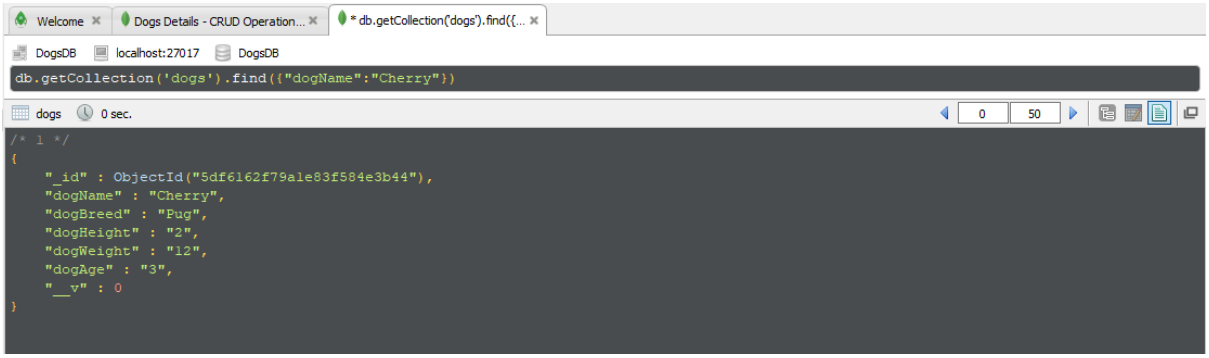
On click of view all records link in the create page, All existed records in Database are read and displayed onto UI



Dog Name	Dog Breed	Dog Height	Dog Weight	Dog Age	Actions
Snoopy	Pomeranian	3	33	2	Edit Delete
Cherry	Pug	2	12	3	Edit Delete
Dooby	Bulldog	2	10	5	Edit Delete

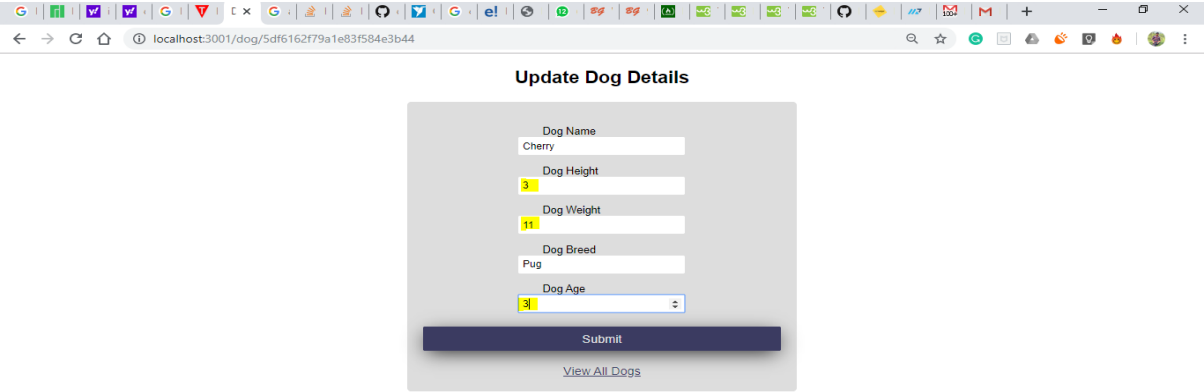
Code snippet - retrieve

MongoDB query



Update:

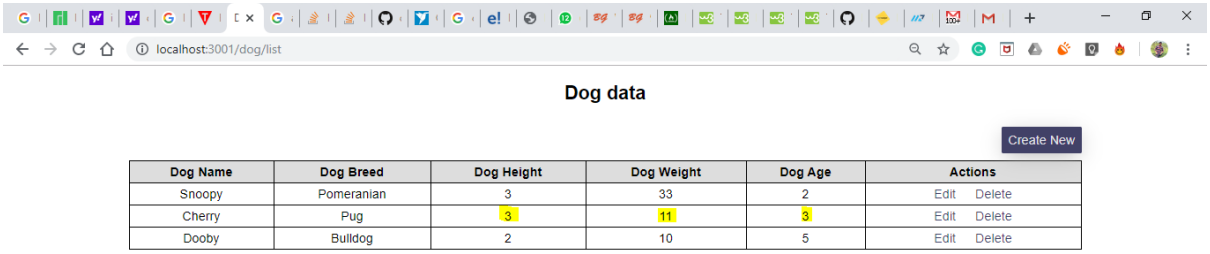
Let “Cherry” have to be updated with her Age, Height and Weight , Click on Edit button , then Submit



Code snippet - Update:

```
//Router to update a Dog using it's ID
router.get('/:id', (req, res) => {
Dog.findById(req.params.id, (err, doc) => {
if (!err) {
res.render("dog/dogAddEdit", {
viewTitle: "Update Dog Details",
dog: doc
});
}
});
});
```

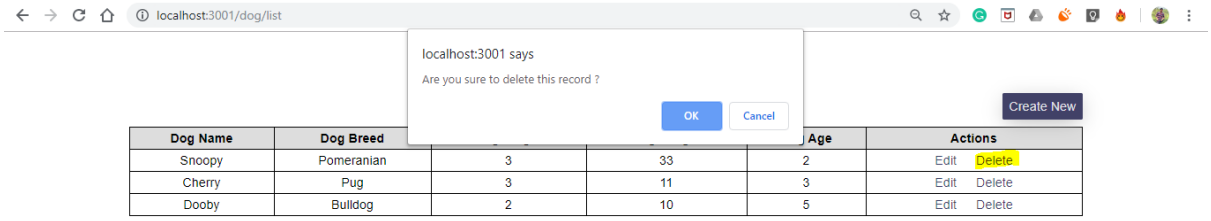
After Update, the data looks like



MongoDB Query

Delete:

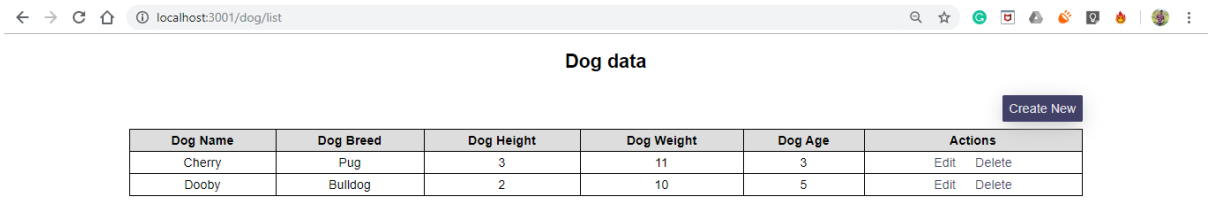
To delete a record . click on *Delete* button, a pop up asks for confirmation



Code snippet - Delete:

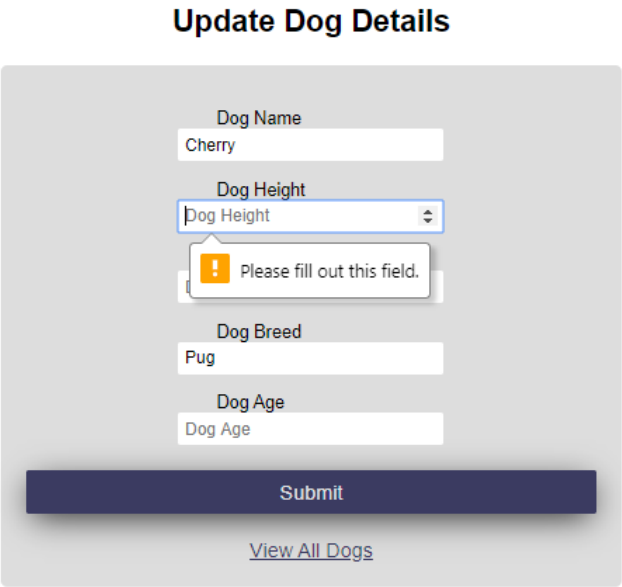
```
//Router Controller for DELETE request
router.get('/delete/:id', (req, res) => {
Dog.findByIdAndRemove(req.params.id, (err, doc) => {
if (!err) {
res.redirect('/dog/list');
}
else { console.log('Failed to Delete Dog Data: ' + err); }
});
});

module.exports = router;
```

After Deleting “**Snoopy**”



MongoDB query

Validations :



Some manual operations performed on “DogsDB” attached here

Dogs Details -
CRUD Operations in