Part 3 Testing the web application as a client

First startup the web server, this should be the same on any platform

Open up the terminal, and navigate to the directory with the web application project

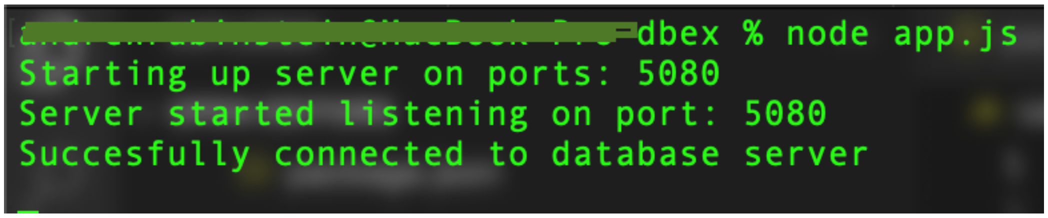
If you haven’t already run the command
npm install

This will install any dependencies for the project, like the express, and dotenv libraries.

Next run

Node app.js

After pressing enter you should see the following popup if everything is correctly configured to connect to your database:



```
[redacted]dbex % node app.js
Starting up server on ports: 5080
Server started listening on port: 5080
Succesfully connected to database server
```

Once you have the web server running we can test the service, we should be able to create new users, and get all of them represented as json.

Test the web app

ON MAC OSX AND LINUX:

On unix like systems we can use the curl program from the terminal

```
curl -d '{"userName":"value1", "password":"value2"}' -H
"Content-Type: application/json" -X POST http://
localhost:5080/
```

Will enter a record into the users collection that we can then query later. You should see the html output:

```
<h1>Record posted!</h1>%
```

If you see anything else like:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot POST /data</pre>
</body>
</html>
```

An error has occurred please request help

To then request the data in the collection users from our web service we can again use curl, when we enter the following parameters from curl we will make a get request, and our get handler will be called

```
 curl http://localhost:5080
```

The above line of code should print out the response:

```
[{“_id":"62992a4883f4557d64b8e7e5","userName":"value1","pas
sword":"value2","isAdmin":false,"__v":0}]
```

If we examine the above code we can see all the data defined in our userSchema, with the addition of an “__id” field which is a unique identifier mongodb will automatically create for each object record.