# Setup:

C:\data\db || npm init –yes || npm I mongoose || npm install - - save mongoose

# DB Connection:

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/playground')

.then(() => console.log("yes connected mongo db.."))

.catch(err => console.log('not connected..', err));

# Schema Creation:

const courseschema = new mongoose.Schema({

name: String,

author: String,

tags: [String],

date: {type: Date, default: Date.now},

isPublished: Boolean });

# Saving in DB:

async function createCourse(){

const Course = mongoose.model('Course', courseschema);

const course = new Course({

name: 'Node.js Course',

author: 'Mosh',

tags: ['node', 'backend'],

isPublished: false });

const result = await course.save();

console.log(result); }

createCourse();

# Retrieving data From DB:

async function getCourses(){

const courses = await Course

.find({ author: ‘Mosh’, isPublished: true})

.limit(10)

.sort({name: 1})

.select({name: 1, tags: 1});

console.log(courses);}

getCourses();

Note: 1 for Ascending & -1 for Descending;

# Comparison Operators:

eq (equal) || ne (not equal) || gt (greater than) || gte (greater than or equal to)

lt (less than) || lte (less than or equal to) || in || nin (not in)

async function getCourses(){

const courses = await Course

.find({price:{$gte:10, $lte: 20}})

//.find({price:{$in:[10, 20, 15]};

Console.log(courses);

}

async function getCourses(){

const courses = await Course

.find()

.or([{author: ‘Mosh’}, {isPublished:true}])

.and([{author: ‘Mosh’}, {isPublished:true}])

Console.log(courses);

}

const express = require('express'); const router = express.Router();

const mongoose = require('mongoose'); const User = require('./user');

# Connection Establish:

const db = 'mongodb://localhost:27017/sampleDB';

mongoose.connect(db, err => {

if(err) {

console.error('Error!' + err);

} else {

console.log('Connected to mongoDB');

}

})

**Get:**

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

User.find({}, (err, users) => {

if(err) res.send(err);

res.send(users);

})

})

**Get/{id}:**

router.get('/register/:id', (req, res) => {

res.send(req.params);

})

**Put:**

router.put('/register', (req, res) => {

let userdata = req.body;

User.findOneAndUpdate({email: userdata.email}, {email: userdata.email, password:userdata.password}, (error) => {

if(error){

console.log(error);

}

res.send('success..');

})

})

User.set({

isPublished: true,

author: ‘Another Author’

})

**Post:**

router.post('/register', (req, res) => {

let userData = req.body;

let user = new User(userData);

user.save((error, registerdUser) => {

if(error){

console.log(error);

}else{

res.status(200).send(registerdUser);

}

})

})

**Delete:**

router.delete('/dele', (req, res) => {

let userData = req.body;

User.findOneAndDelete({email:userData.email}, (error, user) => {

if(error){

console.log(error);

}else{

if(!user){

res.status(404).send('Not Found Email ID..');

}else{

res.status(200).send(userData);

}

}

})

})

module.exports = router;

User.findOne({email: userData.email}, (err, user) => {

if(error){

console.log(error);

}else{

if(!user){

res.status(404).send('Not Found Email ID..');

}else{

res.status(200).send(userData);

}

}

}

**Schema-Creation:**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

let colegueSchema = new Schema({

name: String,

organisation: String,

location: String

});

module.exports = mongoose.model('colegue', colegueSchema);

const courseSchema = new mongoose.Schema({

**name**: {

type: String,

required: true,

minlength: 5,

maxlength: 255

}

**category**: {

type: String, required: true,

enum: [‘web’, ‘mobile’, ‘network’],

lowercase: true, // uppercase: true

trim: true

}

Date: {type: Date, default: Date.now},

price: {

type: Number,

min: 10, max: 200,

get: v => Math.round(v),

set: v => Math.round(v)

}

})