MongoDB and NodeJS

Setting things up

1. Try to create a folder Week10 and inside it create a file called app.js
2. In Command line or Terminal run **npm init** to create package.json file. Package.json file contains all libraries/dependencies needed for the project.
3. In order to make NodeJS work with MongoDB we need to install 3 dependencies as follows: express (web), body-parser (json), mongoose (client)

npm install --save express body-parser mongoose

4. Once we have all required libraries we start to code our **app.js**:

- First thing we need to import all libraries:

```
//import required libraries
var app = require('express')()

var mongoose = require('mongoose')

var bodyParser = require('body-parser')

app.use(bodyParser.json())
```

- We need to create db object which connects to MongoDB server:

```
//create a connection to database
mongoose.connect('mongodb://localhost/rmit')
```

```
You can replace it with Mongo Atlas connection string, i.e.
```

Define a schema. It is like a table

```
//define a "table" structure
var StudentSchema = new mongoose.Schema({
   name: String
})
```

Create a model and map it to a collection in MongoDB

```
//create a model Student ==> students (database collection)
//Teacher => teachers , Course => courses
var Student = mongoose.model('Student', StudentSchema)
```

5. Start to code the endpoints:
The first endpoint is get all students:

```
app.get('/students', function(req, res){
   Student.find({}, function(err, students){
       res.send(students)
   })
})
```

The find function has 2 params: 1st is condition and 2nd is a callback function.

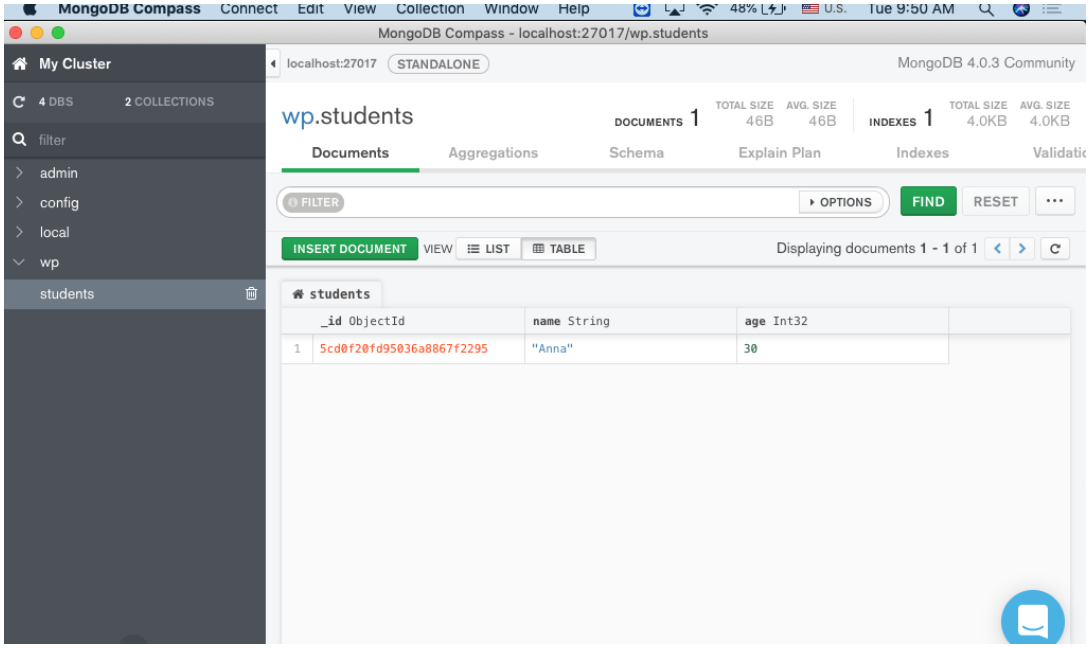
The result will be stored in docs. We just need to return by: res.send(students)

6. We start our app by adding**: app.listen(9000)**

Run it by: node app.js

7. Make sure that you have started the MongoDB server. On Windows it often runs as a Service. You can go to Windows/Services to search and check if the MongoDb server is running.

8. You can test it by starting Compass program:
Because we connect to a database named **rmit. Make sure that you have created this db. If the db does not exist, MongoDB will create a new one**

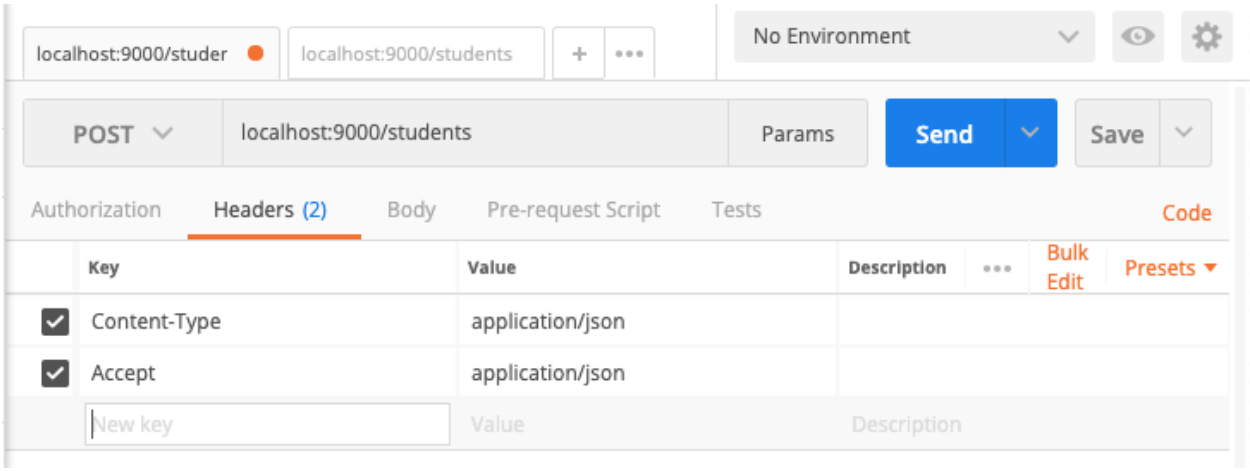


Define POST endpoint:
The create() function also takes 2 params: 1st is request body, 2nd is the callback function:

```
app.post('/students', function(req, res){
   Student.create(req.body, function(err, student){
       res.send(student)
   })
})
```

We can test it using postman:

First, setup the headers, we need to put Content-Type and Accept which both will take application/json



Then, setup the body:

```
{
   "id": "s1",
   "name": "Tom"
}
```

Define endpoints for delete, put, and get one:

```
app.delete('/students/:id', function(req, res){
   Student.deleteOne({_id: req.params.id}, function(err, result){
       res.send(result)
   })
})

app.put('/students', function(req, res){
   Student.findOneAndUpdate({_id: req.body.id},{ name: req.body.name},
function(err, result){
       res.send(result)
   })
})


app.get('/students/search/:keyword', function(req, res){
```

```
   Student.find({name: req.params.keyword}, function(err, result){
       res.send(result)
   })
})
```

Get complete code here:

```
var app = require('express')()
var mongoose = require('mongoose')
var bodyParser = require('body-parser')
app.use(bodyParser.json())

const url =
"mongodb+srv://user:password@cluster0.yourserver.mongodb.net/mydb?retryWrites=true&w=majority"
mongoose.connect(url)

//define a "table" structure
var StudentSchema = new mongoose.Schema({
   id: String,
   name: String
})
//create a model Student ==> students (database collection)
//Teacher => teachers , Course => courses
var Student = mongoose.model('Student', StudentSchema)

app.get('/students', function(req, res){
   Student.find({}, function(err, students){
       res.send(students)
   })
})
```

```
app.post('/students', function(req, res){
   Student.create(req.body, function(err, student){
       res.send(student)
   })
})
 app.delete('/students/:id', function(req, res){
   Student.deleteOne({_id: req.params.id}, function(err, result){
       res.send(result)
   })
})
 app.put('/students', function(req, res){
   Student.findOneAndUpdate({_id: req.body.id},{ name: req.body.name}, function(err,
result){
       res.send(result)
   })
})

app.get('/students/search/:keyword', function(req, res){
   Student.find({name: req.params.keyword}, function(err, result){
       res.send(result)
   })
})


app.listen(9000)
```

Video tutorials
https://www.youtube.com/watch?v=aMUxBSja4m0&list=PLA7OXTqVjVmc-IafQpLr-6LCAg7ODLE-Y

Official document on Mongoose

To complete the assignment 3, students need to learn more complex query from Mongoose official document:

https://mongoosejs.com/docs/queries.html