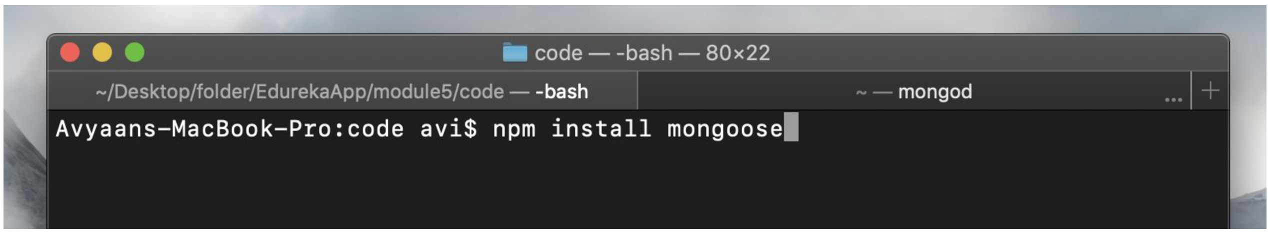
Module 5: REST APIs And GraphQL

Demo Document 2

edureka!

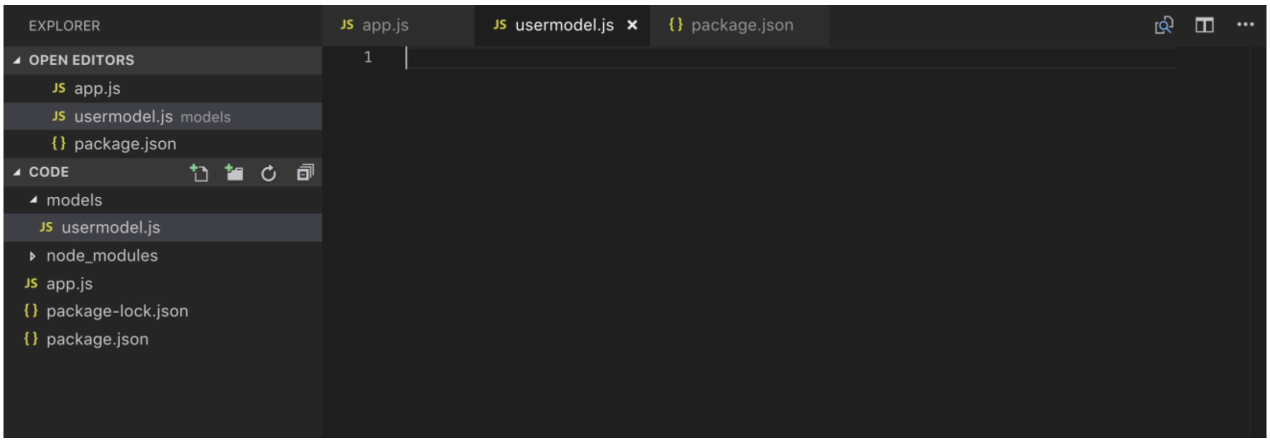
Working of mongoose API

Step 1: In continuation with the demo 1. We will integrate mongoose to it, to maintain the schema. For that let us, install mongoose locally in folder using ‘npm install mongoose’

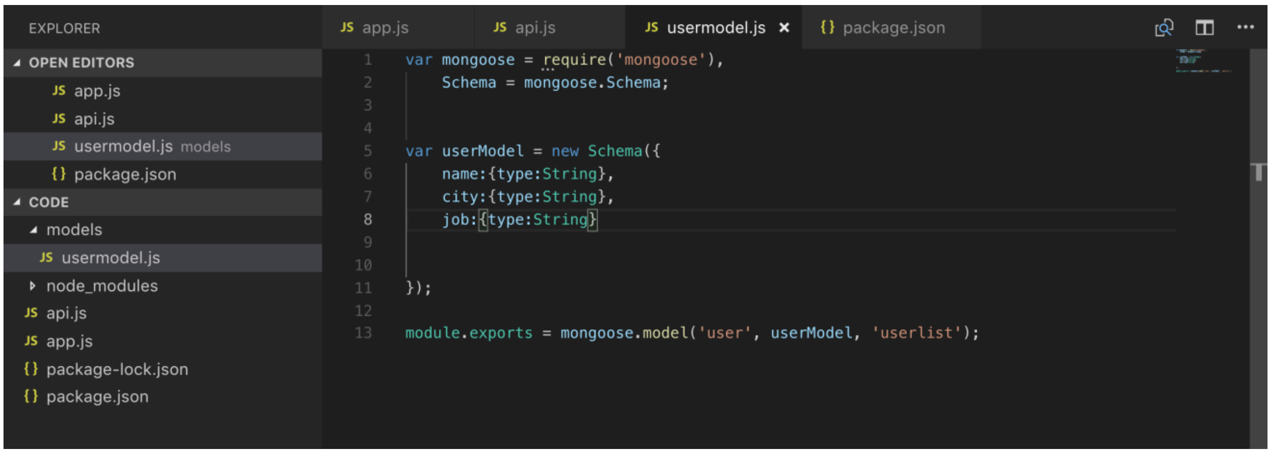


```
Avyaans-MacBook-Pro:code avi$ npm install mongoose
```

Step 2: First we must create one model file to design the schema. We have done it by creating a folder by the name of models, under which we have a file named usermodel.js.

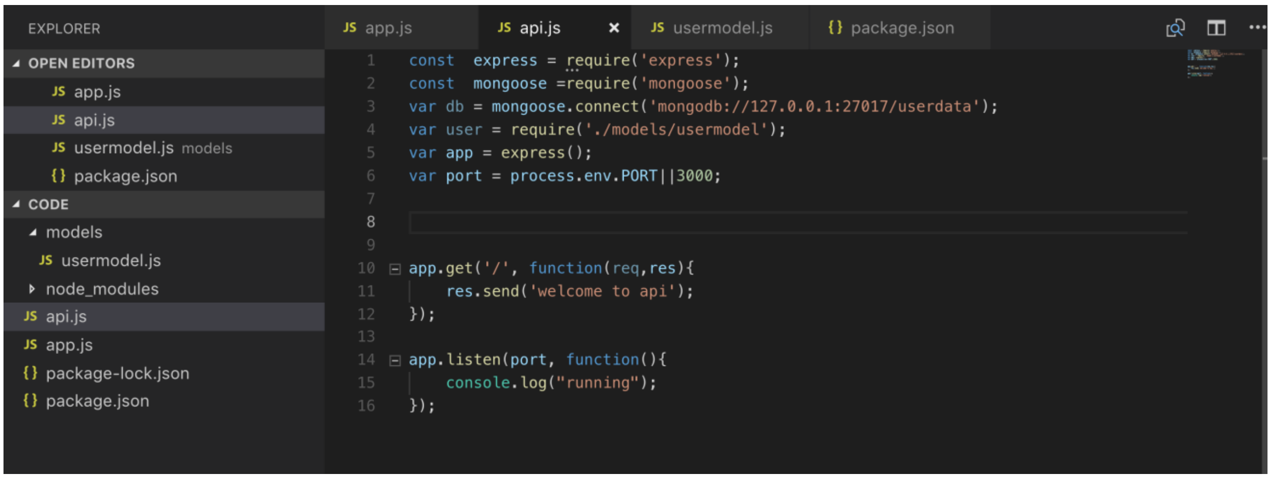


Step 3: In model file, define the schema with their data type and exports it. So that the APIs can consume it



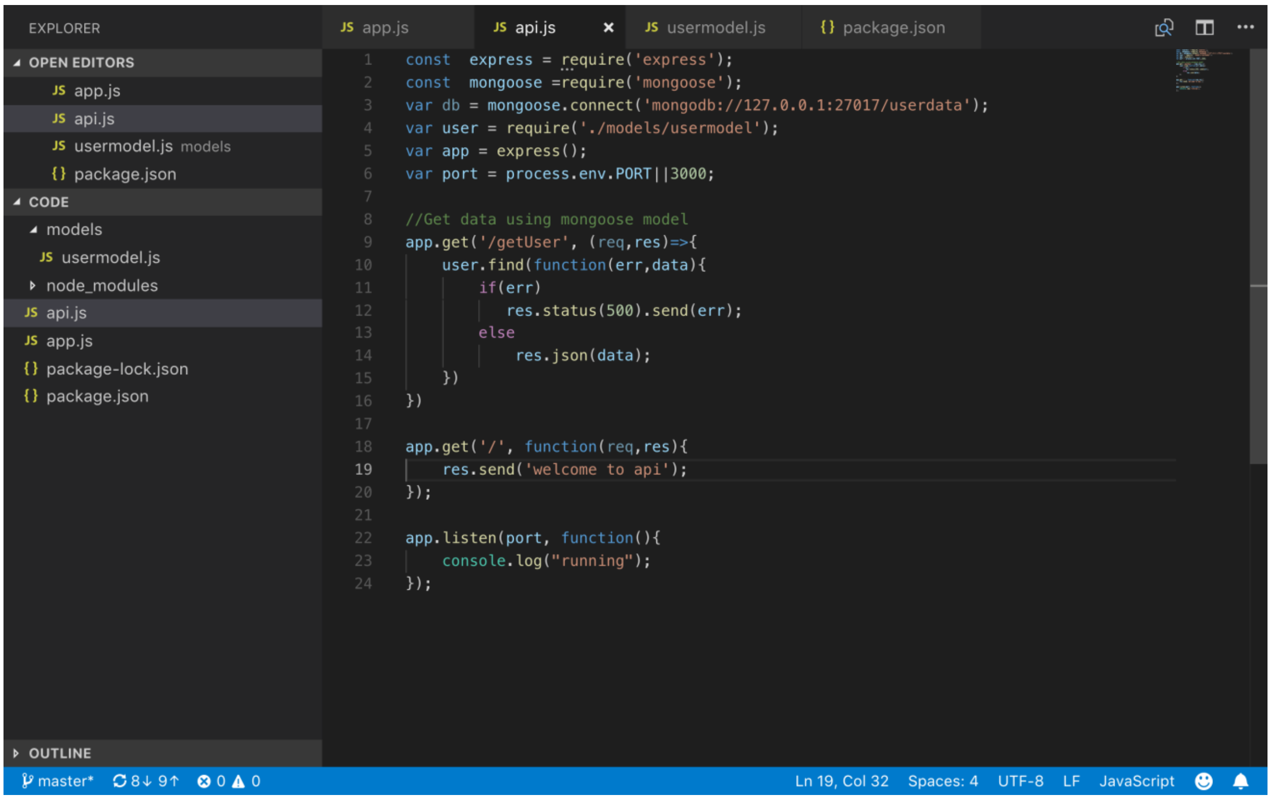
```
var mongoose = require('mongoose'),
    Schema = mongoose.Schema;


var userModel = new Schema({
    name:{type:String},
    city:{type:String},
    job:{type:String}


});

module.exports = mongoose.model('user', userModel, 'userlist');
```

Step 4: Introduce mongoose in the api file and connect it to the database using mongoose.



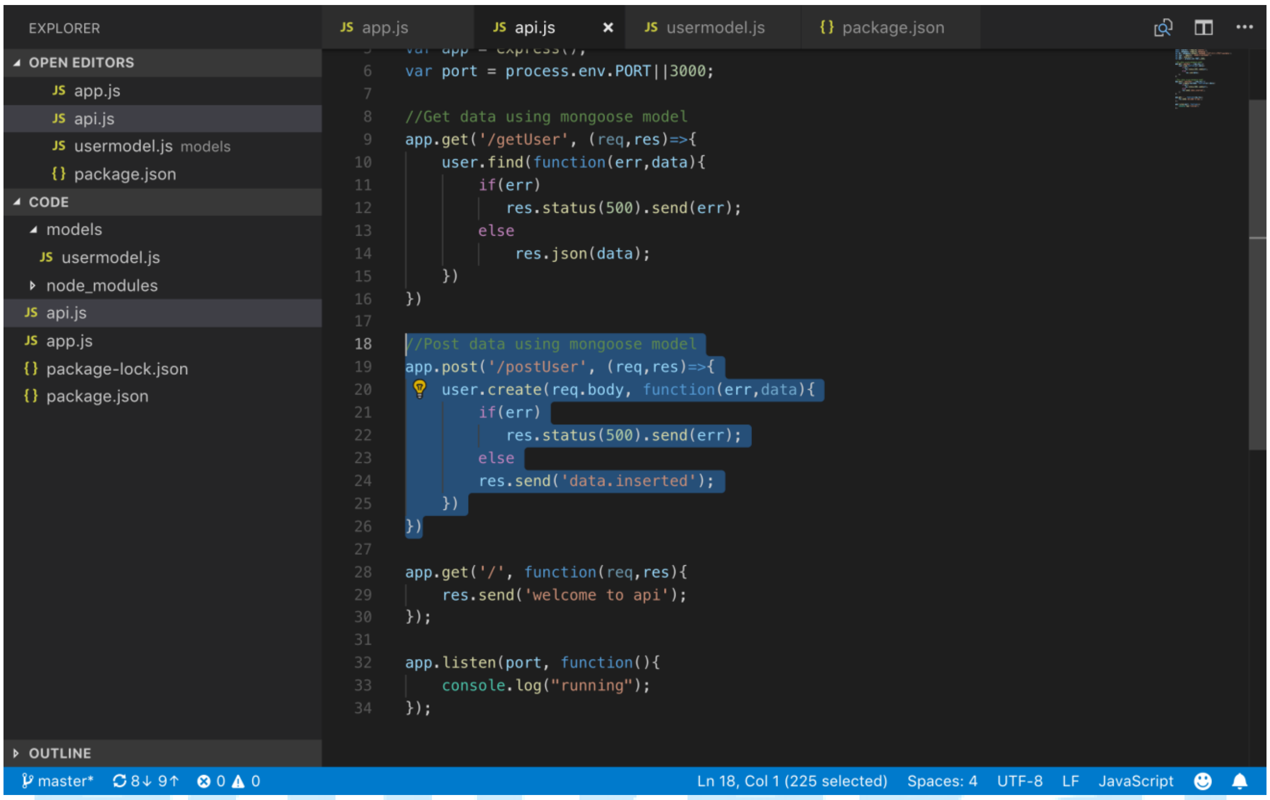
```
const  express = require('express');
const  mongoose =require('mongoose');
var db = mongoose.connect('mongodb://127.0.0.1:27017/userdata');
var user = require('./models/usermodel');
var app = express();
var port = process.env.PORT||3000;


app.get('/', function(req,res){
    res.send('welcome to api');
});

app.listen(port, function(){
    console.log("running");
});
```

Step 5: GET request will be same as before but here we don’t need to specify the database name or connection string as its maintained by Mongoose



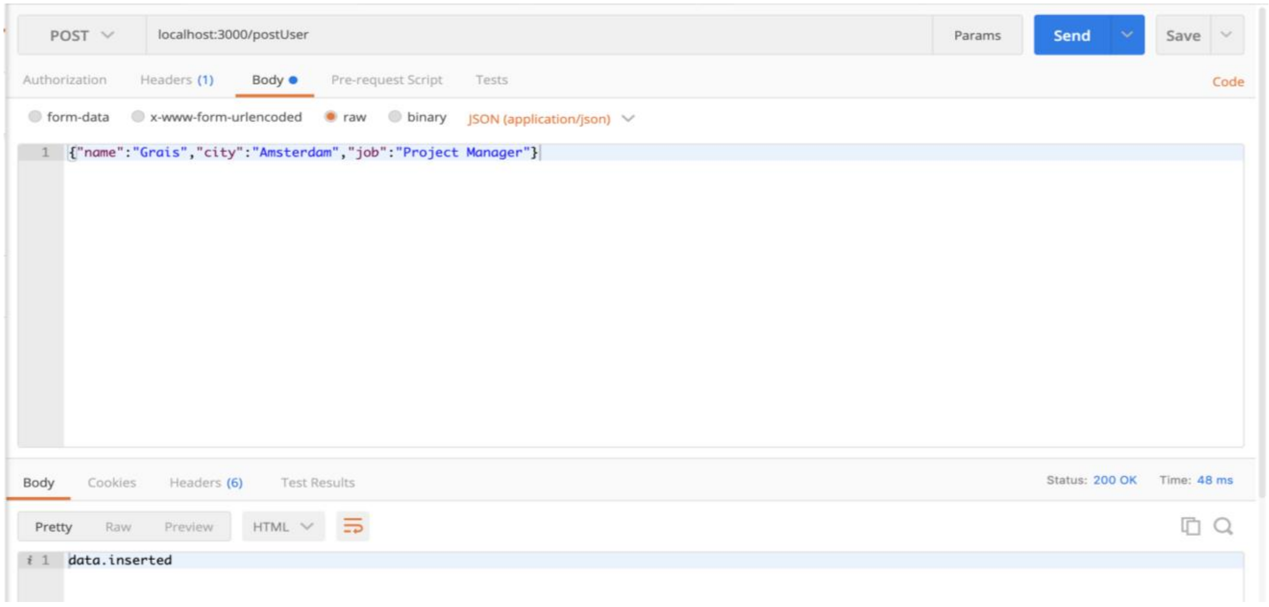
```
const  express = require('express');
const  mongoose =require('mongoose');
var db = mongoose.connect('mongodb://127.0.0.1:27017/userdata');
var user = require('./models/usermodel');
var app = express();
var port = process.env.PORT||3000;

//Get data using mongoose model
app.get('/getUser', (req,res)=>{
    user.find(function(err,data){
        if(err)
            res.status(500).send(err);
        else
            res.json(data);
    })
})

app.get('/', function(req,res){
    res.send('welcome to api');
});

app.listen(port, function(){
    console.log("running");
});
```

Step 6: For POST call, write the codes to insert the data



```
var port = process.env.PORT||3000;

//Get data using mongoose model
app.get('/getUser', (req,res)=>{
    user.find(function(err,data){
        if(err)
            res.status(500).send(err);
        else
            res.json(data);
    })
})

//Post data using mongoose model
app.post('/postUser', (req,res)=>{
    user.create(req.body, function(err,data){
        if(err)
            res.status(500).send(err);
        else
        res.send('data.inserted');
    })
})

app.get('/', function(req,res){
    res.send('welcome to api');
});

app.listen(port, function(){
    console.log("running");
});
```

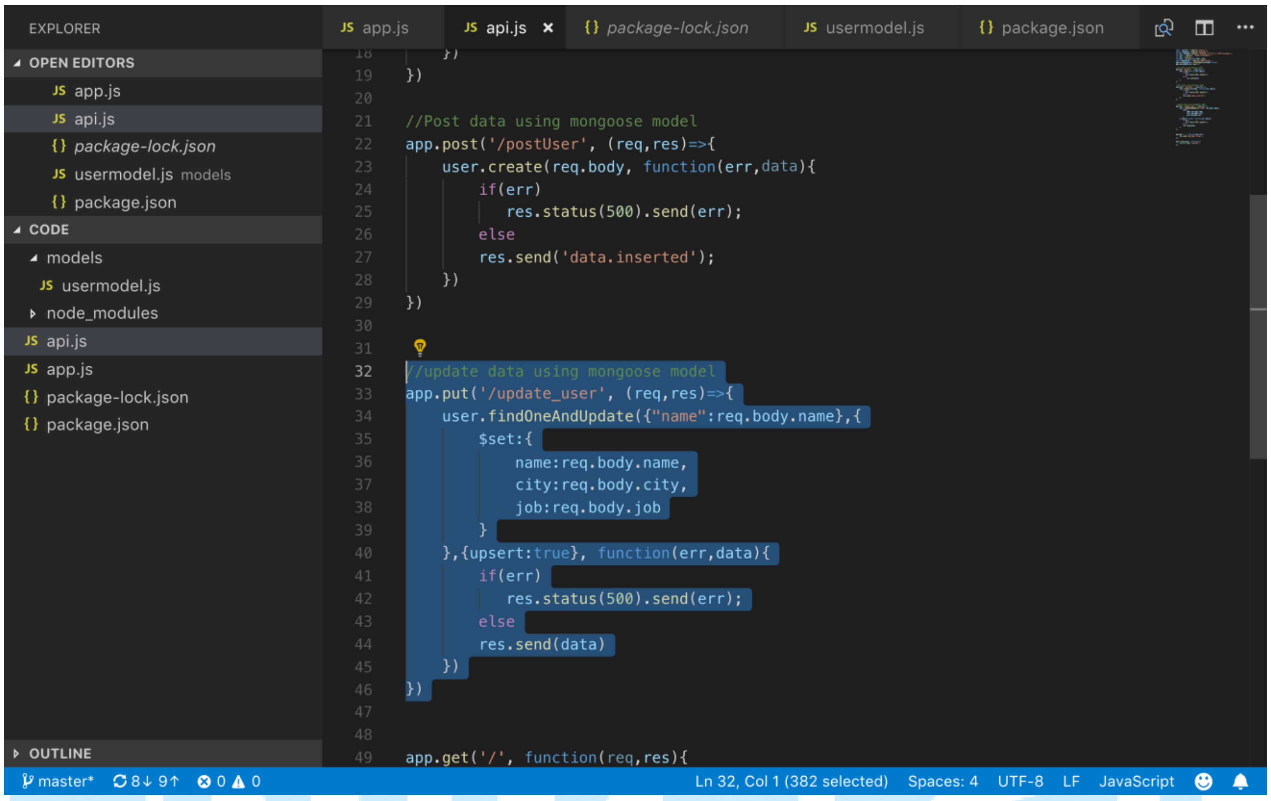
Step 7: When we test API with postman, we will provide the same result but the main difference is that now we have schema designed and connects with mongoose



```
{"name":"Grais","city":"Amsterdam","job":"Project Manager"}
```

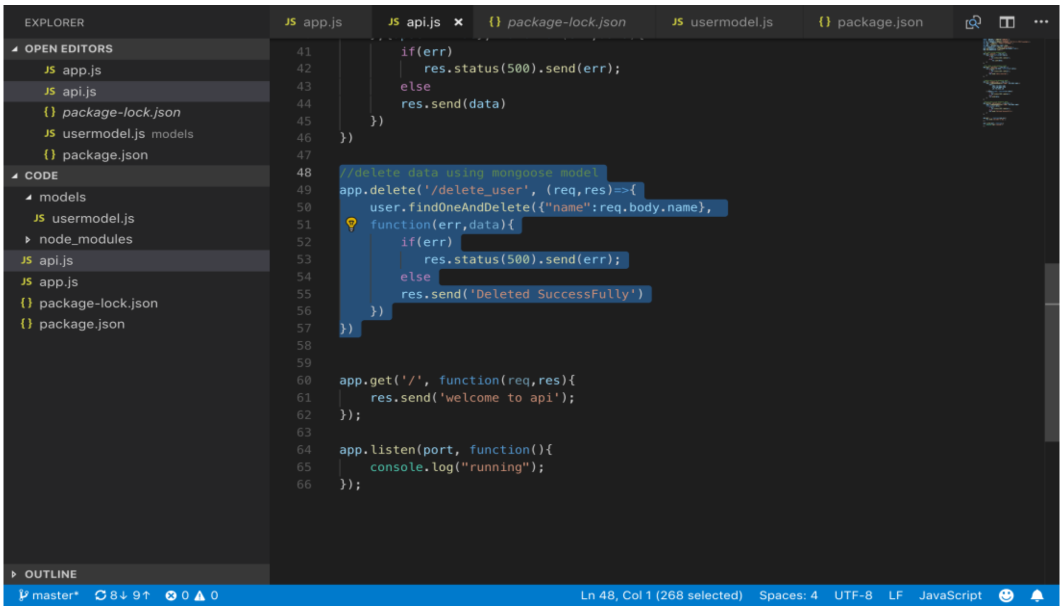
```
data.inserted
```

Step 8: With mongoose PUT query remain same. We are not creating connection again and we don't need to define connection string



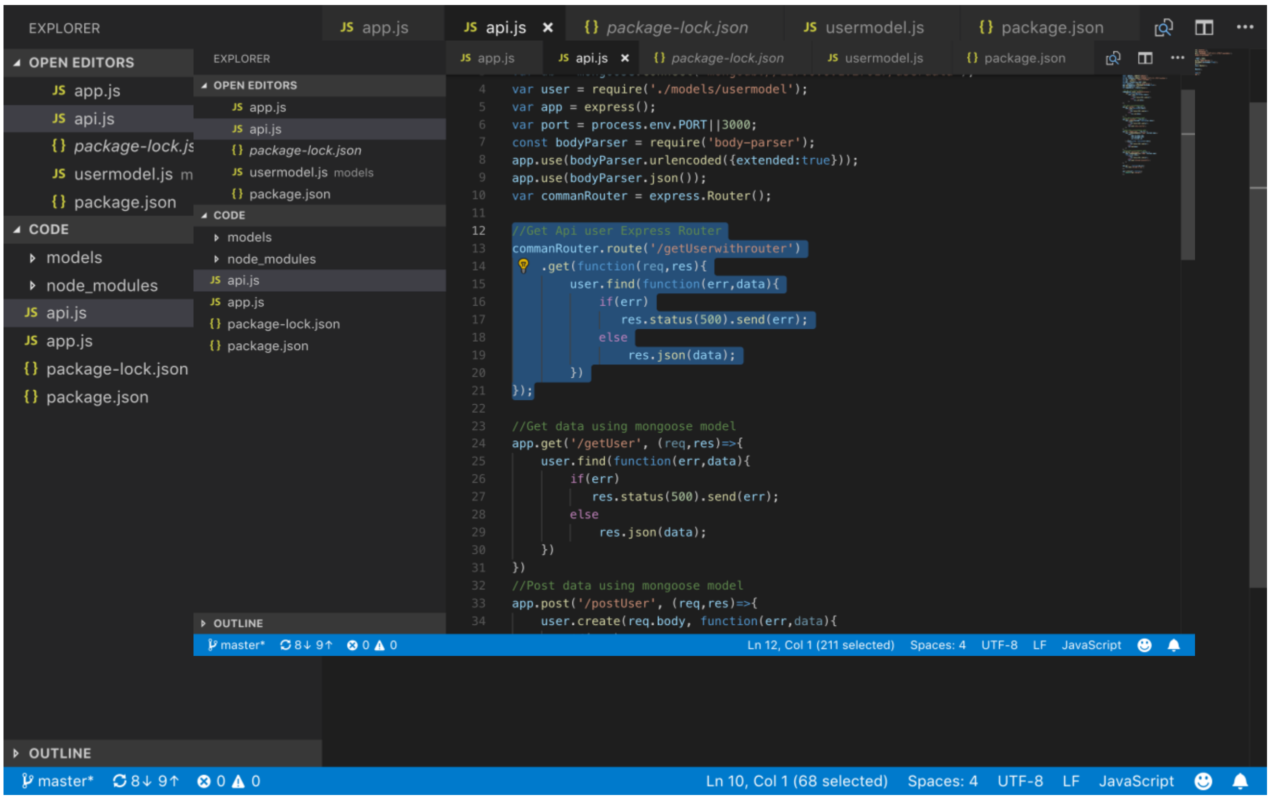
```
})

//Post data using mongoose model
app.post('/postUser', (req,res)=>{
    user.create(req.body, function(err,data){
        if(err)
            res.status(500).send(err);
        else
        res.send('data.inserted');
    })
})


//update data using mongoose model
app.put('/update_user', (req,res)=>{
    user.findOneAndUpdate({"name":req.body.name},{
        $set:{
            name:req.body.name,
            city:req.body.city,
            job:req.body.job
        }
    },{upsert:true}, function(err,data){
        if(err)
            res.status(500).send(err);
        else
        res.send(data)
    })
})


app.get('/', function(req,res){
```

Step 9: With mongoose we can add DELETE query, by using 'findOneAndDelete' of mongodb.



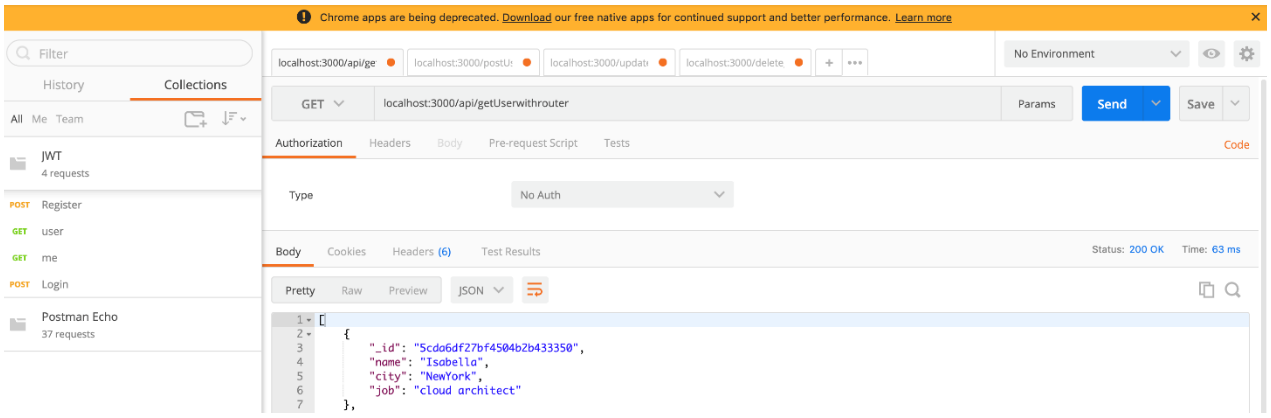
```
        if(err)
            res.status(500).send(err);
        else
        res.send(data)
    })
})

//delete data using mongoose model
app.delete('/delete_user', (req,res)=>{
    user.findOneAndDelete({"name":req.body.name},
    function(err,data){
        if(err)
            res.status(500).send(err);
        else
        res.send('Deleted SuccessFully')
    })
})


app.get('/', function(req,res){
    res.send('welcome to api');
});

app.listen(port, function(){
    console.log("running");
});
```

Step 10: Express have ‘express router’ to maintain rating for details or submenu in the application. Express Router is inbounded in express. We can use it as middleware with app.use. For that first, we need to define common router for the app and then define sub routes



```
var user = require('./models/usermodel');
var app = express();
var port = process.env.PORT||3000;
const bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({extended:true}));
app.use(bodyParser.json());
var commanRouter = express.Router();

//Get Api user Express Router
commanRouter.route('/getUserwithrouter')
    .get(function(req,res){
        user.find(function(err,data){
            if(err)
                res.status(500).send(err);
            else
                res.json(data);
        })
});

//Get data using mongoose model
app.get('/getUser', (req,res)=>{
    user.find(function(err,data){
        if(err)
            res.status(500).send(err);
        else
            res.json(data);
    })
})
//Post data using mongoose model
app.post('/postUser', (req,res)=>{
    user.create(req.body, function(err,data){
```

Step 11: With common router we can define all CRUD operation, but the API linked with it will change like ‘/api/getuserwithrouter’ as /api is common router.



GET localhost:3000/api/getUserwithrouter

Status: 200 OK Time: 63 ms

```
[
    {
        "_id": "5cda6df27bf4504b2b433350",
        "name": "Isabella",
        "city": "NewYork",
        "job": "cloud architect"
    },
```

Thus, we have successfully performed CRUD Operation in Moongoose as well