



# 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"} (current: {"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 constructor.
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

The screenshot displays the MongoDB Atlas web interface. On the left sidebar, the 'mongoDB Atlas' logo is at the top, followed by 'All Clusters'. Under the 'CONTEXT' dropdown, 'mongodb-dog' is selected. The sidebar lists various sections: ATLAS (Clusters, Data Lake BETA), SECURITY (Database Access, Network Access, Advanced), PROJECT (Access Management, Activity Feed, Alerts, Integrations, Settings), and SERVICES (Charts). The main panel shows the 'Collections' tab for 'dogsDB.dogs'. It indicates a collection size of 365B, 3 total documents, and a 16KB index size. Below this, there's a 'Find' button and a filter input field containing '{"filter": "example"}'. The 'QUERY RESULTS 1-3 OF 3' section shows a single document: 

```
{
  "_id": ObjectId("5df83d1ece58424718bcc015"),
  "dogName": "Cherry",
  "dogBreed": "Pug",
  "dogHeight": "4",
  "dogHeight": "10",
  "dogAge": "3",
  "__v": 0
}
```

. The bottom of the image shows a Windows taskbar with several open applications and the system clock at 11:44 on 17-12-2019.

## Cluster Implementation:

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

The screenshot shows the MongoDB Atlas interface. At the top, it says "mongoDB Atlas" and "All Clusters". Below this, there's a "CONTEXT" section with a dropdown menu set to "mongodb-dog". The main heading is "Clusters". On the left sidebar, under "ATLAS", "Clusters" is selected. Below it are "Data Lake BETA", "SECURITY" (with sub-items: Database Access, Network Access, Advanced), and "PROJECT" (with sub-items: Access Management, Activity Feed, Alerts, Integrations, Settings). The main content area shows a search bar "Find a cluster..." and a list of clusters. One cluster, "mongodb01", is highlighted. It is a "SANDBOX" cluster, version 4.0.13. Below the cluster name are buttons for "CONNECT", "METRICS", "COLLECTIONS", and a menu icon. Further down, it shows "CLUSTER TIER: M0 Sandbox (General)", "REGION: AWS / N. Virginia (us-east-1)", "TYPE: Replica Set - 3 nodes", and "LINKED STITCH APP: None Linked". On the right, there are two performance graphs: "Operations" (R: 0, W: 0, 100.0/s) and "Connections" (2, 500 max). Both graphs show data for the "Last 6 Hours".

Created Database Access for below users

The screenshot shows the MongoDB Atlas "Database Access" page. At the top, it says "mongoDB Atlas" and "All Clusters". Below this, there's a "CONTEXT" section with a dropdown menu set to "mongodb-dog". The main heading is "Database Access". On the left sidebar, under "ATLAS", "Clusters" is selected. Below it are "Data Lake BETA", "SECURITY" (with sub-items: Database Access, Network Access, Advanced), and "PROJECT" (with sub-items: Access Management, Activity Feed). The main content area shows a blue banner at the top that says "We are deploying your changes (current action: configuring MongoDB)". Below this, there's a "Database Access" section with a sub-heading "MongoDB Users". There are two tabs: "MongoDB Users" (selected) and "MongoDB Roles". Below the tabs is a table with columns: "User Name", "Authentication Method", "MongoDB Roles", and "Actions". There are two rows of users, both with "SCRAM" authentication method and "atlasAdmin@admin" role. The first row has a user icon labeled "abhi" and the second row has a user icon labeled "kt". Each row has "EDIT" and "DELETE" buttons. At the top right of the table is a green button that says "+ ADD NEW USER".

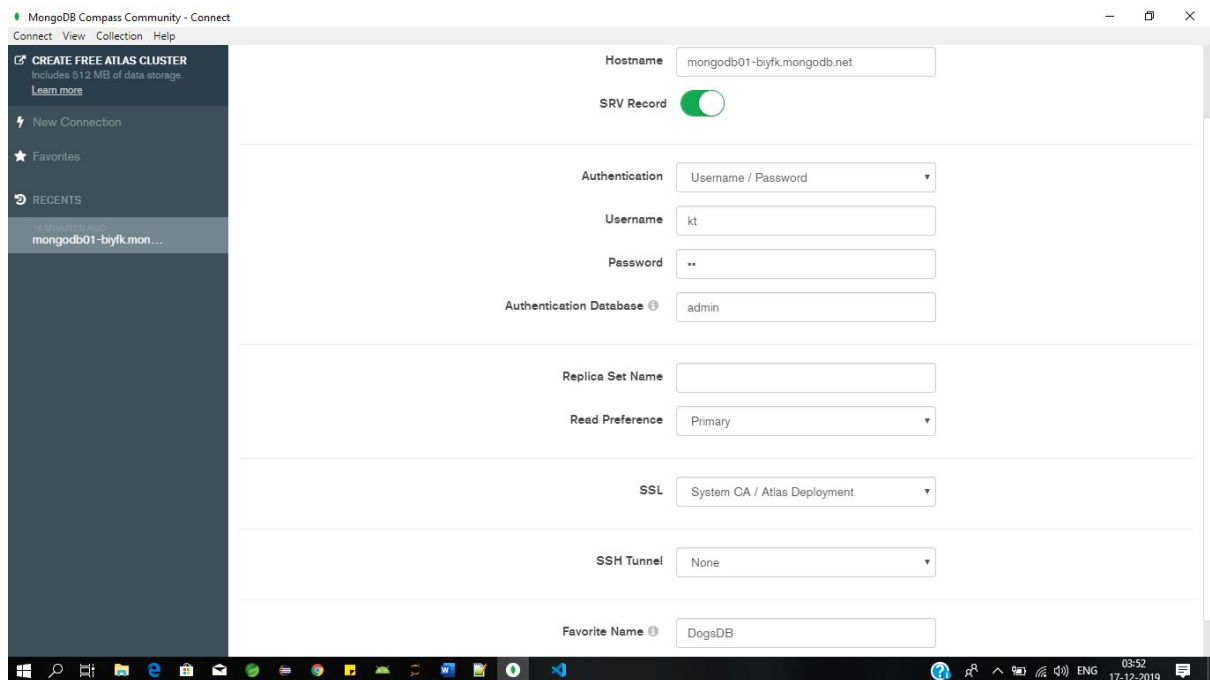
## Procedure to connect Cluster to Mongo DB Compass

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

**Hostname : mongodb01-biyfk.mongodb.net**

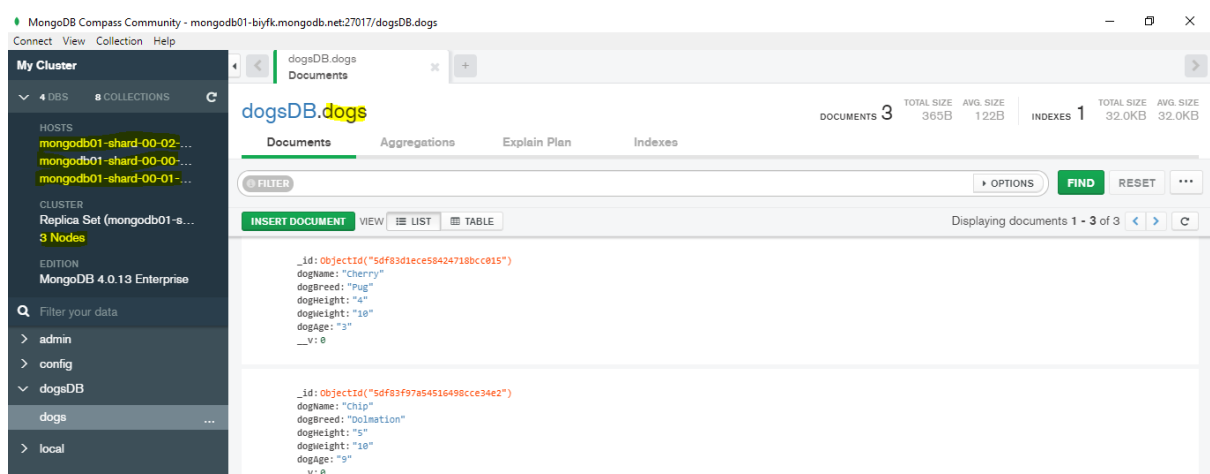
**Username : kt or abhi**

**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:

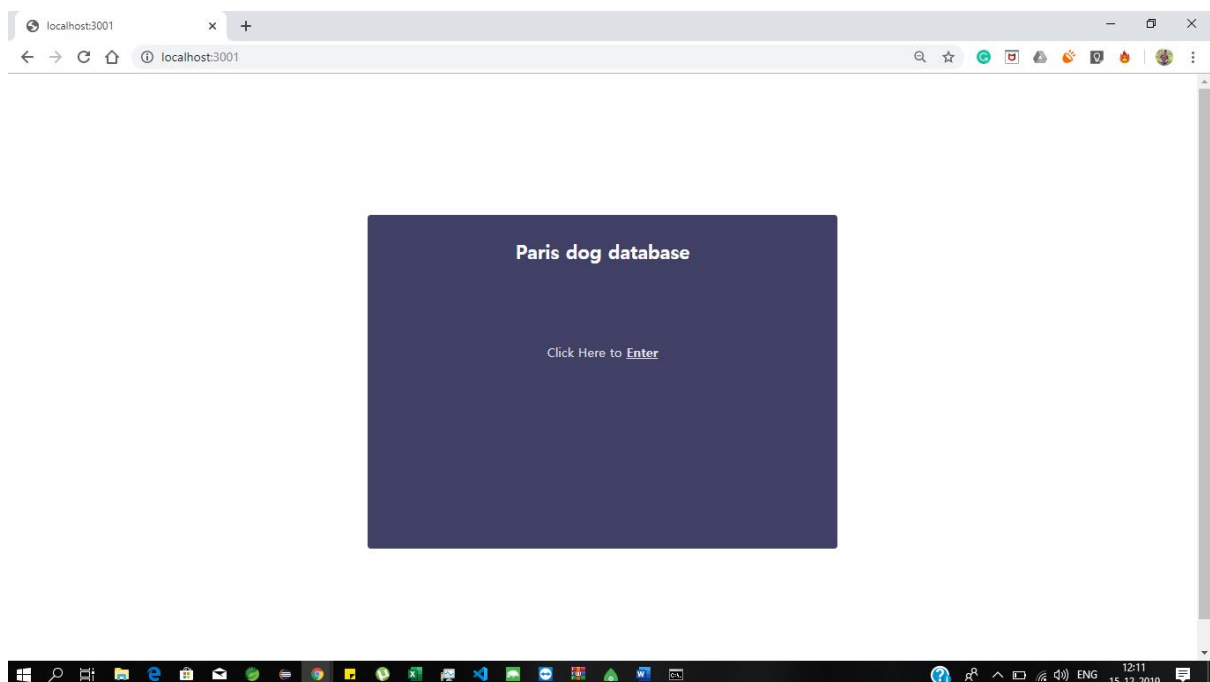
We have 5 fields

- Name
- Height
- Weight
- Breed
- Age

## Application structure

### Home page

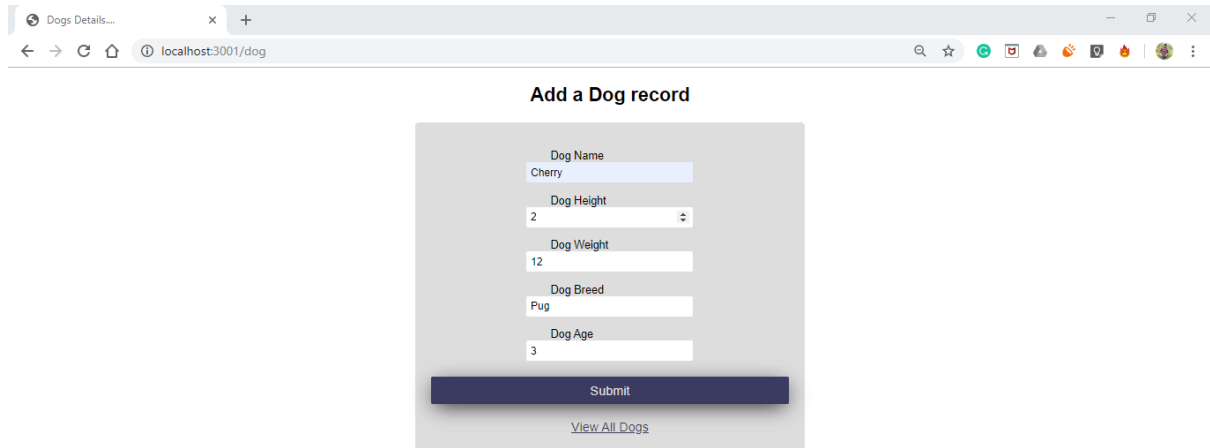
Launch your application in any browser at <http://localhost:3001>



## CRUD operations

### Create :

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



The screenshot shows a web browser window with the address bar displaying 'localhost:3001/dog'. The page title is 'Dogs Details...'. The main content is a form titled 'Add a Dog record'. The form contains the following fields:

- Dog Name: Cherry
- Dog Height: 2
- Dog Weight: 12
- Dog Breed: Pug
- Dog Age: 3

At the bottom of the form is a 'Submit' button. Below the button is a link that says 'View All Dogs'.



### Code Snippet - Create

```
age-lock.json  mainLayout.hbs  dogAddEdit.hbs  list.hbs  dog.model.js
controllers > dogController.js > router.get('/delete/id') callback
28 updateInMongoDB(req, res);
29 });
30
31 //Creating function to insert data into MongoDB
32 function insertIntoMongoDB(req,res) {
33   var dog = new Dog();
34   dog.dogName = req.body.dogName;
35   dog.dogBreed = req.body.dogBreed;
36   dog.dogHeight = req.body.dogHeight;
37   dog.dogWeight = req.body.dogWeight;
38   dog.dogAge = req.body.dogAge;
39   dog.save((err, doc) => {
40     if (!err)
41       res.redirect('dog/list');
42     else
43       console.log('Error during inserting Dog details : ' + err);
44   });
45 }
```

### Read

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

Dogs Details... x Pomeranian - Google Search x +

localhost:3001/dog/list

### Dog data

[Create New](#)

Dog Name	Dog Breed	Dog Height	Dog Weight	Dog Age	Actions
Snoopy	Pomeranian	3	33	2	<a href="#">Edit</a> <a href="#">Delete</a>
Cherry	Pug	2	12	3	<a href="#">Edit</a> <a href="#">Delete</a>
Dooby	Bulldog	2	10	5	<a href="#">Edit</a> <a href="#">Delete</a>

## Code snippet - retrieve

```

age-lock.json  mainLayout.hbs  dogAddEdit.hbs  list.hbs  dog.model.js
controllers > dogController.js router.get('/delete/id') callback
64   }
65
66   //Router to retrieve the complete list of available Dogs
67   router.get('/list', (req,res) => {
68     Dog.find((err, docs) => {
69       if(!err){
70         res.render("dog/list", {
71           list: docs
72         });
73       }
74       else {
75         console.log('Failed to retrieve the Dog data: '+ err);
76       }
77     });
78   });

```

## MongoDB query

Welcome x Dogs Details - CRUD Operation... x db.getCollection('dogs').find({... x

DogsDB localhost:27017 DogsDB

```
db.getCollection('dogs').find({"dogName":"Cherry"})
```

dogs 0 sec.

```

/* 1 */
{
  "_id" : ObjectId("5df6162f79a1e83f584e3b44"),
  "dogName" : "Cherry",
  "dogBreed" : "Pug",
  "dogHeight" : "2",
  "dogWeight" : "12",
  "dogAge" : "3",
  "__v" : 0
}

```

## Update:

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



**Update Dog Details**

Dog Name

Dog Height

Dog Weight

Dog Breed

Dog Age

[View All Dogs](#)

Code snippet - Update:

```

93 //Router to update a Dog using it's ID
94 router.get('/:id', (req, res) => {
95   Dog.findById(req.params.id, (err, doc) => {
96     if (!err) {
97       res.render("dog/dogAddEdit", {
98         viewTitle: "Update Dog Details",
99         dog: doc
100       });
101     }
102   });
103 });

```

After Update, the data looks like

**Dog data**

[Create New](#)

Dog Name	Dog Breed	Dog Height	Dog Weight	Dog Age	Actions
Snoopy	Pomeranian	3	33	2	<a href="#">Edit</a> <a href="#">Delete</a>
Cherry	Pug	3	11	3	<a href="#">Edit</a> <a href="#">Delete</a>
Dooby	Bulldog	2	10	5	<a href="#">Edit</a> <a href="#">Delete</a>

## MongoDB Query

Welcome x Dogs Details - CRUD Operation... x db.getCollection(dogs).find({... x

DogsDB localhost:27017 DogsDB

```
db.getCollection('dogs').find({"dogName":"Cherry"})
```

dogs 0.001 sec.

```

/* 1 */
{
  "_id" : ObjectId("5df6162f79a1e83f584e3b44"),
  "dogName" : "Cherry",
  "dogBreed" : "Pug",
  "dogHeight" : "3",
  "dogWeight" : "11",
  "dogAge" : "3",
  "__v" : 0
}

```

## Delete:

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

The screenshot shows a web browser at localhost:3001/dog/list. A modal dialog is open with the text "localhost:3001 says" and "Are you sure to delete this record?". It has "OK" and "Cancel" buttons. In the background, a table lists dogs: Snoopy (Pomeranian, 3, 33, 2), Cherry (Pug, 3, 11, 3), and Dooby (Bulldog, 2, 10, 5). Each row has "Edit" and "Delete" links. A "Create New" button is in the top right.

Dog Name	Dog Breed			Age		Actions
Snoopy	Pomeranian	3	33	2	Edit	Delete
Cherry	Pug	3	11	3	Edit	Delete
Dooby	Bulldog	2	10	5	Edit	Delete

Code snippet - Delete:

```
105 //Router Controller for DELETE request
106 router.get('/delete/:id', (req, res) => {
107   Dog.findByIdAndRemove(req.params.id, (err, doc) => {
108     if (!err) {
109       res.redirect('/dog/list');
110     }
111     else { console.log('Failed to Delete Dog Data: ' + err); }
112   });
113 });
114
115 module.exports = router;
```

After Deleting “Snoopy”

The screenshot shows the web browser at localhost:3001/dog/list. The table now only contains Cherry and Dooby. The "Delete" link for Snoopy is no longer present. A "Create New" button is still in the top right.

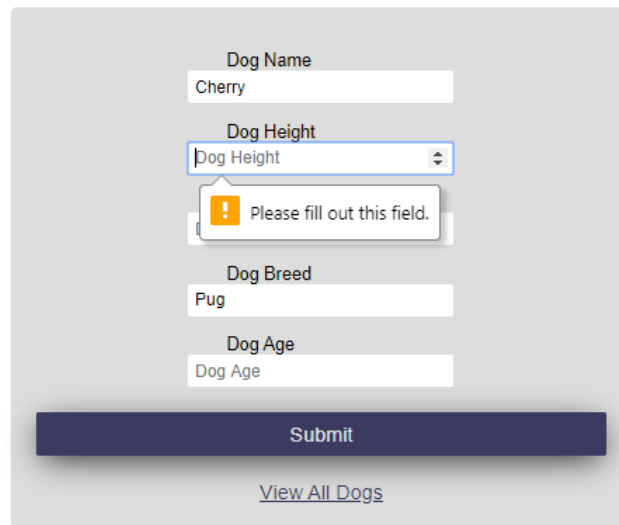
Dog Name	Dog Breed	Dog Height	Dog Weight	Dog Age		Actions
Cherry	Pug	3	11	3	Edit	Delete
Dooby	Bulldog	2	10	5	Edit	Delete

MongoDB query

The screenshot shows the MongoDB Shell interface. The command bar contains the query: `db.getCollection('dogs').find({"dogName":"Snoopy"})`. The status bar shows "0.001 sec." and "Fetched 0 record(s) in 1ms".

*Validations :*

### Update Dog Details



The screenshot shows a web form titled "Update Dog Details" with the following fields and values:

- Dog Name:** Cherry
- Dog Height:** Dog Height (This field is highlighted with a blue border and has a validation error message: "Please fill out this field.")
- Dog Breed:** Pug
- Dog Age:** Dog Age

At the bottom of the form is a dark blue "Submit" button and a link labeled "View All Dogs".

Some manual operations performed on “DogsDB” attached here



Dogs Details -  
CRUD Operations in