PART 1

- Explanation of what MongoDB and how to install it
- Run Mongo and Mongo shell
- Run basic mongo commands
    - How to create a database
    - How to create a collection
    - How to use basic commands such as insert, find , findOne

- Explain Mongoose and connect our express.js app to MongoDB via Mongoose

---

MongoDB

MongoDB is a database application which allows us to store data in a persistent manner. This means even after we refresh the browser the data will still be available. MongoDB stores data in a format and syntax that is near identical to JSON. If you understand JSON you can understand how to work with MongoDB data sets. Technically MongoDB refers to their data format as BSON (Binary JSON)

Installing MongoDB

To install go to: https://www.mongodb.org/downloads#production

Once you pick your installer download it and run it.

Once installed go to the windows programs folder (or on Mac the applications folder) and inside the mongoDB folder open each folder until you have reached the Bin folder – then open it. You should see a list of files that begin with “mongo”. Find and create a shortcut of the **mongo** file and the **mongod** file to your desktop.

Launching MongoDB

To launch mongoDB click the **mongod** shortcut you just created on your desktop, then wait for it to load, then click the shortcut titled **mongo**

From here on out you will work in the **mongo** terminal window, however you must keep **mongod** open in the background to work.

Creating a database

To create a database in mongo all you need to do is run the following command:

```
use name-of-database
```

So for example do:

```
use todo
```

If you want to view all available databases use:

```
show dbs
```

To delete a database first “use” it then

```
db.dropDatabase()
```

Adding a collection

A collection is basically an umbrella term for a list of objects. The following code adds a new item to a collection called todos.

```
db.todos.insert({item:"get milk"});
```

To see the item run the following:

```
db.todos.find()
```

The result will return

```
{ "_id" : ObjectId("5631e43fbeac04a6f5b95841"), "item" : "get milk" }
```

If you add multiple items to a collection the find() command will list them all.

If you simply want to see what one item in a collection looks like you can do this:

```
db.todos.findOne();
```

If you want to view all collections in the currently selected database do this:

```
Show collections
```

If you want to find an object based on its key/values you can do this:

```
db.todos.find({"item" : "get milk"})
```

You can also remove items based on this criteria

```
db.todos.remove({"item" : "get milk"})
```

There are many commands available to search and modify data with MongoDB and we are only going over the very basics. Search the documentation for more information if need be.

Mongoose

Now that you have seen some mongodDB basics, it is time to connect mongoDB to node.js. This can be done using a module called mongoose.js

To begin, make a copy of the non-mongo version of your app. We are going to modify this and make it interact with mongoDB.

Modify the package of the copy to use mongoose.

```
{
    "name": "todo_list",
    "version": "1.0.0",
    "main": "app.js",
    "author": "",
    "license": "ISC",
    "dependencies": {
        "express": "*",
        "express-handlebars": "*",
        "body-parser": "*",
        "mongoose": "*"
    },
    "description": ""
}
```

Now In your app.js file set the require method after all the other required modules

```
var express = require("express");
var app = express();
var expressHbs = require('express-handlebars');
var bodyParser = require('body-parser');
var mongoose = require('mongoose');
```

```
mongoose.connect('mongodb://localhost/app');  // connection code
```

Now, empty out all the route bodies in your code. The result should look like the following (feel free to copy and paste this for brevity)

```
//_____________________________________________________BEGIN app
var express = require("express");
var app = express();
var expressHbs = require('express-handlebars');
var bodyParser = require('body-parser');
var mongoose = require('mongoose');

app.use(bodyParser.urlencoded({
    extended: true
}));

app.use(express.static('public'))
app.set('view engine', 'hbs');
app.engine('hbs', expressHbs({
    extname: 'hbs'
}));
mongoose.connect('mongodb://localhost/app');



app.get("/", function(req, res) {

});
```

```
app.post("/client_to_server", function(req, res) {

});

app.get("/delete/:id", function(req, res) {

});

app.get("/edit/:id", function(req, res) {

});

app.post("/update/:id", function(req, res) {

});

app.get('/static', function(req, res) {
    res.sendFile('static_example.html', {
        root: "public"
    });
});

app.get("*", function(req, res) {
    res.redirect("/")
});
```

```
app.listen(3000, function(err) {
    if (err) {
        console.log('Server is not working ');
    } else {
        console.log('Server works')
    }
});

//_____________________________________________________END app
```

Creating a Mongoose model

A model is an object template that data from the client is expected to conform to. The purpose of Mongoose is to create these models as a means to insure the data that users enter is the right kind of data. You can look at these templates (or models) as a type of filter. In Mongoose this is referred to as ODM - or Object Data Modeling.

To create your first model type the following below the mongoose connect method. This code is a template (or model) called Todo, which is designed to contain a property called task. Task is a required field and has to be a string in order to be saved.

```
mongoose.connect('mongodb://localhost/app');

var Todo = mongoose.model('Todo', {
    task: {
        type: String,
        required: true
```

```
    }
});
```

Storing a new Todo in the database

Create an initial route for the homepage:

```
app.get("/", function(req, res) {
    res.render("index")
});
```

Then create a post request to send data to database:

```
app.post("/client_to_server", function(req, res) {
    var task = new Todo({
        task: req.body.userData
    }).save()
    res.redirect("/")
});
```

In the MongoDB shell use the app database:

```
use app
```

Now make a Todo post from the homepage.

You can then check if the data made it to the database using the following command in the Mongo Shell:

```
db.todos.find()
```

This should return something like:

```
{ "_id" : ObjectId("56352cae7e9242a0009cfdc0"), "task" : "ZOINK", "__v" : 0 }
```

Retrieving the Todos from the database and placing them on the homepage

In app.js modify the homepage get route to look like the following.

```
app.get("/", function(req, res) {
    Todo.find(function(err, arrayOfItems) {
        res.render("index", {
            item: arrayOfItems
        })
    })
});
```

In the index.hbs modify the code to the following

```
<body>
            <form method="post"  action="client_to_server">
                  <input type="text" name="userData">
                  <input type="submit">
            </form>
            <ul>
            {{#each item}}
                  <li>
                        <a href="delete/{{this.id}}">
                              {{this.task}}:
                        </a>
                        <a href="edit/{{this.id}}">
                              <b>Edit</b>
                        </a>
                  </li>
            {{/each}}
            </ul>
      </body>
```

Deleting Todos

```
app.get('/delete/:id', function(req, res) {
    Todo.findById(req.params.id, function(err, todo) {
        if (!err) {
            todo.remove();
        } else {
            return err
        }
    });
    return res.redirect('/');
})
```

Editing/Updating Todos

Modify app.js

```
app.get("/edit/:id", function(req, res) {
    Todo.findById(req.params.id, function(err, item) {
        res.render("edit", {
            todo: item
        })
    })
})
```

Modify edit.hbs:

```
<body>
      <h1> Editing</h1>
      <form method="post" action = "/update/{{todo._id}}">
            <input type="text" name = "updated_task">
            <input type = "submit">
      </form>
</body>
```

Then modify apps.js

```
app.post('/update/:id', function(req, res) {
    Todo.findById(req.params.id, function(err, todo) {
        todo.task = req.body.updated_task
        todo.save();
    });
    return res.redirect('/');

})
```