Node.JS and MongoDB
Resit Project Application Guide

Hilal EKİCİ

03/02/2021

1. Introduction

This is a guide of the **GMT 351 – Geospatial Data Management** resit project application. It will explain how to create an application that uses Node.JS, MongoDB and npm.

In addition, this guide will guide you in using the Windows operating system. You have to search for other operating systems yourself.

We will follow these steps to implement this application:

1. To create *Node.js* package
2. To create *Node.js* environment
3. To connect *MongoDB* database
4. To access database through *JavaScript*
5. To display and insert data to database.
6. To filter data according to tree height
7. To sort data according to all inputs: name, latitude, longitude and tree height
8. To delete and update data

Explanation of the Tools

Javascript

JavaScript, sometimes abbreviated as JS, is a high-level, interpreted programming language that lets you implement complex elements on web pages.
It is not the same thing with Java. They are different programming languages.

Node.js

Node.js is run-time environment includes everything you need to execute a program written in JavaScript. It also is a JavaScript runtime built on Chrome's V8 JavaScript engine.

In this project, I will use Node.js as a backend development platform. A backend developer uses Node.js for backend work. The Node.js allows a developer to handle data from the front end and build scalable network applications. It ables to process many simultaneous user requests.

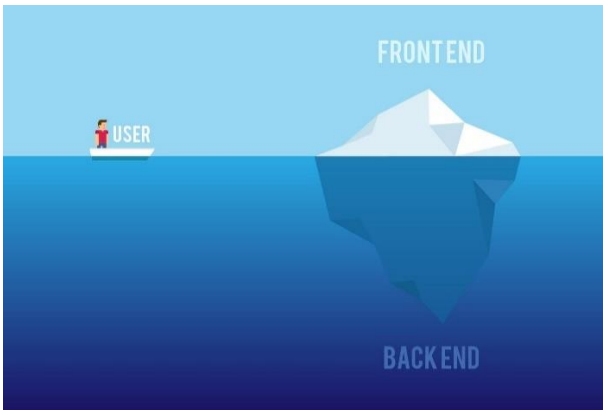
Front-End vs Back-End

Front-End and Back-End are most popular terms used in web development. The terms are very crucial for web development. They are quite different from each other. They both need to communicate and operate effectively with the other as a single unit to improve the website's functionality.

Front-End is the part of a website that user interacts with directly. It also knowns as 'client side' of the application. It includes everything that users experience directly. Text colors, styles, images, buttons, colors, and everything that comes across visually when you open a website.

Back-End means that for web development that occurs at the back end of programs. A back-end developer writes code to help a database and application communicate. Essentially, a back end developer handles what you do not see.

In this image, we can say that, a user only sees small part of an application. Big details are in the backend side.



MongoDB

MongoDB is an object-oriented, simple, dynamic, and scalable NoSQL database. It based on the NoSQL document store model. The data objects are stored as separate documents inside a collection instead of storing the data into the columns and rows of a traditional relational database.

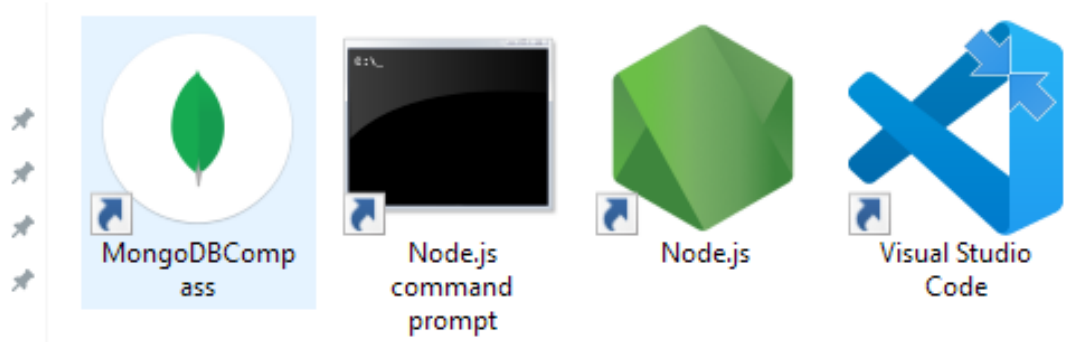
Application Required

Steps

Step 1: Required Tools

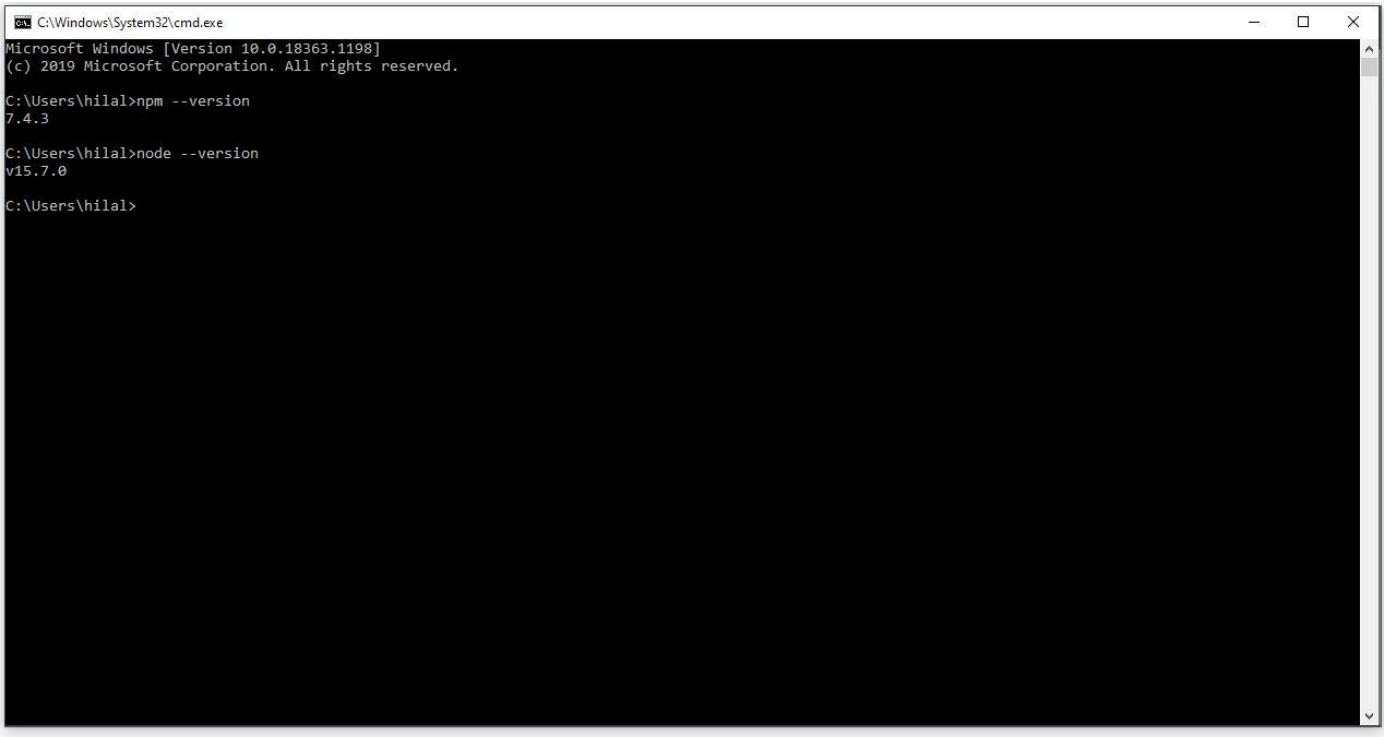
The required tools for this application: Visual Studio Code, MongoDB, NodeJS, Npm
You can use another text editor instead of Visual Studio Code
You should install Node.Js, Npm and MongoDB Compass to show the table and data

1. VS Code https://code.visualstudio.com/
2. Node.js with Npm https://nodejs.org/en/download/releases/
3. MongoDB Compass https://www.mongodb.com/products/compass



Step 2: Check for installs

Before implementing the project, we should check whether MongoDB, Node.js and Npm installed on the computer. We should also check from Node.js's own site that Node.js and Npm have compatible versions. This is the website: https://nodejs.org/en/download/releases/

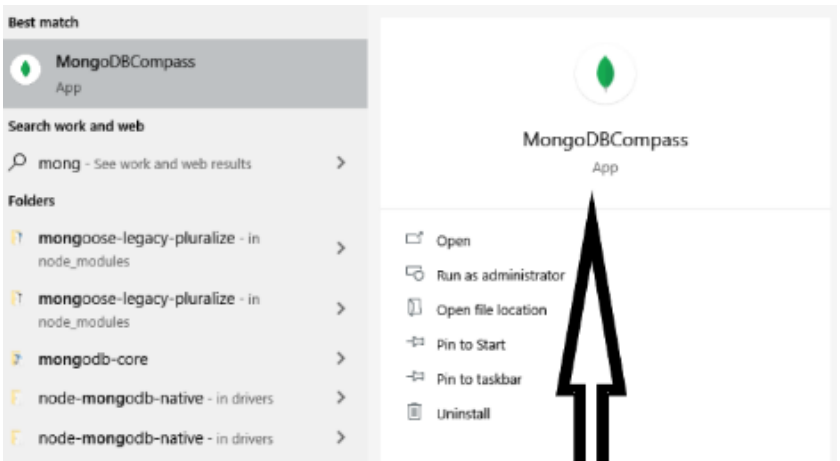


Version	LTS	Date	V8	npm	NODE_MODULE_VERSION[1]			
Node.js 15.7.0		2021-01-25	8.6.395.17	7.4.3	88	Downloads	Changelog	Docs
Node.js 15.6.0		2021-01-14	8.6.395.17	7.4.0	88	Downloads	Changelog	Docs
Node.js 15.5.1		2021-01-04	8.6.395.17	7.3.0	88	Downloads	Changelog	Docs
Node.js 15.5.0		2020-12-22	8.6.395.17	7.3.0	88	Downloads	Changelog	Docs
Node.js 15.4.0		2020-12-09	8.6.395.17	7.0.15	88	Downloads	Changelog	Docs
Node.js 15.3.0		2020-11-24	8.6.395.17	7.0.14	88	Downloads	Changelog	Docs
Node.js 15.2.1		2020-11-16	8.6.395.17	7.0.8	88	Downloads	Changelog	Docs
Node.js 15.2.0		2020-11-10	8.6.395.17	7.0.8	88	Downloads	Changelog	Docs
Node.js 15.1.0		2020-11-04	8.6.395.17	7.0.8	88	Downloads	Changelog	Docs
Node.js 15.0.1		2020-10-21	8.6.395.17	7.0.3	88	Downloads	Changelog	Docs
Node.js 15.0.0		2020-10-20	8.6.395.16	7.0.2	88	Downloads	Changelog	Docs
Node.js 14.15.4	Fermium	2021-01-04	8.4.371.19	6.14.10	83	Downloads	Changelog	Docs
Node.js 14.15.3	Fermium	2020-12-17	8.4.371.19	6.14.9	83	Downloads	Changelog	Docs
Node.js 14.15.2	Fermium	2020-12-15	8.4.371.19	6.14.9	83	Downloads	Changelog	Docs
Node.js 14.15.1	Fermium	2020-11-16	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.15.0	Fermium	2020-10-27	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.14.0		2020-10-15	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.13.1		2020-10-07	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.13.0		2020-09-29	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.12.0		2020-09-22	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.11.0		2020-09-15	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.10.1		2020-09-10	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.10.0		2020-09-08	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.9.0		2020-08-27	8.4.371.19	6.14.8	83	Downloads	Changelog	Docs
Node.js 14.8.0		2020-08-11	8.4.371.19	6.14.7	83	Downloads	Changelog	Docs
Node.js 14.7.0		2020-07-29	8.4.371.19	6.14.7	83	Downloads	Changelog	Docs
Node.js 14.6.0		2020-07-20	8.4.371.19	6.14.6	83	Downloads	Changelog	Docs
Node.js 14.5.0		2020-06-30	8.3.110.9	6.14.5	83	Downloads	Changelog	Docs
Node.js 14.4.0		2020-06-02	8.1.307.31	6.14.5	83	Downloads	Changelog	Docs
Node.js 14.3.0		2020-05-19	8.1.307.31	6.14.5	83	Downloads	Changelog	Docs

- npm --version or npm --v
- node --version or node --v

When you see if node.js and npm versions are compatible, then this

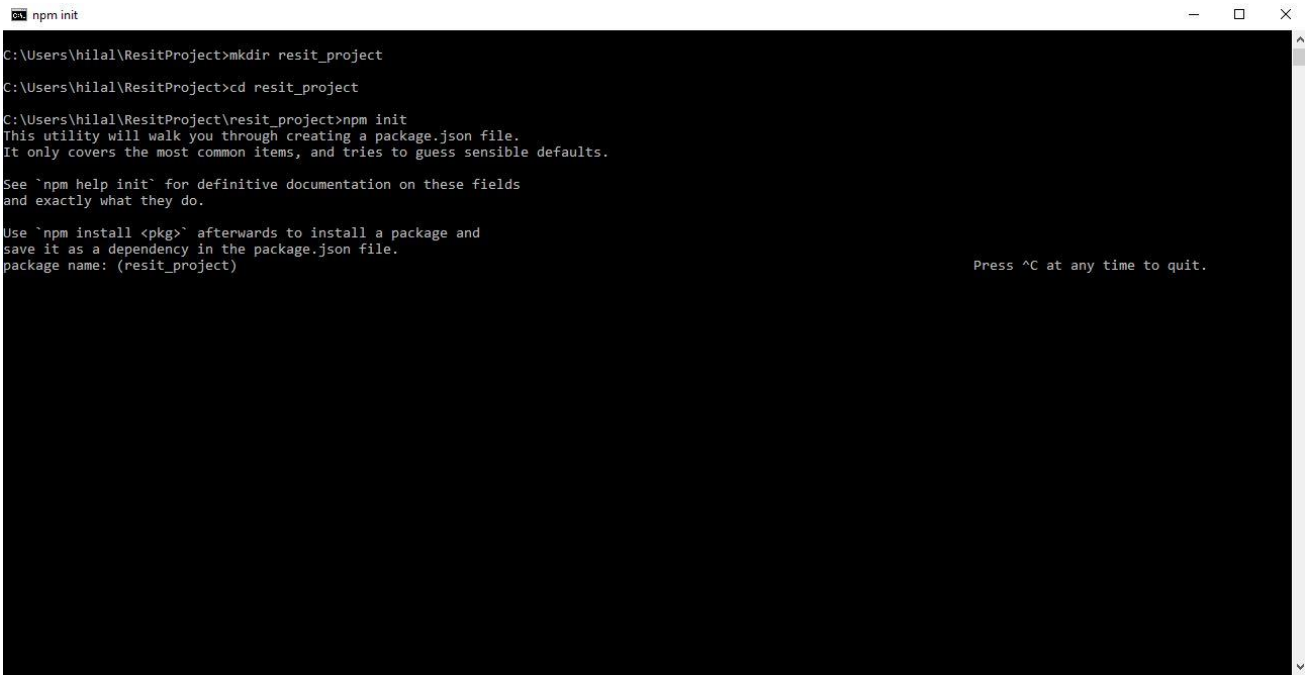
For **mongoDB**: Start > Search > MongoDB



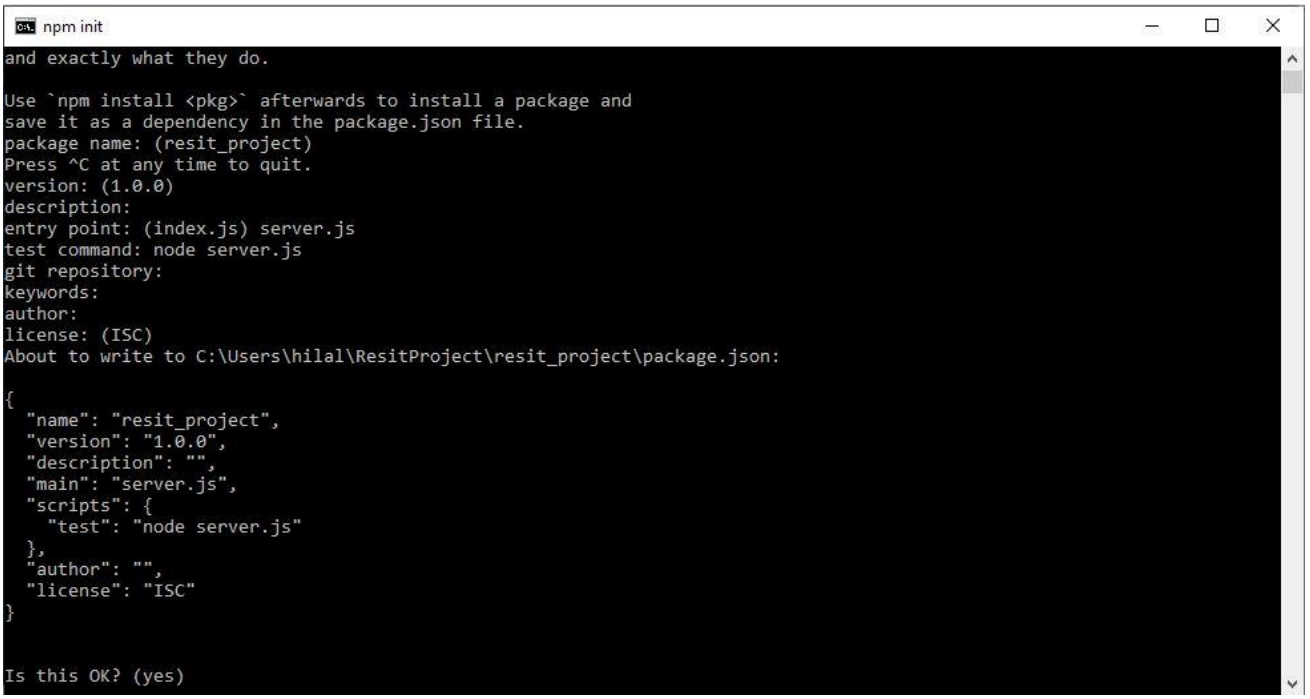
Step 3: Creating Package

After that, we will create our package for the application. As you show from the screenshot;

- Firstly, we open **cmd** with run as administrator (command line).
- Change your path with **cd** command where you want the project in.
- Create project directory using **mkdir** command.

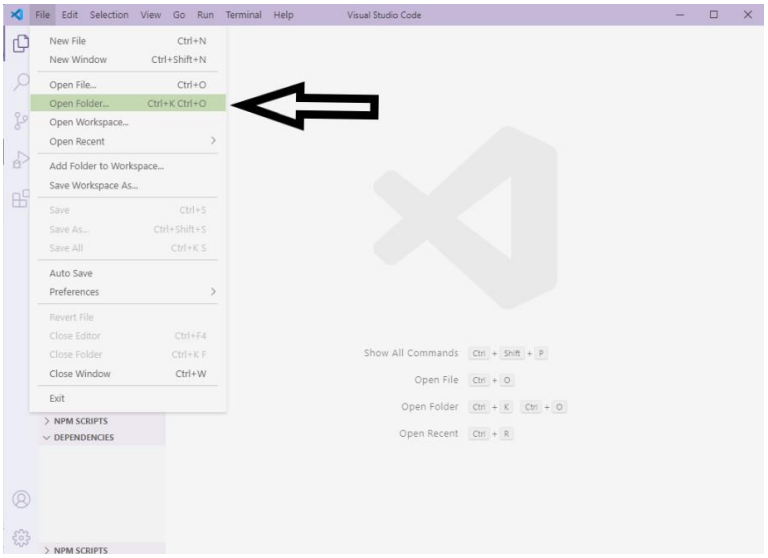


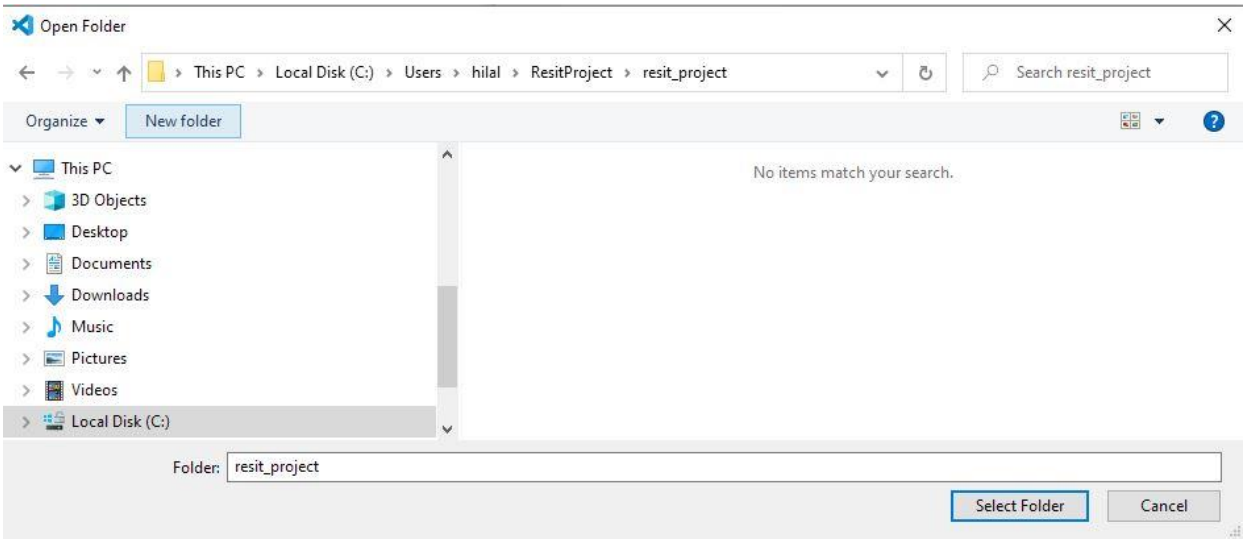
- After that, we should run **npm init** to set up a new **npm** package. This command shows questions to create package json. You can press enter to skip questions, all of them will default value. Only part is **Entry Point**. The project's entry point (meaning the project's main file), The project's test command (to trigger testing with something like Standard). We set it as **server.js**



Step 4: Creating Environment

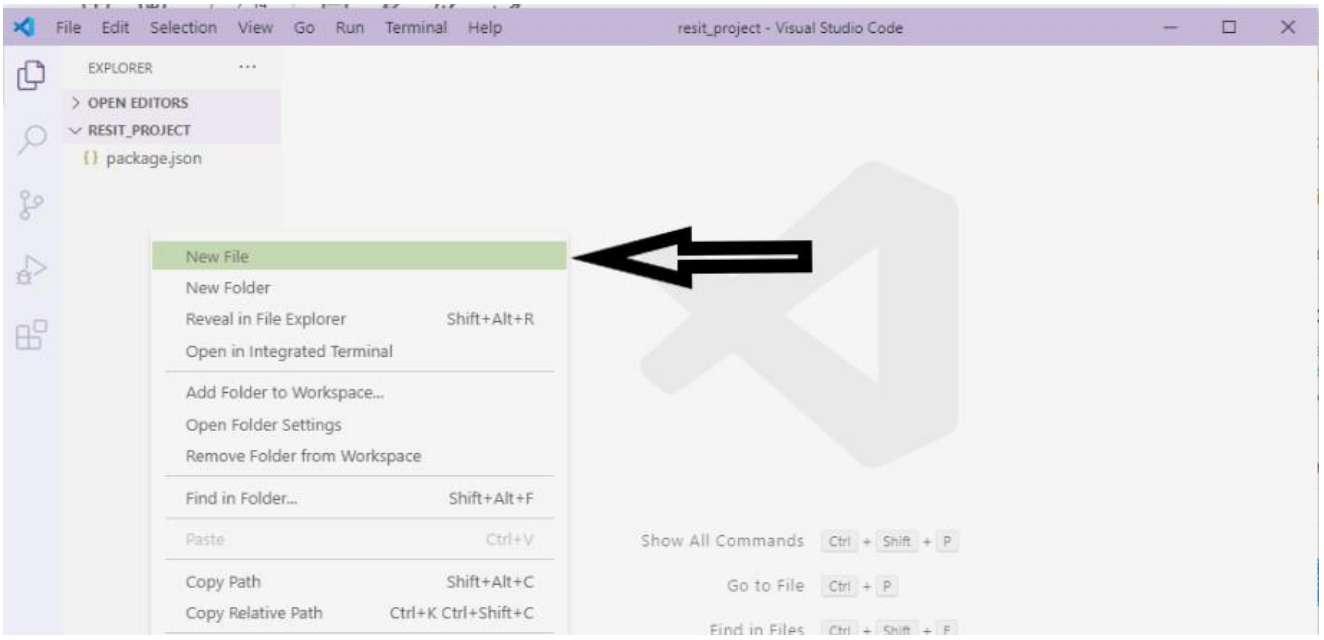
- After, installed VS Code, open it.
- File > Open folder
- Open the folder containing package.json



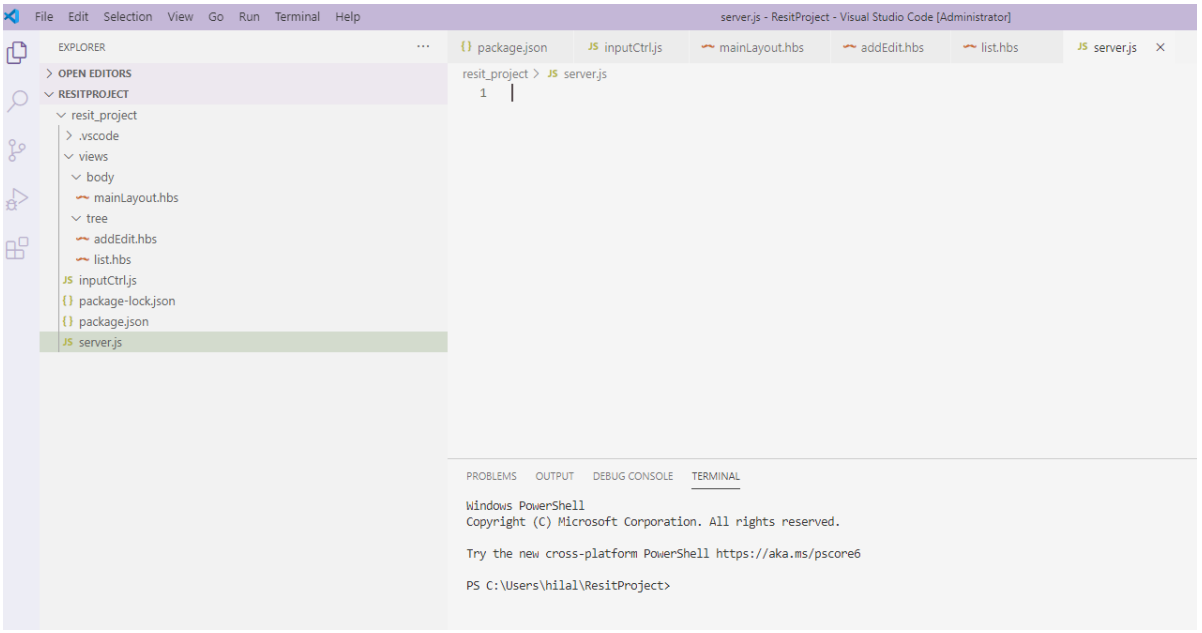


Step 5: Creating Code File

- Right click on VSCode > "**New File**"
- Give name "**server.js**"
- Do same thing for "**inputCtrl.js**", "**mainLayout.hbs**", "**list.hbs**", "**addEdit.lbs**"



After all steps, you should check the application folder:

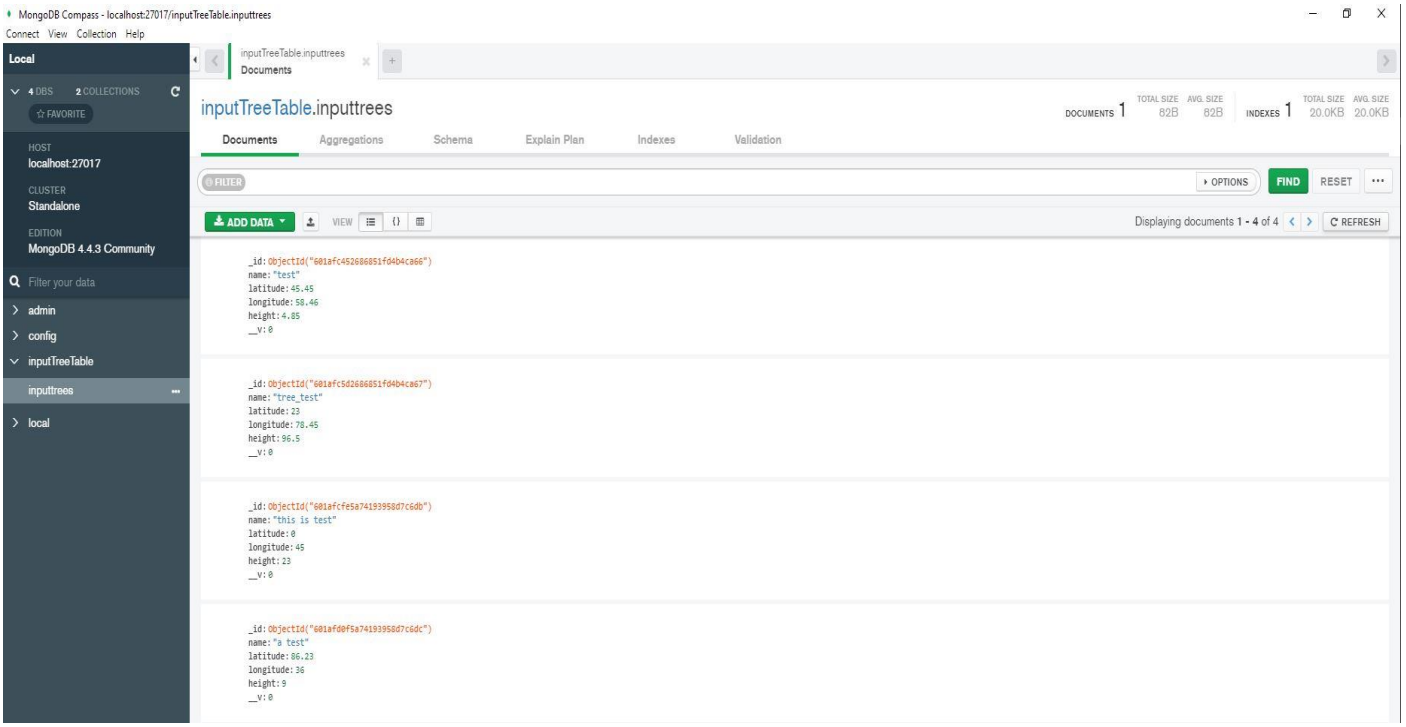


Step 6: Designing Database

In MongoDB, you can handle all database detail from your code files.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/inputTreeTable', { useNewUrlParser: true }, (err) => {
    if (!err) {
        console.log('DB connection is successful.')
    }
    else {
        console.log('Error in DB connection : ' + err)
    }
});
var inputTable = new mongoose.Schema({
    name: {
        type: String,
        required: 'This field is required.'
    },
    latitude: {
        type: Number,
        required: 'This field is required.'
    },
    longitude: {
        type: Number,
        required: 'This field is required.'
    },
    height: {
        type: Number,
        required: 'This field is required.'
    }
});
mongoose.model('InputTree', inputTable);
```

We use **mongoose** for connecting to the database. I choosed table name: **inputTreeTable**. After that, created a schema with four fields: ***name***, ***latitude***, ***longitude*** and ***height***. None of them can be empty. These code lines ensures inputs' **validation**. User cannot enter string data for latitude, longitude and height. Beside, user cannot enter number data for name field.



Step 7: Packages

We should install packages now. To install packages, use **npm install** command.

npm install package_name: Install the dependencies in the local node_modules folder.

We need run these commands from terminal:

- **npm install mongoose**

  This package is for database operations.

- **npm install express**

  Express.js is a Node.js web application server framework, which is specifically designed for building single-page, multi-page, and hybrid web applications..

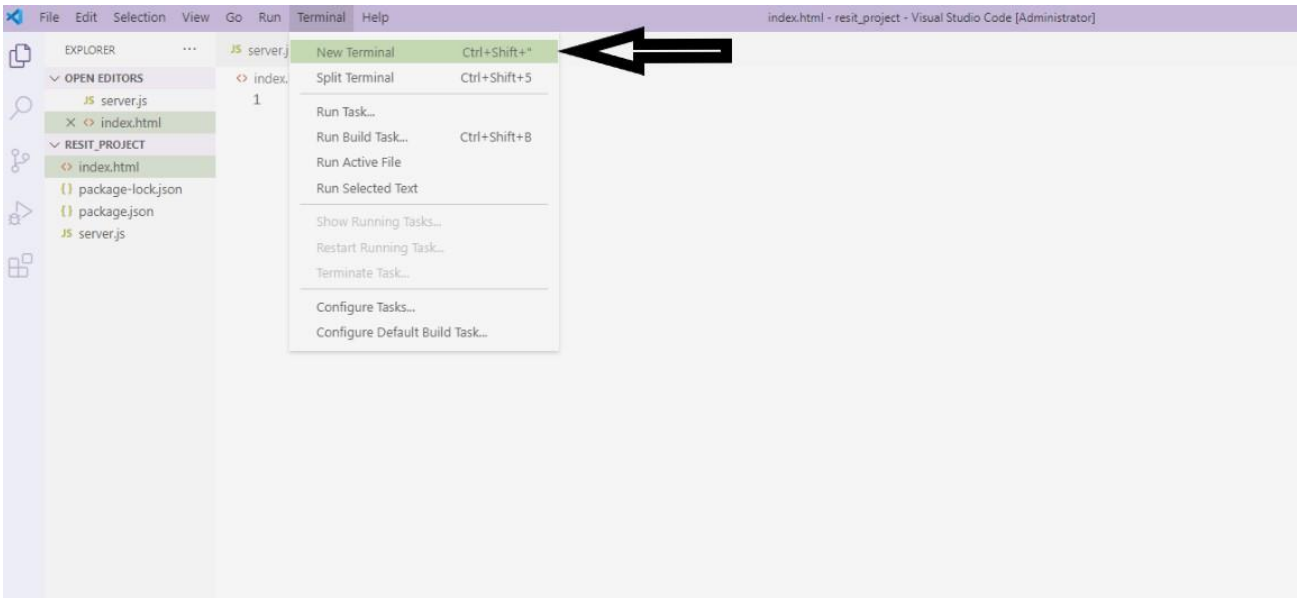
- **npm install express-handlebars**

This package is for user interface part. It Includes Handlebar Helper to manage sections in layouts.

• **npm install body-parser**

Node.js body parsing middleware.

- **Open terminal:** Terminal > New Terminal



- **Type your commands for installing:**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\hilal\ResitProject\resit_project> npm install express

added 50 packages, and audited 51 packages in 4s

found 0 vulnerabilities
PS C:\Users\hilal\ResitProject\resit_project>
```

```
PS C:\Users\hilal\ResitProject\resit_project> npm install mongoose

up to date, audited 82 packages in 2s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\hilal\ResitProject\resit_project>
```

```
PS C:\Users\hilal\ResitProject\resit_project> npm install express-handlebars

added 18 packages, and audited 100 packages in 2s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\hilal\ResitProject\resit_project>
```

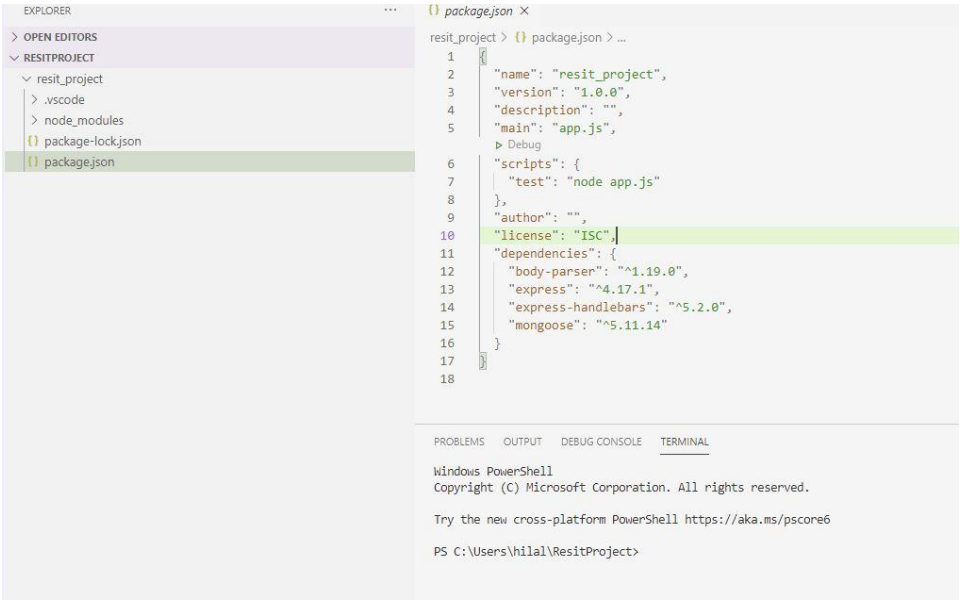
```
PS C:\Users\hilal\ResitProject\resit_project> npm install body-parser

up to date, audited 70 packages in 916ms

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\hilal\ResitProject\resit_project>
```

- **After installing,** check your package.json and node_modules folder if your packages installed.



Step 8: Coding

server.js

https://github.com/HilalEKC/resit_project/blob/main/server.js

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/inputTreeTable', { useNewUrlParser: true }, (err) => {
    if (!err) {
        console.log('DB connection is successful.')
    }
    else {
        console.log('Error in DB connection : ' + err)
    }
});
var inputTable = new mongoose.Schema({
    name: {
        type: String,
        required: 'This field is required.'
    },
    latitude: {
        type: Number,
        required: 'This field is required.'
    },
    longitude: {
        type: Number,
        required: 'This field is required.'
    },
    height: {
        type: Number,
        required: 'This field is required.'
    }
});
mongoose.model('InputTree', inputTable);
const express = require('express');
const path = require('path');
const exphbs = require('express-handlebars');
const bodyparser = require('body-parser');
var app = express();
app.use(bodyparser.urlencoded({
    extended: true
}));
app.use(bodyparser.json());
app.set('views', path.join(__dirname, '/views/'));
app.engine('hbs', exphbs({ extname: 'hbs', defaultLayout: 'mainLayout', layoutsDir: __dirname + '/views/body/' }));
app.set('view engine', 'hbs');
app.listen(5000, () => {
    console.log('Express server started at port : 5000');
});
const inputCtrl = require('./inputCtrl');
app.use('/tree', inputCtrl);
```

list.hbs

https://github.com/HilalEKC/resit_project/blob/main/views/tree/list.hbs

```
<h3 style="text-align: center;">
    All Data From Database
</h3>

<button style="float:left"><a class="btn" style="width:250px" href="/tree/list">Show All Data</a> </button>
<button style="float:right"><a class="btn" style="width:250px" href="/tree">Submit New Data</a> </button>
<table class="table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Latitude</th>
            <th>Longitude</th>
            <th>Tree Height</th>
        </tr>
    </thead>
    <tbody>
        {{#each list}}
        <tr>
            <td>{{this.name}}</td>
            <td>{{this.latitude}}</td>
            <td>{{this.longitude}}</td>
            <td>{{this.height}}</td>
            <td>
                <a href="/tree/{{this._id}}">Edit</a>
                <a href="/tree/delete/{{this._id}}" onclick="return confirm('Do you want to delete ?');">
                    Delete
                </a>

            </td>
        </tr>
        {{/each}}
    </tbody>
</table>

<button><a class="btn btn-block" style="width:250px" href="/tree/api/sortLatitude">Sort Ascending Latitude </a> </button>
<button style="float: right;"><a class="btn btn-block" style="width:250px" href="/tree/api/sortName">Sort Ascending  Name </a> </button>
<button><a class="btn btn-block" style="width:250px" href="/tree/api/sortLongitude">Sort Ascending Longitude </a> </button>
<button style="float: right;"><a class="btn btn-block" style="width:250px" href="/tree/api/sortHeight">Sort Ascending  Tree Height </a> </button>
<button><a class="btn btn-block" style="width:250px" href="/tree/api/filterHeightLt50">Filter Tree Height Less Than  50 </a> </button>
<button style="float: right;"><a class="btn btn-block" style="width:250px;" href="/tree/api/filterHeightGt50">Filter Tree Height Greater Than  50 </a> </button>
```

mainLayout.hbs

https://github.com/HilalEKC/resit_project/blob/main/views/body/mainLayout.hbs

```
<html>
<head>
    <title>Resit Project</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPk
</head>
<body>
    <div class="row">
        <div class="col-md-6 offset-md-3" style="background-color: #fff">
            {{{body}}}
        </div>
    </div>
</body>
</html>
```

addEdit.hbs

https://github.com/HilalEKC/resit_project/blob/main/views/tree/addEdit.hbs

```
<h3>{{viewTitle}}</h3>
<div class="form-group " style="float: right;">
        <button><a class="btn" href="/tree/list">Show All Data</a></button>
</div>
<form action="/tree" method="POST">

    <input type="hidden" name="_id" value="{{tree._id}}">
    <div class="form-group">
        <label>Name</label>
        <input type="text" class="form-control" name="name" placeholder="Name" value="{{tree.name}}">
        <div class="text-danger">
            {{tree.nameError}}</div>
    </div>
    <div class="form-group">
        <label>Latitude</label>
        <input type="text" class="form-control" name="latitude" placeholder="39.865649" value="{{tree.latitude}}">
        <div class="text-danger">
            {{tree.latitudeError}}</div>
    </div>

    <div class="form-group">
        <label>Longitude</label>
        <input type="text" class="form-control" name="longitude" placeholder="32.733520" value="{{tree.longitude}}">
        <div class="text-danger">
            {{tree.longitudeError}}</div>
    </div>
    <div class="form-group">
        <label>Tree Height</label>
        <input type="text" class="form-control" name="height" placeholder="4.20" value="{{tree.height}}">
        <div class="text-danger">
            {{tree.heightError}}</div>
    </div>

    <div class="form-group">
        <button type="submit">Submit</button>
    </div>

</form>
```

inputCtrl.js – 1

https://github.com/HilalEKC/resit_project/blob/main/inputCtrl.js

```
const express = require('express');
var router = express.Router();
const mongoose = require('mongoose');
const treeInput = mongoose.model('InputTree');

router.get('/', (req, res) => {
    res.render("tree/addEdit", {
        viewTitle: "Input Tree"
    });
});

router.post('/', (req, res) => {
    if (req.body._id == ''){
        insertRecord(req, res);
    }
    else{
        updateRecord(req, res);
    }

});

function insertRecord(req, res) {
    var tree = new treeInput();
    tree.name = req.body.name;
    tree.latitude = req.body.latitude;
    tree.longitude = req.body.longitude;
    tree.height = req.body.height;

    tree.save((err) => {
        if (!err)
            res.redirect('tree/list');
        else {
            if (err.name == 'ValidationError') {
                handleValidationError(err, req.body);
                res.render("tree/addEdit", {
                    viewTitle: "Input Tree",
                    tree: req.body
                });
            }
        }
    });
}
```

inputCtrl.js - 2

```
function updateRecord(req, res) {
    treeInput.findOneAndUpdate({ _id: req.body._id }, req.body, { new: true }, (err, doc) => {
        if (!err) {
            res.redirect('tree/list');
        }
        else {
            if (err.name == 'ValidationError') {
                handleValidationError(err, req.body);
                res.render("tree/addEdit", {
                    viewTitle: 'Update Input',
                    tree: req.body
                });
            }
        }
    });
}
router.get('/list', (req, res) => {
    treeInput.find((err, docs) => {
        res.render("tree/list", {
            list: docs
        });
    });
});
router.get('/api/data', (req, res) => {
    treeInput.find((err, docs) => {
        res.send(docs);
    });
});

router.get('/api/sortLatitude', (req, res) => {

    treeInput.find({}).sort({latitude: 1}).exec(function(err, docs){
        res.render("tree/list", {
            list: docs
        });
    })
})
router.get('/api/sortName', (req, res) => {

    treeInput.find({}).sort({name: 1}).exec(function(err, docs){
        res.render("tree/list", {
            list: docs
        });
    })
});
```

inputCtrl.js - 3

```
router.get('/api/sortLongitude', (req, res) => {

    treeInput.find({}).sort({longitude: 1}).exec(function(err, docs){
        res.render("tree/list", {
            list: docs
        });
    })
});
router.get('/api/sortHeight', (req, res) => {

    treeInput.find({}).sort({height: 1}).exec(function(err, docs){
        res.render("tree/list", {
            list: docs
        });
    })
});
router.get('/api/filterHeightLt50', (req, res) => {

    treeInput.find({'height' : {$lt: 50}}).exec(function(err, docs){
        res.render("tree/list", {
            list: docs
        });
    })
});
router.get('/api/filterHeightGt50', (req, res) => {

    treeInput.find({'height' : {$gt: 50}}).exec(function(err, docs){
        res.render("tree/list", {
            list: docs
        });
    })
});
```

inputCtrl.js - 4

```
function handleValidationError(err, body) {
    for (field in err.errors) {
        switch (err.errors[field].path) {
            case 'name':
                body['nameError'] = err.errors[field].message;
                break;
            case 'latitude':
                body['latitudeError'] = err.errors[field].message;
                break;
            case 'longitude':
                body['longitudeError'] = err.errors[field].message;
                break;
            case 'height':
                body['heightError'] = err.errors[field].message;
                break;
            default:
                break;
        }
    }
}
router.get('/:id', (req, res) => {
    treeInput.findById(req.params.id, (err, doc) => {
            res.render("tree/addEdit", {
                viewTitle: "Update Input",
                tree: doc
            });
    });
});
router.get('/delete/:id', (req, res) => {
    treeInput.findByIdAndRemove(req.params.id, (err, doc) => {
        res.redirect('/tree/list');
    });
});
module.exports = router;
```

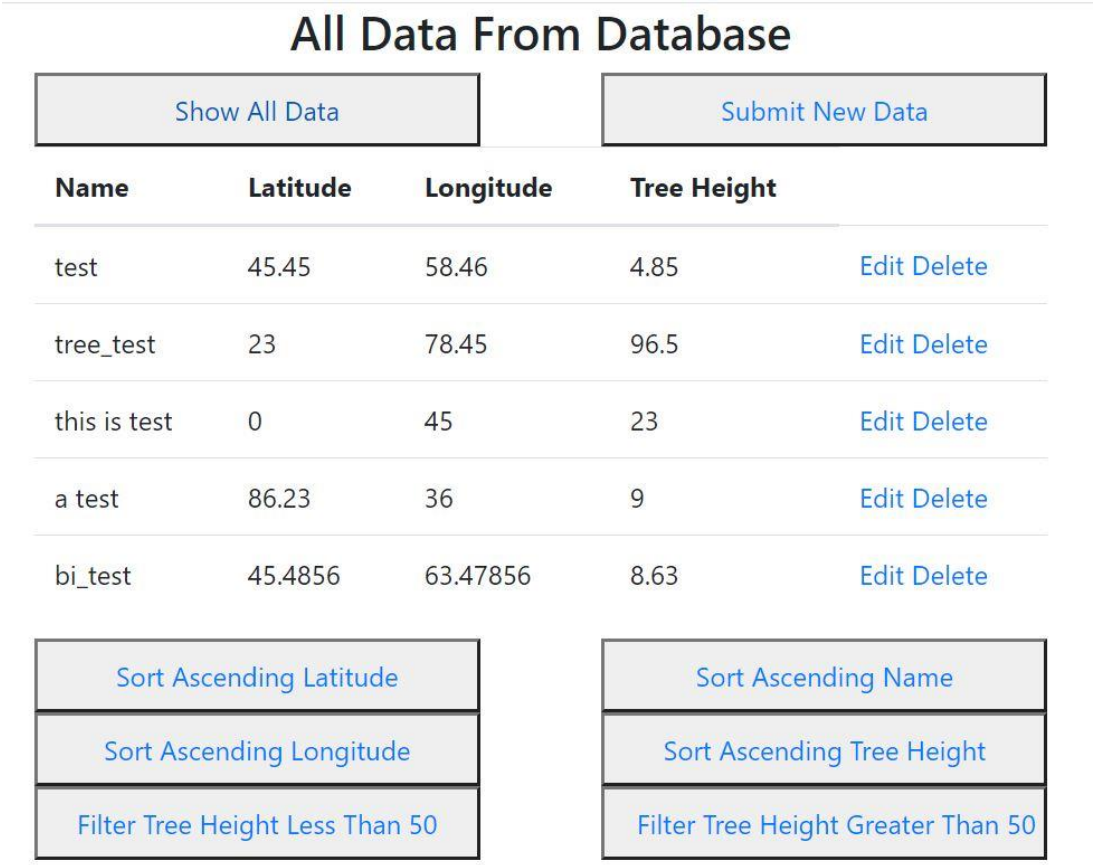
Step 9: Testing

We can test our project with VS Code terminal, browser and MongoDB Compass.

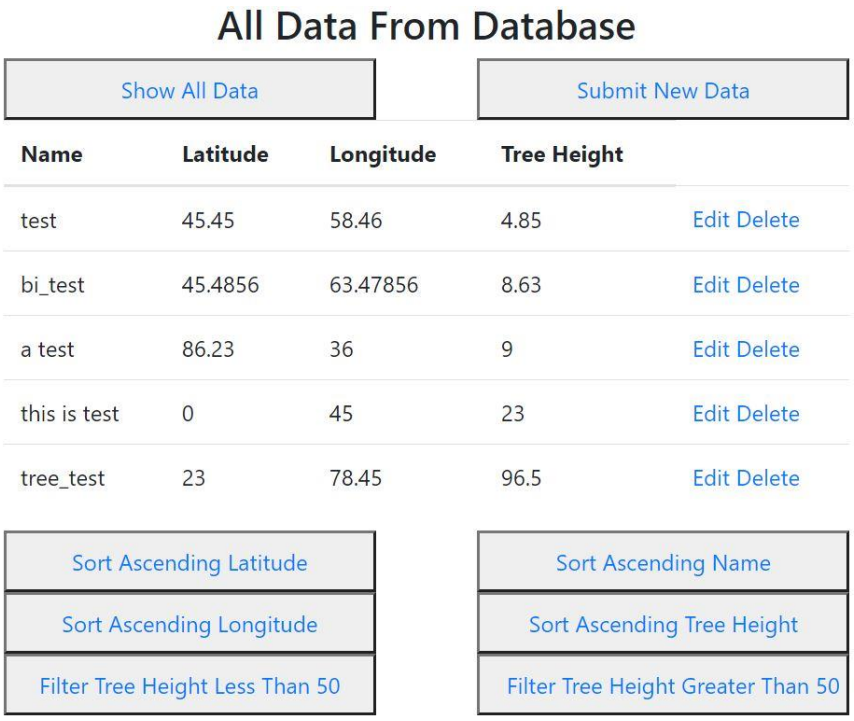
- **Type** "**node server.js**" to VS Code terminal
- **Open** MongoDB Compass
- **Go** to http://localhost:5000/tree

1. First page: submit input

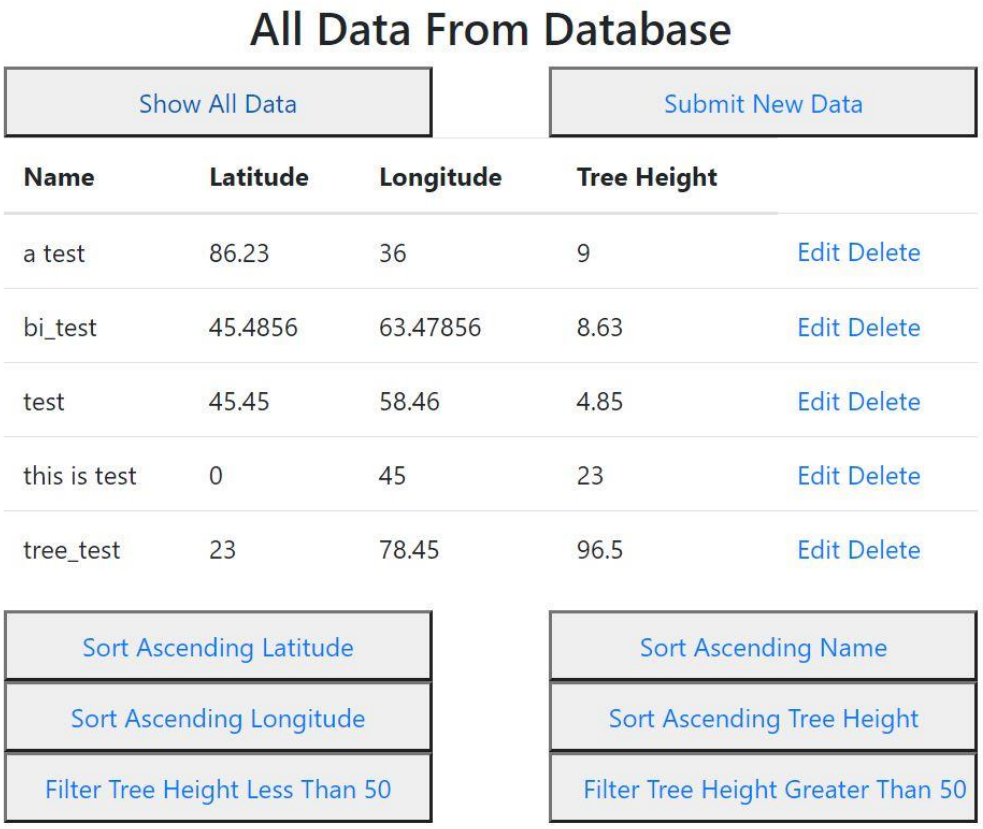
2. After submit button clicked



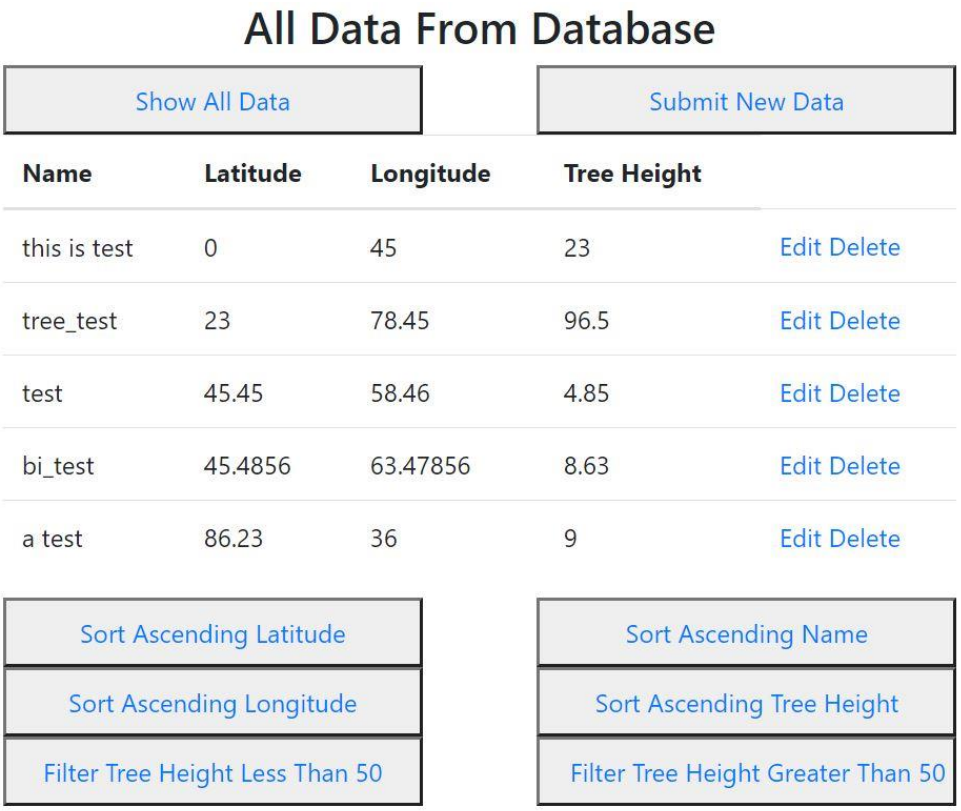
3. Sort ascending order by tree height



4. Sort ascending order by name



5. Sort ascending order by latitude



All Data From Database

Show All Data | Submit New Data

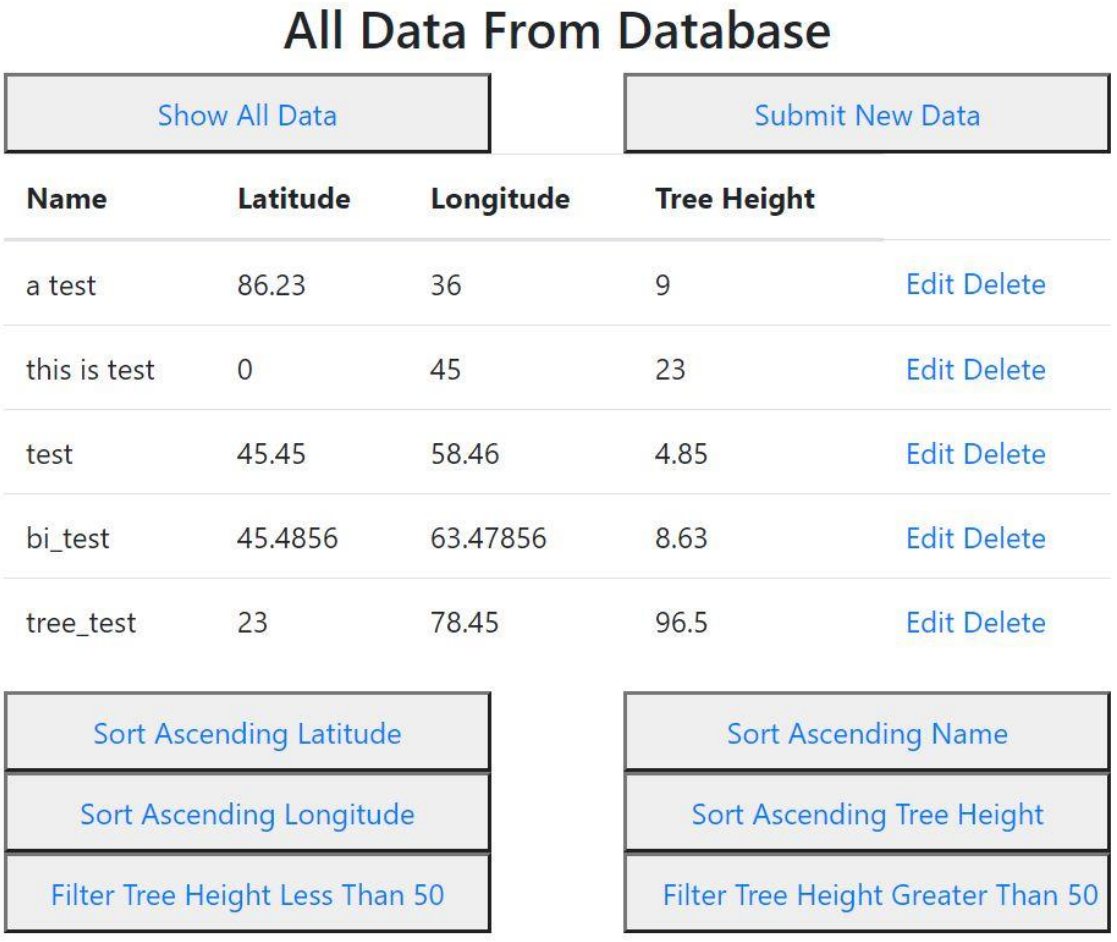
Name	Latitude	Longitude	Tree Height	
this is test	0	45	23	Edit Delete
tree_test	23	78.45	96.5	Edit Delete
test	45.45	58.46	4.85	Edit Delete
bi_test	45.4856	63.47856	8.63	Edit Delete
a test	86.23	36	9	Edit Delete

Sort Ascending Latitude | Sort Ascending Name

Sort Ascending Longitude | Sort Ascending Tree Height

Filter Tree Height Less Than 50 | Filter Tree Height Greater Than 50

6. Sort ascending order by longitude



All Data From Database

Show All Data | Submit New Data

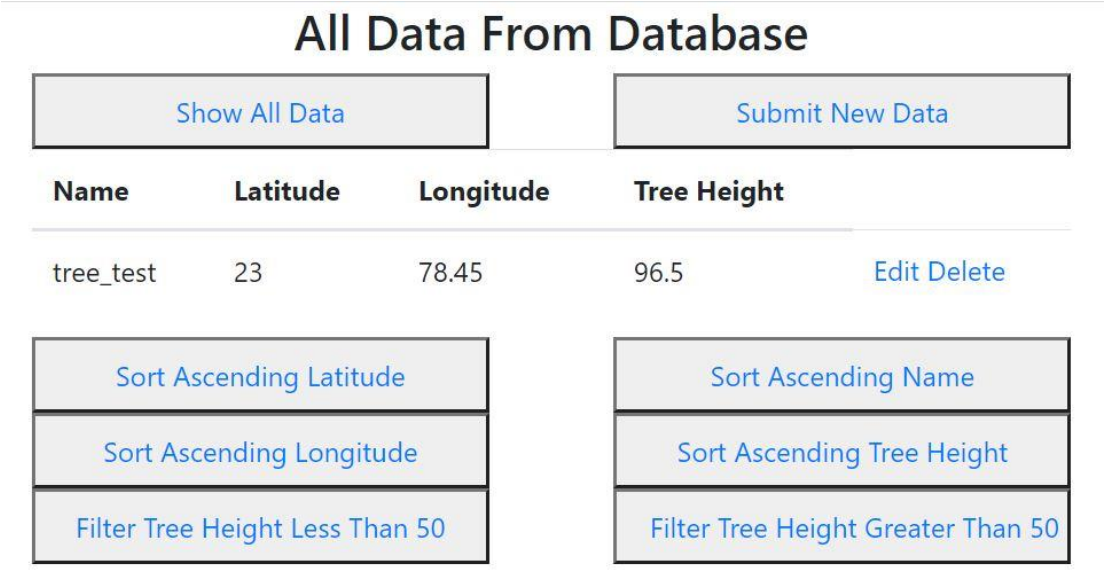
Name	Latitude	Longitude	Tree Height	
a test	86.23	36	9	Edit Delete
this is test	0	45	23	Edit Delete
test	45.45	58.46	4.85	Edit Delete
bi_test	45.4856	63.47856	8.63	Edit Delete
tree_test	23	78.45	96.5	Edit Delete

Sort Ascending Latitude | Sort Ascending Name

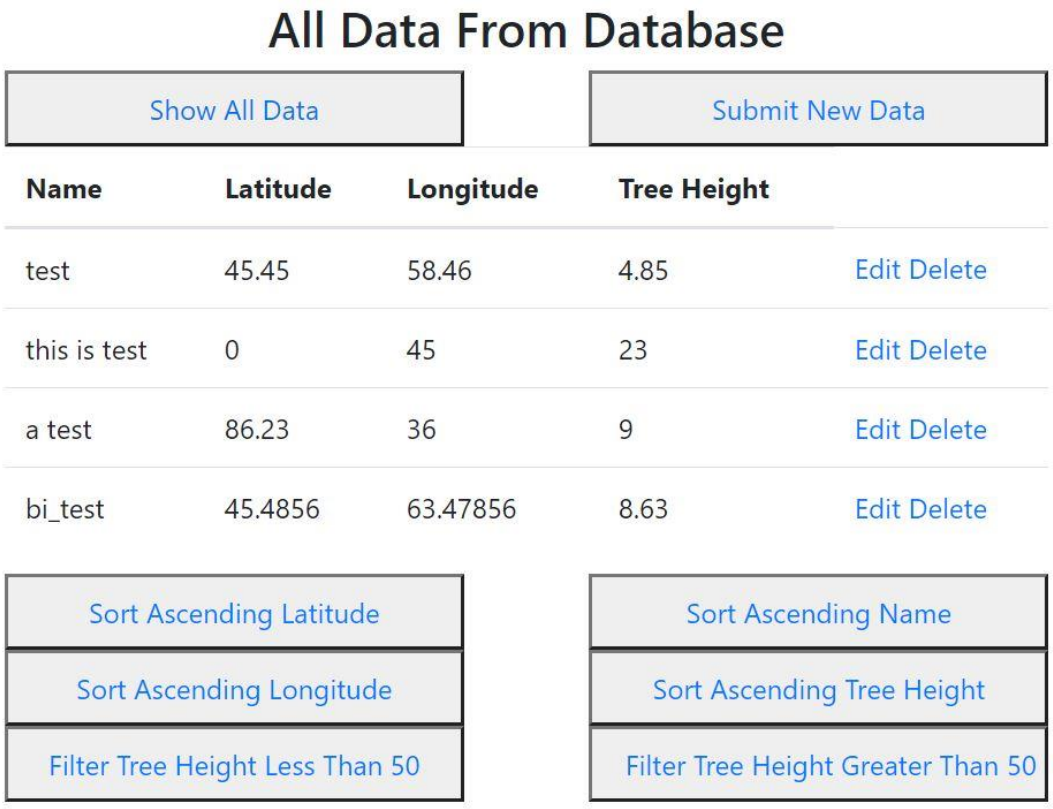
Sort Ascending Longitude | Sort Ascending Tree Height

Filter Tree Height Less Than 50 | Filter Tree Height Greater Than 50

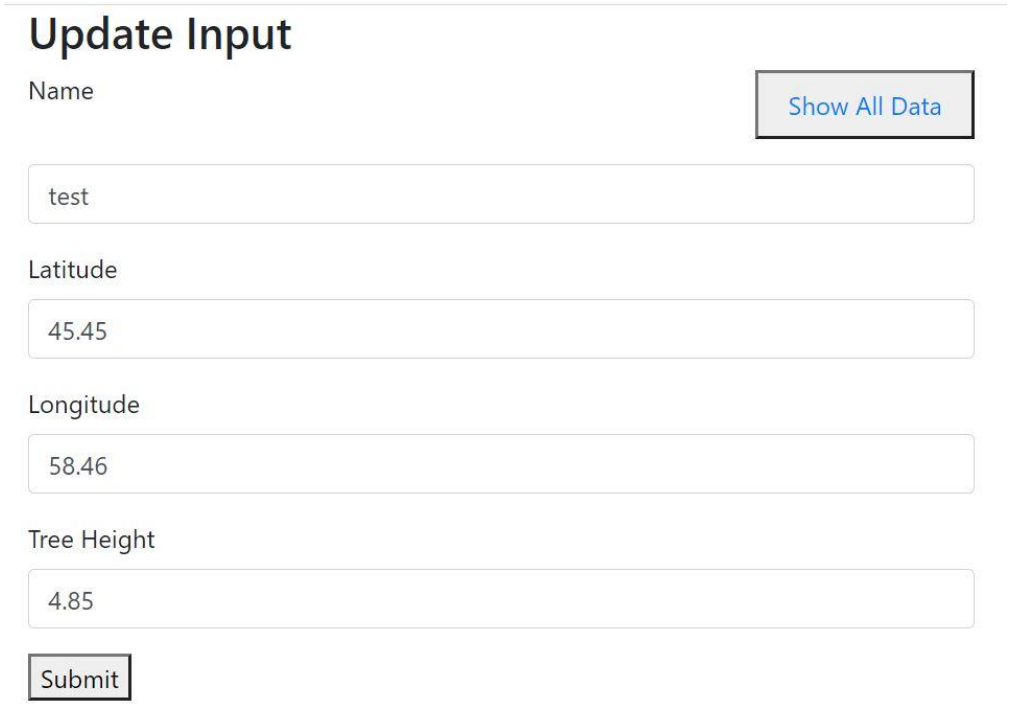
7. Filter tree height greater than 50



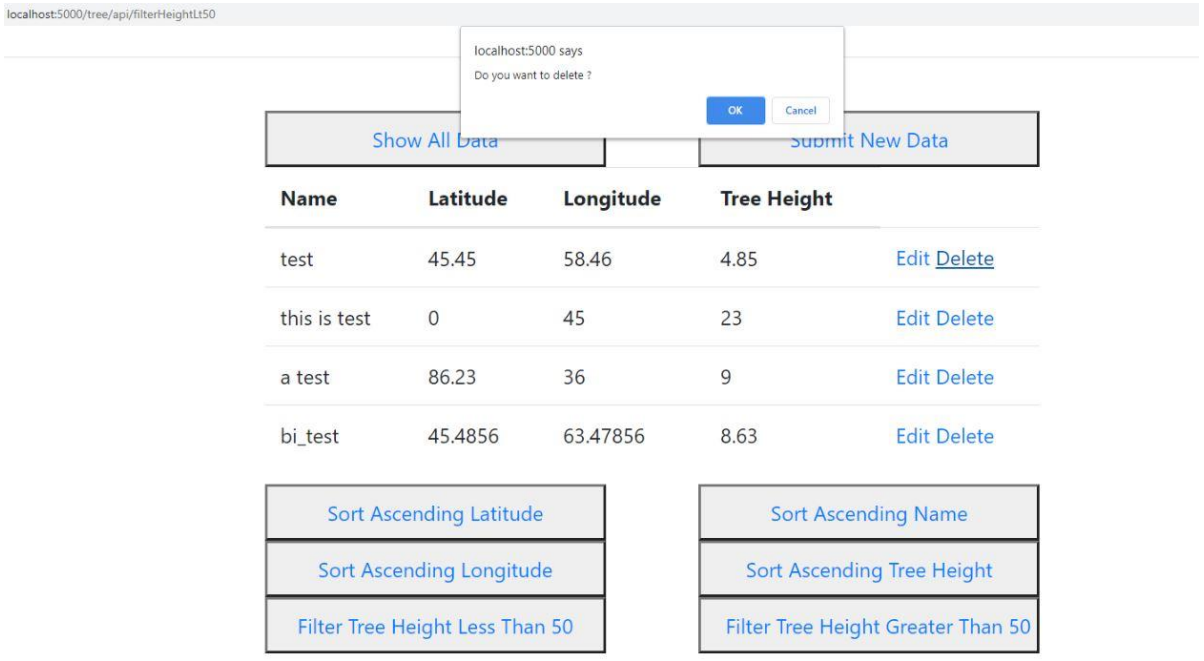
8. Filter tree height less than 50



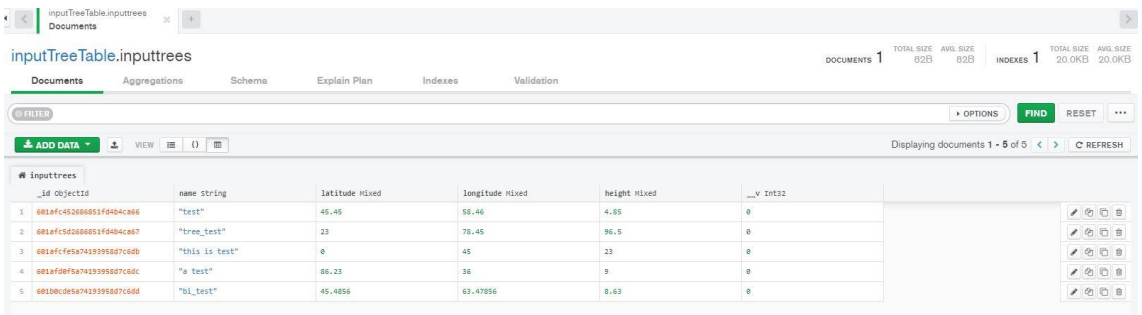
9. Edit data



10. Delete data



11. Database



Notes

- And we can view all data in JSON format using API:
  http://localhost:5000/tree/api/data


```
[{"_id":"601afc5d2686851fd4b4ca67","name":"tree_test","latitude":23,"longitude":78.45,"height":96.5,"__v":0},
{"_id":"601afcfe5a74193958d7c6db","name":"this is test","latitude":0,"longitude":45,"height":23,"__v":0},
{"_id":"601afd0f5a74193958d7c6dc","name":"a test","latitude":86.23,"longitude":36,"height":9,"__v":0},
{"_id":"601b0cde5a74193958d7c6dd","name":"bi_test","latitude":45.4856,"longitude":63.47856,"height":8.63,"__v":0
}]
```


- Here are the all urls:
  - http://localhost:5000/tree/
  - http://localhost:5000/tree/list
  - http://localhost:5000/tree/api/sortLatitude
  - http://localhost:5000/tree/api/sortName
  - http://localhost:5000/tree/api/sortLongitude
  - http://localhost:5000/tree/api/sortHeight
  - http://localhost:5000/tree/api/filterHeightLt50
  - http://localhost:5000/tree/api/filterHeightGt50
  -
- All steps are done. **Thank you**.

END OF FILE