120A3051
Shreya Idate
Batch: E3

Experiment 9

Aim: Build a RESTful API using MongoDB.

Theory:

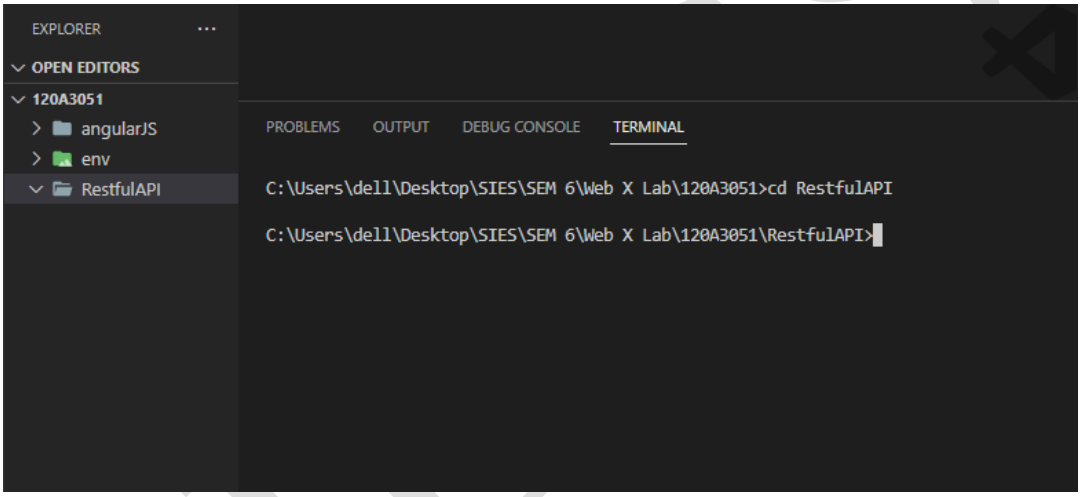
Rest API Overview:

REST stands for REpresentational State Transfer. When a RESTful API is called, the server will transfer to the client a representation of the state of the requested resource. For example, when a developer calls OpenWeather API to fetch weather for a specific city (the resource), the API will return the state of that city, including the temperature, humidity, wind speed, current forecast, extended forecast, and more. The representation of the state can be in a JSON format, and for most web APIs, this is indeed the case. Other possible data formats include XML or HTML.

What does the server does when you call it depends on two things that you need to provide to the server:

1. An identifier for the resource. – This is the URL for the resource, also known as the endpoint. In fact, URL stands for Uniform Resource Locator.
2. The operation you want the server to perform on that resource, in the form of an HTTP method. The common HTTP methods are GET, POST, PUT, and DELETE.

Create folder RestfulAPI and launch in VS code



Enter command npm init to initialize packages .json

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab\120A3051\RestfulAPI>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (restfulapi)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab\120A3051\RestfulAPI\package.json:

{
  "name": "restfulapi",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes)
```

Install Express by
'npm install express' command

```
C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab\120A3051\RestfulAPI>npm install express

added 57 packages, and audited 58 packages in 5s

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

found 0 vulnerabilities
```

Install MongoDB and mongoose if not having in machine

npm install mongodb
npm install mongoose

```
C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab\120A3051\RestfulAPI>npm install mongodb

added 16 packages, and audited 74 packages in 10s

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

found 0 vulnerabilities

C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab\120A3051\RestfulAPI>npm install mongoose

added 9 packages, and audited 83 packages in 8s

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

found 0 vulnerabilities
```

if any change is made in server.to reflect automatically without restarting the server
npm install -g nodemon --save-dev

```
C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab\120A3051\RestfulAPI>npm install -g nodemon --save-dev

changed 32 packages, and audited 33 packages in 5s

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

found 0 vulnerabilities
```

package.json

RestfulAPI > package.json > ...

```
{
  "name": "restfulapi",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2",
    "mongodb": "^5.3.0",
    "mongoose": "^7.0.4"
  }
}
```

Remove test script from package.json and add
"start": "nodemon index.js"

package.json

RestfulAPI > package.json > {} scripts > start

```
{
  "name": "restfulapi",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  Debug
  "scripts": {
    "start": "nodemon index.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2",
    "mongodb": "^5.3.0",
    "mongoose": "^7.0.4"
  }
}
```

Create a file index.js under root folder and modify code like below to connect nodejs with mongodb

index.js

RestfulAPI > index.js > ...

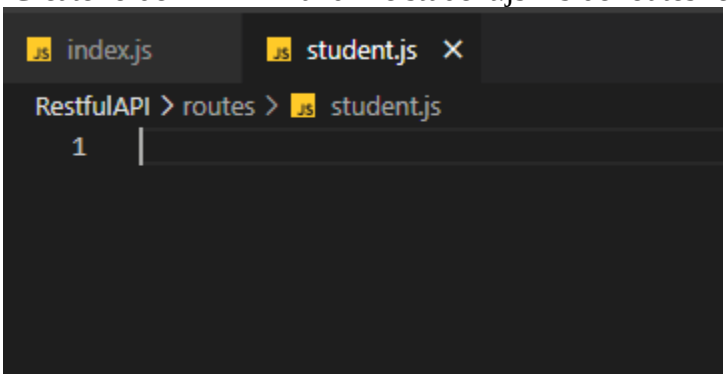
```
const express = require('express')
const mongoose = require('mongoose')
const url = 'mongodb://localhost/reports'
const app = express()
mongoose.connect(url, { useNewUrlParser: true })
const con = mongoose.connection
con.on('open', function () {

    console.log('connected.....')
})
```

Run index.js by command nodemon run start

```
C:\Users\dell\Desktop\SIES\SEM 6\Web X Lab\120A3051\RestfulAPI>nodemon run start
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node run start index.js`
connected.....
```

Create folder “routes” and file student.js inside routes folder



Modify index.js and student.js file as below

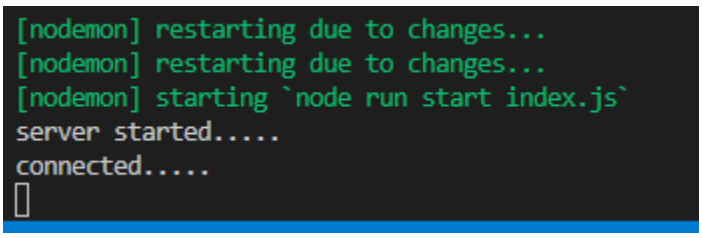
Index.js

```
RestfulAPI > index.js > ...
1   const {Router} = require('express')
2   const express = require('express')
3   const mongoose = require('mongoose')
4   const url = 'mongodb://localhost/reports'
5   const app = express()
6   mongoose.connect(url, { useNewUrlParser: true })
7   const con = mongoose.connection
8   con.on('open', function () {
9
10      console.log('connected.....')
11  })
12
13  const studentRouter=require('./routes/student');
14  app.use('/student', studentRouter)
15  app.listen(9000, () => {
16
17      console.log('server started.....')
18
19  })
```

Student.js

```
index.js   student.js
RestfulAPI > routes > student.js > ...
1   const express = require('express')
2   const router = express.Router()
3   router.get('/', () => { console.log('Get Started') })
4   module.exports = router
5
```

After the output will be below



Here, we are using Router from Express, and we are exporting it too using module.exports. And now, our app will work fine.

How to Write our Endpoints

Now, let's write our endpoints here in this routes file. We will have five routes for the following actions:

Posting data to Database.
Getting all the data from the Database.
Getting data based on the ID.
Updating data based on the ID.
Deleting data based on the ID.

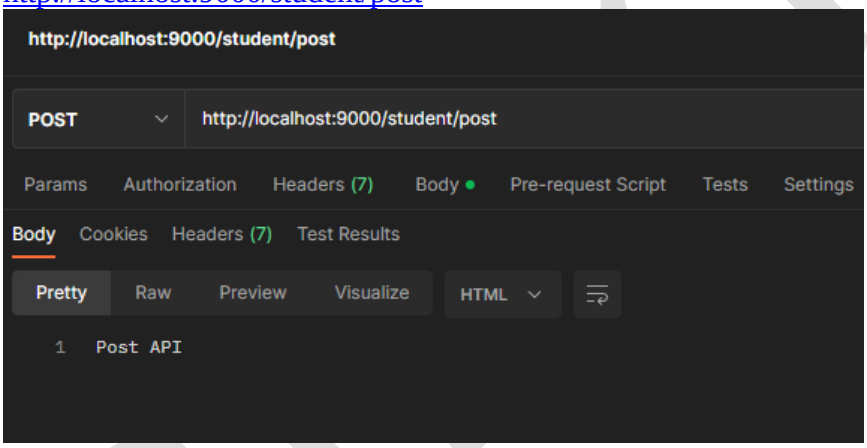
So, let's create the routes for these actions:

```
//Post Method
router.post('/post', (req, res) => {
    res.send('Post API')
})
//Get all Method
router.get('/getAll', (req, res) => {
    res.send('Get All API')
})
//Get by ID Method
router.get('/getOne/:id', (req, res) => {
    res.send('Get by ID API')
})
//Update by ID Method
router.patch('/update/:id', (req, res) => {
    res.send('Update by ID API')
})
//Delete by ID Method
router.delete('/delete/:id', (req, res) => {
    res.send('Delete by ID API')
})
```

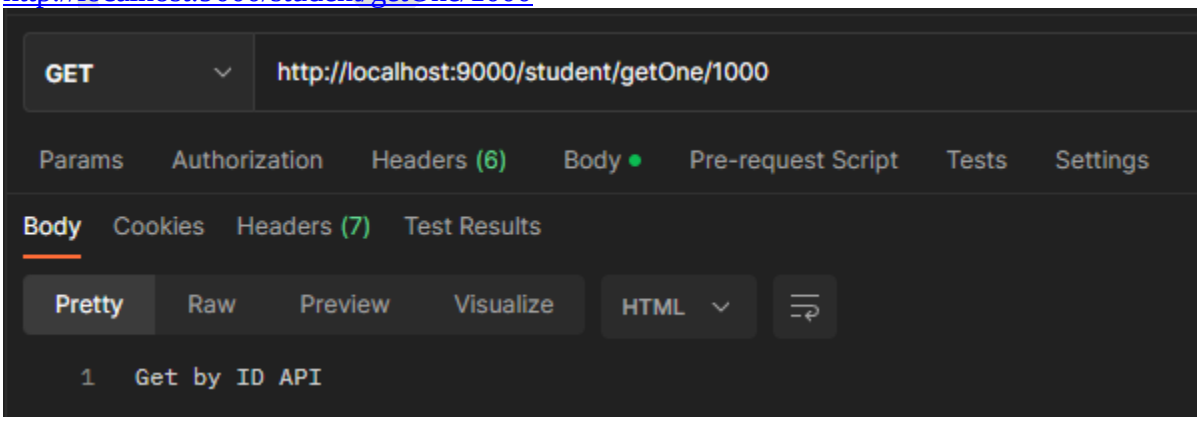
Student.js

```
index.js   student.js
RestfulAPI > routes > student.js > ...
1   const express = require('express')
2   const router = express.Router()
3   // router.get('/', () => { console.log('Get Started') })
4   module.exports = router
5
6   //Post Method
7   router.post('/post', (req, res) => {
8       res.send('Post API')
9   })
10  //Get all Method
11  router.get('/getAll', (req, res) => {
12      res.send('Get All API')
13  })
14  //Get by ID Method
15  router.get('/getOne/:id', (req, res) => {
16      res.send('Get by ID API')
17  })
18  //Update by ID Method
19  router.patch('/update/:id', (req, res) => {
20      res.send('Update by ID API')
21  })
22  //Delete by ID Method
23  router.delete('/delete/:id', (req, res) => {
24      res.send('Delete by ID API')
25  })
```

Save this, and open Postman to check the endpoints.
Download Postman if you don't have it using link- https://www.postman.com/downloads/
http://localhost:9000/student/post



http://localhost:9000/student/getOne/1000



How to Create the Model
Now, let's create a Model that will define our database structure.
Create a folder called model and inside, a file called model.js.Add below code

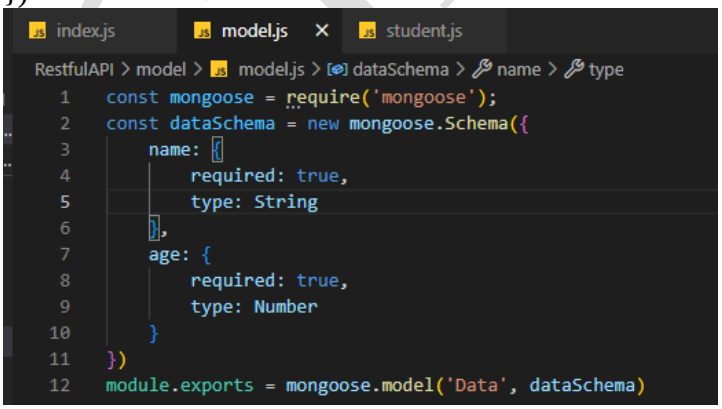
```
const mongoose = require('mongoose');
const dataSchema = new mongoose.Schema({
    name: {
        required: true,
        type: String
    },
    age: {
        required: true,
        type: Number
    }
})
module.exports = mongoose.model('Data', dataSchema)
```

Modify student.js file as below

```
const express=require('express')
const router=express.Router()
const Model = require('../model/model');
module.exports=router
router.post('/post', async (req, res) => {
    const data = new Model({
        name: req.body.name,
        age: req.body.age
    })

    try {

        const dataToSave = await data.save();
        res.status(200).json(dataToSave)
    }
    catch (error) {
        res.status(400).json({message:  error.message})
    }
})
```



Modify index.js file as highlited below

```
const { Router } = require('express')
const express=require('express')
const mongoose=require('mongoose')
const url='mongodb://localhost/reports'
mongoose.connect(url);
const database = mongoose.connection;
database.on('error', (error) => {
   console.log(error)
})
database.once('connected', () => {
   console.log('Database Connected');
})const app=express()
mongoose.connect(url,{useNewUrlParser:true})
/*const con=mongoose.connection
con.on('open',function()
{
   console.log('connected..... ')
})*/
app.use(express.json())
const studentRouter=require('./routes/student')
app.use('/student',studentRouter)
app.listen(9000,() =>
{
   console.log('server started. ... ')
})
```

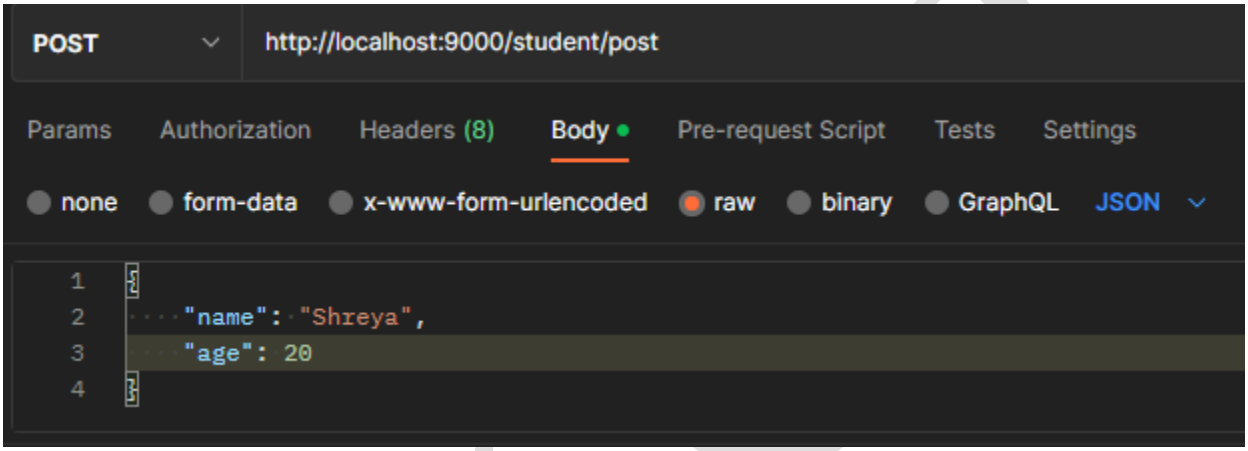
index.js × model.js student.js

RestfulAPI > index.js > ...

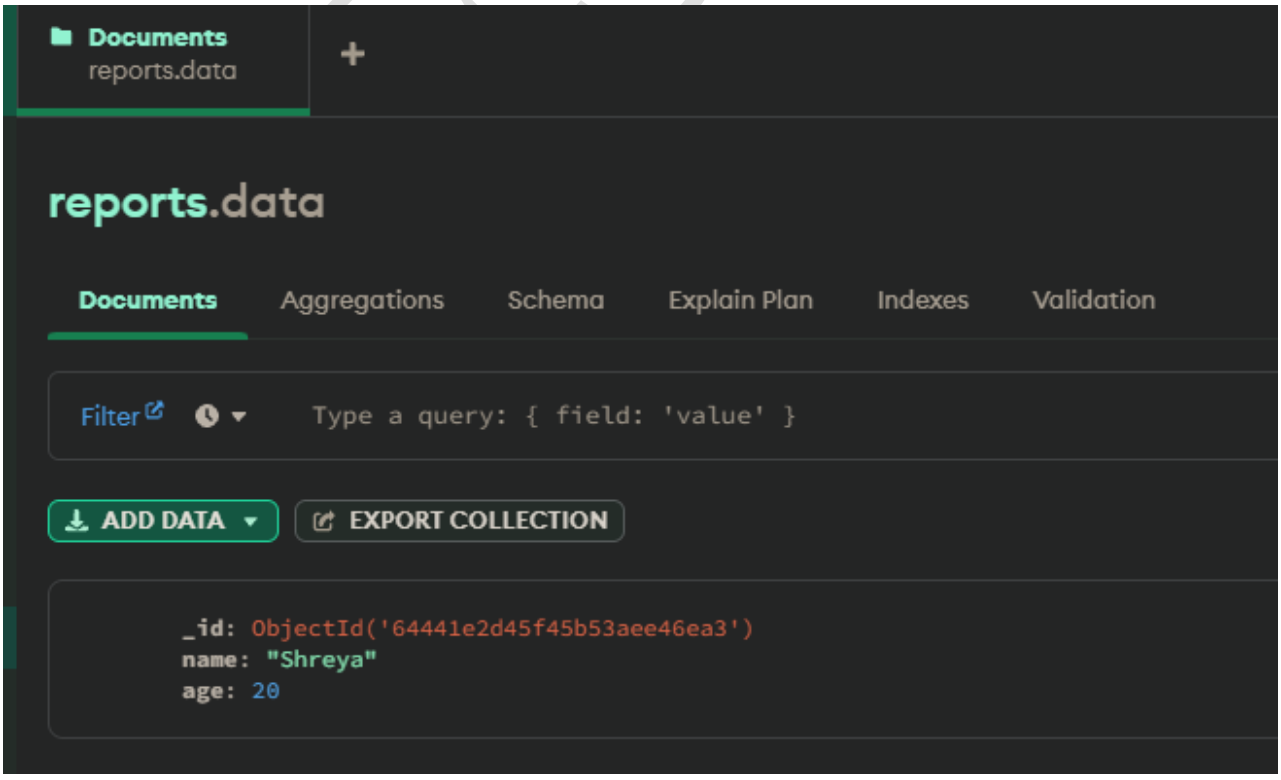
```
1  const { Router } = require('express')
2  const express = require('express')
3  const mongoose = require('mongoose')
4  const url = 'mongodb://localhost/reports'
5  mongoose.connect(url);
6  const database = mongoose.connection;
7  database.on('error', (error) => {
8      console.log(error)
9  })
10 database.once('connected', () => {
11     console.log('Database Connected');
12 })
13 const app = express()
14 mongoose.connect(url, { useNewUrlParser: true })
15 /*const con=mongoose.connection con.on('open',function()
16 {
17 console.log('connected  ')
18 })*/
19 app.use(express.json())
20 const studentRouter = require('./routes/student')
21 app.use('/student', studentRouter)
22 app.listen(9000, () => {
23     console.log('server started.    ')
24 })
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node run start index.js`
server started.
Database Connected
```

If we add the data in the body and click Send, we will get the following:



It's also generating a unique ID. Open the MongoDB Compass app, and you will see the database and this record you just created:



Conclusion: Thus, we have successfully created a RESTAPI using flask and MongoDB.