I-Notebook

1. Create an app

PS D:\Workspace\React> npx create-react-app inotebook

1. Create a folder named backend for node.js
2. Open terminal
3. Fire command and answer the questions

$npm init

1. Now install packages like express, mongoose and nodemon

$npm i express

$npm i mongoose

$npm i nodemon

1. Now creating models for user and notes
2. Create folder models
3. Create user model

//user.js

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const userSchema = new Schema({

    name: {

        type: String,

        required: true,

    },

    email: {

        type: String,

        required: true,

        unique: true,

    },

    password: {

        type: String,

        require: true,

    },

    time: {

        type: Date,

        default: Date.now

    },

});

module.exports = mongoose.model('user', userSchema);

1. Create notes model

//notes.js

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const notesSchema = new Schema({

    title: {

        type: String,

        required: true,

    },

    desciption: {

        type: String,

        required: true,

    },

    tag: {

        type: String,

        default: "general",

    },

    time: {

        type: Date,

        default: Date.now

    },

});

module.exports = mongoose.model('notes', notesSchema);

1. Now create a mongo connection

//db.js

const mongoose = require('mongoose');

const url = "mongodb://localhost:27017/"; // copy url from mongo compas

const connectToMongoose = () => {

    mongoose.connect(url).then(() => {

        console.log("connected to mangoose successfuly!");

    }).catch((err) => console.log(err));

}

module.exports = connectToMongoose;

1. Create index.js

//index.js

const express = require('express');

const connectToMongoose = require('./db');

const app = express();

const port = 3000;

connectToMongoose();

app.get('/', () => {

    console.log("hello");

})

app.listen(port, () => {

    console.log(`app is listening at port http://localhost:${port}`)

});

1. Now add a script to run file index.js edit package.json

"scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "nodemon index.js" //this is means ‘npm start’ to run project

  },

1. Now lets create a routers for authentication and notes
2. Routers/notes.js

//notes.js

const express = require('express');

const router = express.Router();

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

    res.send([]);

});

module.exports = router;

1. Routers/auth.js

//auth.js

const express = require('express');

const router = express.Router();

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

    obj = {

        name: 'suraj',

        password: 'khot'

    }

    res.send(obj);

});

module.exports = router;

1. Now add routers to a=index.js files

//index.js

const express = require('express');

const connectToMongoose = require('./db');

const app = express();

const port = 3000;

connectToMongoose();

// routers

app.use('/api/auth',require('./routers/auth'));

app.use('/api/notes',require('./routers/notes'));

app.listen(port, () => {

    console.log(`app is listning at port http://localhost:${port}`)

});

1. now start server

PS D:\Workspace\React\inotebook\Backend> npm start

> inotebook-backend@1.0.0 start

> nodemon index.js

[nodemon] 3.1.4

[nodemon] to restart at any time, enter `rs`

[nodemon] watching path(s): \*.\*

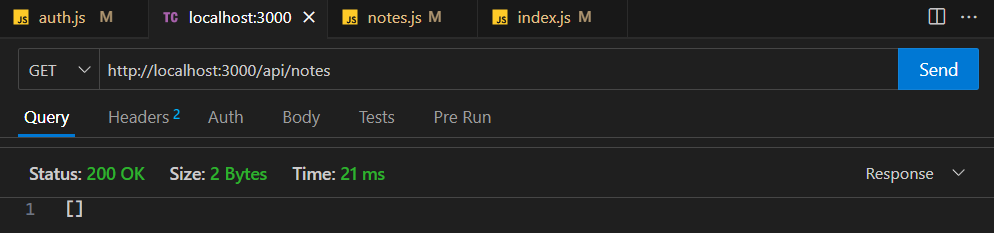
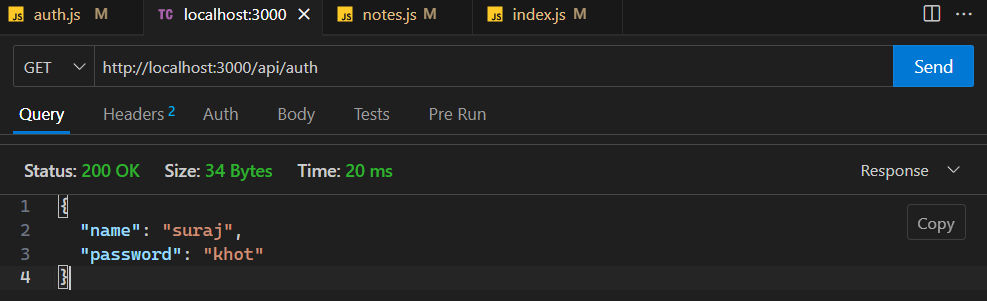
[nodemon] watching extensions: js,mjs,cjs,json

[nodemon] starting `node index.js`

app is listning at port http://localhost:3000

connected to mangoose successfuly!

Now check by using thunder client



1. now creating a router for accepting user using user schema(name,email,password)
2. auth.js

//auth.js

const express = require('express');

const router = express.Router();

const User = require('../models/User'); //import user schema

//sending post request and save user to db

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

//accepting request

    const request = req.body;

    const user = User(request);

    user.save();

    res.send(request);

});

module.exports = router;

1. now for that we require middleware instead we add one middleware in index.js

//index.js

//code

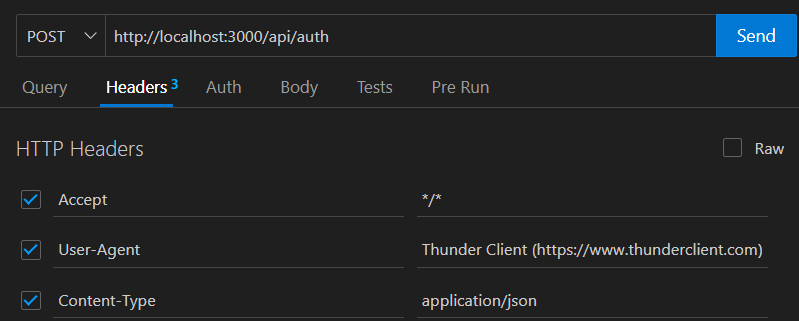
// middleware for accept json

app.use(express.json());

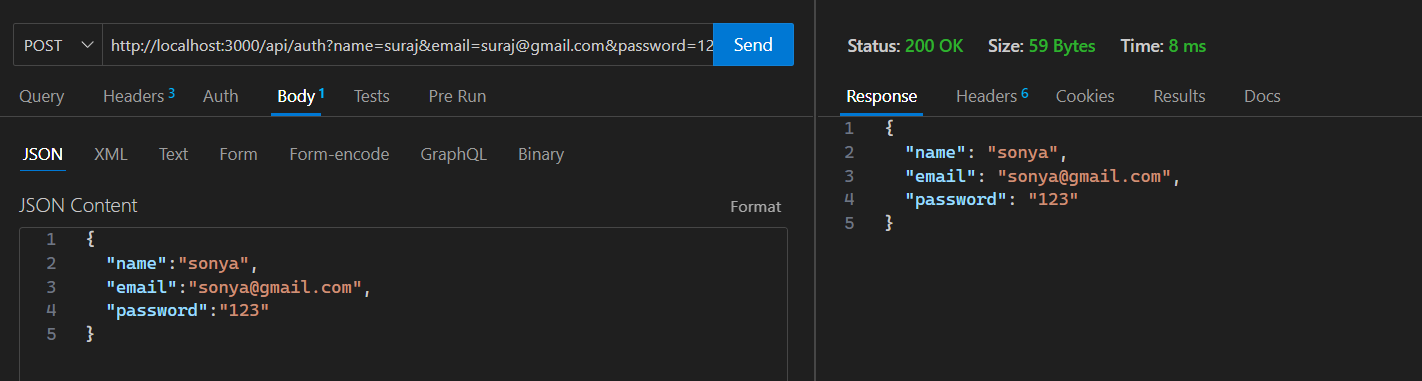
//code

1. now sending req through thunder client

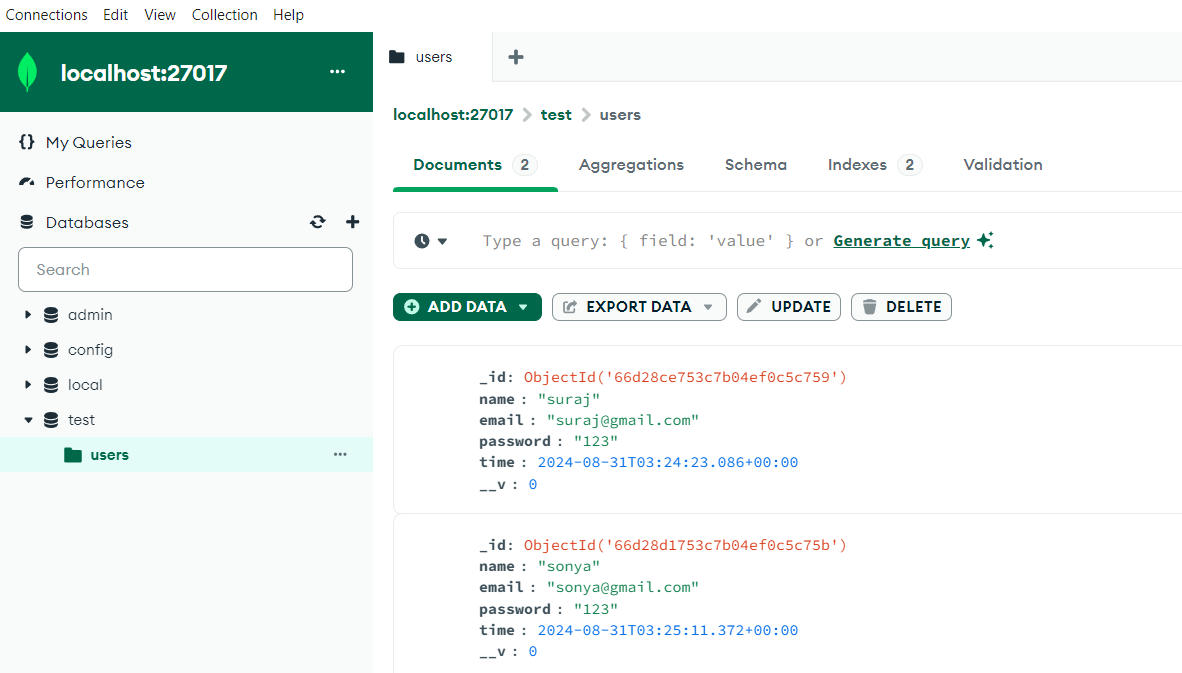
for that define content type as application/json in body and send



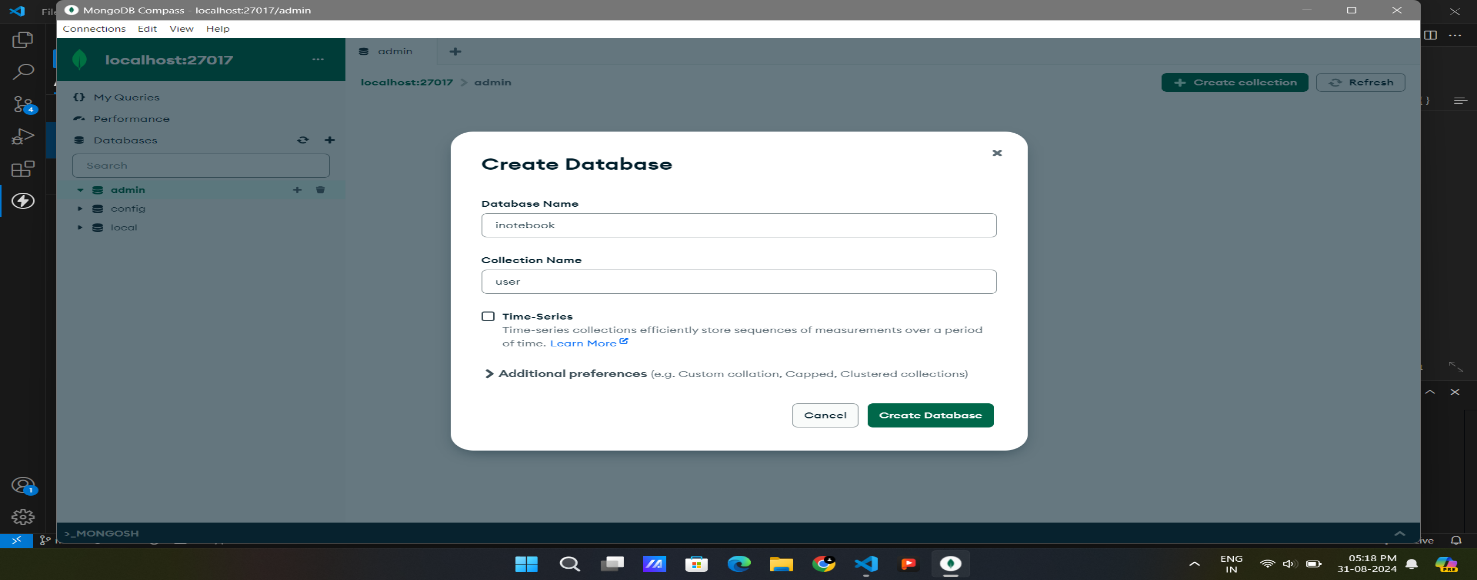
1. now send data



1. now see in mongoodb



1. now creating custom db in mongo and adding validations also saving user
2. create db in mongo



1. now change url in db.js

//db.js

const url = "mongodb://localhost:27017/inotebook";

//code

1. first thing is we do not want duplicate values(email)so that change our user schema logic

createIndexes : it will add indexes so that.. unique

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const userSchema = new Schema({

//code

});

const User = mongoose.model('user', userSchema);

User.createIndexes();

module.exports = User;

1. now use validator of express

url = <https://express-validator.github.io/docs/>

install it PS D:\Workspace\React\inotebook\Backend> npm install express-validator

const express = require('express');

const router = express.Router();

const User = require('../models/User');

const { validationResult, body } = require('express-validator');

router.post(

    '/',

    [

body('name', 'Enter a valid name').isLength({ min: 2 }),

        body('password', 'password must be at least 5 char').isLength({ min: 5 }),

        body('email', 'enter valid email').isEmail(),

    ],

    (req, res) => {

        const errors = validationResult(req);

        if (!errors.isEmpty()) {

            return res.status(400).json({ errors: errors.array() })

        }

        User.create({

            name: req.body.name,

            password: req.body.password,

            email: req.body.email

        }).then(user => res.json(user)).catch(err => res.json({ error: 'This email is already exists in database' }))

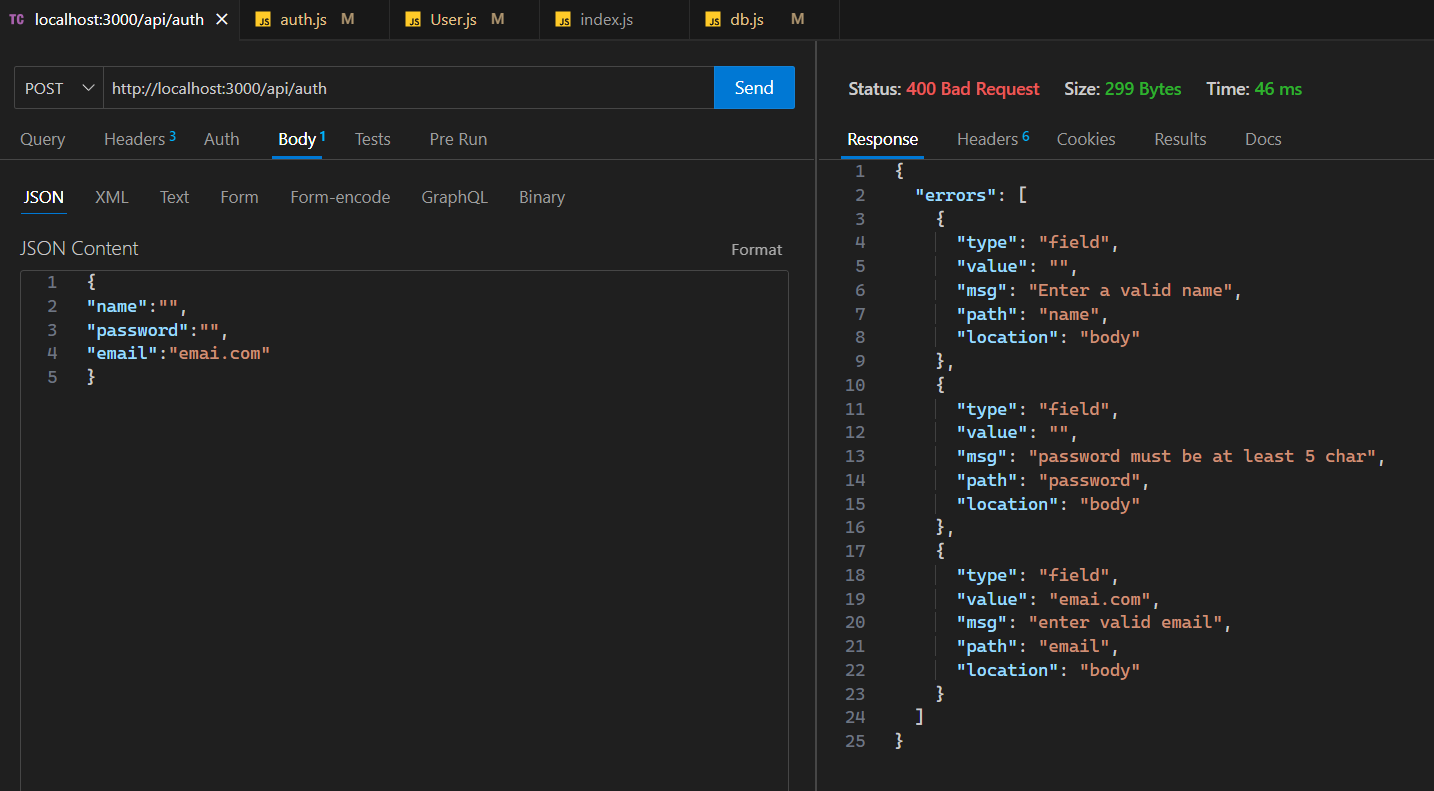
    });

module.exports = router;

/////

body('name', 'Enter a valid name').isLength({ min: 2 }), is like

body(‘key\_name’, ‘error value to display’).fn



///

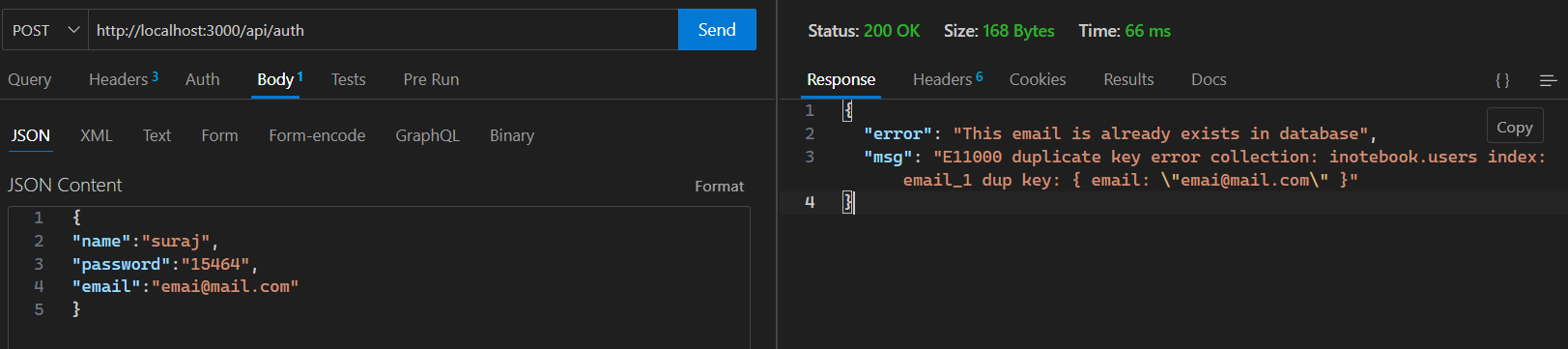
  const errors = validationResult(req);

        if (!errors.isEmpty()) {

            return res.status(400).json({ errors: errors.array() })

        }

This lines are every time same in all pgm



1. output like

