Aim:

Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations

Theory:

In this tutorial, we are going to build a simple Student management application. We will build Rest APIs for creating, listing, editing and deleting a student records using node, express & mongoDB.

Express

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node-based Web applications.

MongoDB and Mongoose

MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need.
Mongoose is an ODM (Object Document Mapping) tool for Node.js and MongoDB. It helps you convert the objects in your code to documents in the database and vice versa.
Setup MongoDB
To work with MongoDB, you need to have MongoDB installed in your system. Check out the official **MongoDB doc** for installing MongoDB in your System.

Let's list the tools and technologies that we will use in this tutorial.

Tools and technologies used

1. node.js (npm)
2. express
3. request body
4. postman
5. visual studio code IDE

Development steps

1. Creating an application
2. Install dependencies

3. Setting up the webserver
4. Configuring and Connecting to the database
5. Defining the student model in MongoDB
6. Defining Routes using Express
7. Developing the Restful APIs
8. Testing our APIs
9. Conclusion

• Creating an application

1. Open a terminal and create a new folder for the application.

```
$ mkdir todo-app
```

2. Initialize the application with a *package.json* file

Go to the root folder of your application and type *npm init* to initialize your app with a *package.json* file.

```
$ cd todo-app
$ npm init
```

Here is the complete *package.json* file:

```
{
  "name": "todo-app",
  "version": "1.0.0",
  "description": "Todo App",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "Express",
    "RestAPI",
    "MongoDB",
    "postman",
    "Todos"
  ],
  "author": "Rohit",
  "license": "avcoe"
}
```

Note that I've specified a file named *index.js* as the entry point of our application. We'll create a *index.js* file in the next section.

• Install dependencies

We will need *express* and *mongodb* modules in our application. Let's install them by typing the following command -

```
$ npm install express
$ npm install mongodb
```

• Setting up the webserver

Let's now create the main entry point of our application. Create a new file named *index.js* in the root folder of the application with the following contents -

```
const express = require('express');

// create express app
const app = express();


// parse application/json
app.use(express.json())

// define a simple route
app.get('/', (req, res) => {
   res.send({"message": "Welcome to Todo app"});
});

// listen for requests
app.listen(4000, () => {
   console.log("Server is listening on port 4000");
});
```

Let's now run the server and go to **http://localhost:4000** to access the route we just defined.

```
$ npm install
$ node index.js

Server is listening on port 4000
```

• Configuring and Connecting to the database

Let's create *mongodb.js* inside the **config** folder with the following contents. We'll now import the above database configuration in *index.js* and connect to the database using **mongodb**

```
const {MongoClient}=require('mongodb');
const url="mongodb://localhost:27017"
const database='student';
const client=new MongoClient(url);

const dbConnect=async()=>{
    const result=await client.connect();
    const db=await result.db(database);
    return db.collection('profile');
}

module.exports=dbConnect;
```
.

The complete *index.js* file looks like:

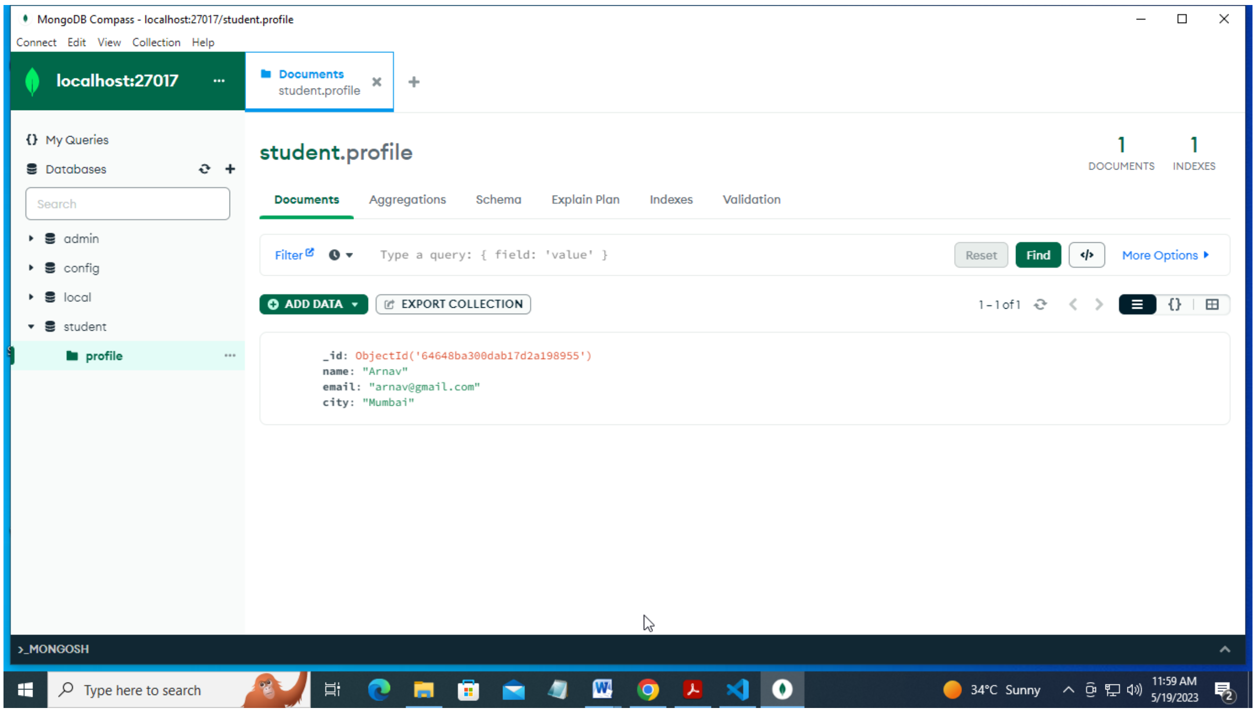
```
const dbConnect=require('./mongodb')
const express=require('express');
const { response } = require('express');
const app=express();
app.use(express.json())
//get API

app.get('/getData',async(req,res)=>{
let result=await dbConnect();
result=await result.find().toArray();
res.send(result);
})

//post API
app.post('/insertData',async(req,res)=>{
    let result=await dbConnect();
    result=await result.insertOne(req.body);
    res.send("Data Inserted Successfully")
})

// Put API
app.put('/updateData/:name',async(req,res)=>{
    let result=await dbConnect();
    result=await result.updateOne({name:req.params.name},{$set:req.body});
    res.send("Data Updated Successfully")
})


//Delete API
app.delete('/deleteData/:name',async(req,res)=>{
    let result=await dbConnect();
    result=await result.deleteOne({name:req.params.name})
    res.send("Data Deleted Successfully");
})



app.listen(3000);
```

Let's fire the below command to make sure that you're able to connect to the database:

```
$ node index.js
Server is listening on port 4000
Successfully connected to the database
```

- ## MongoDB Compass

We have created profile collection inside student database.



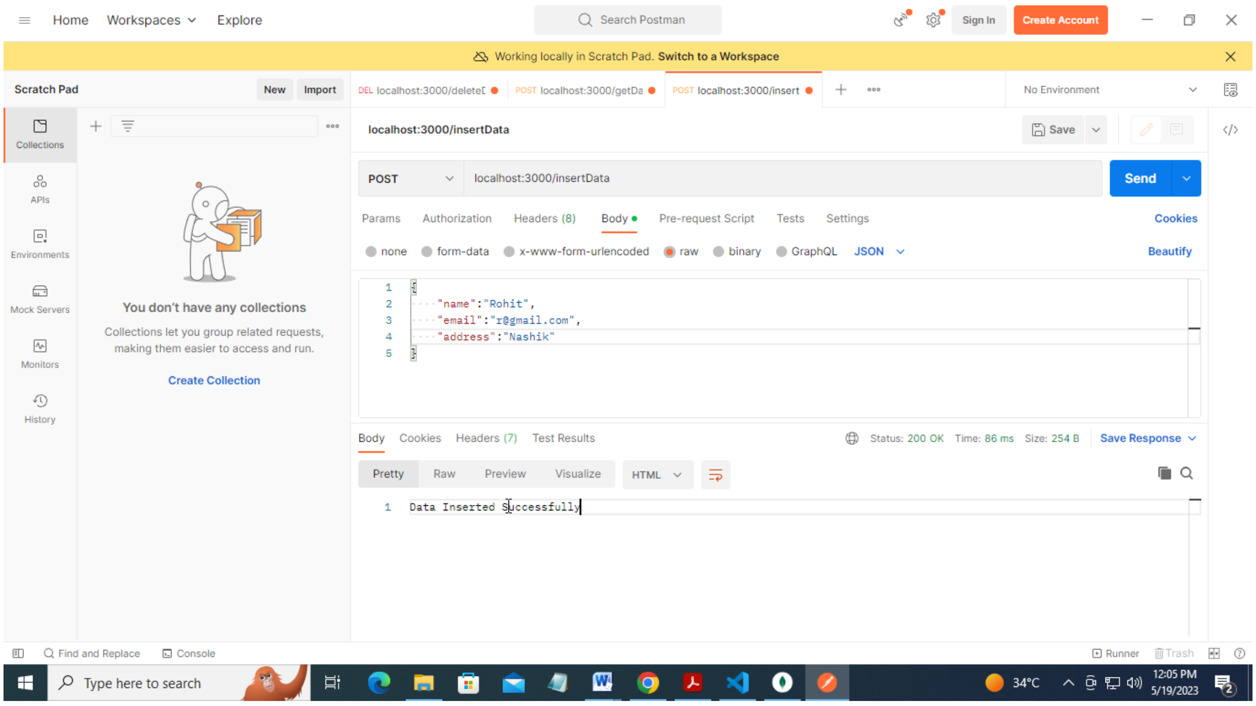
```
_id: ObjectId('64648ba300dab17d2a198955')
name: "Arnav"
email: "arnav@gmail.com"
city: "Mumbai"
```

- ## Testing our APIs

Let's now test all the APIs one by one using postman.

CM/ADL-D-11	Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations	**Page**	**06/08**
Experiment No.: 03-B	**Semester – II**	**Rev.: 00**	**Date: 15-06-17**

Create a new Todo using POST /insertData API



localhost:3000/insertData

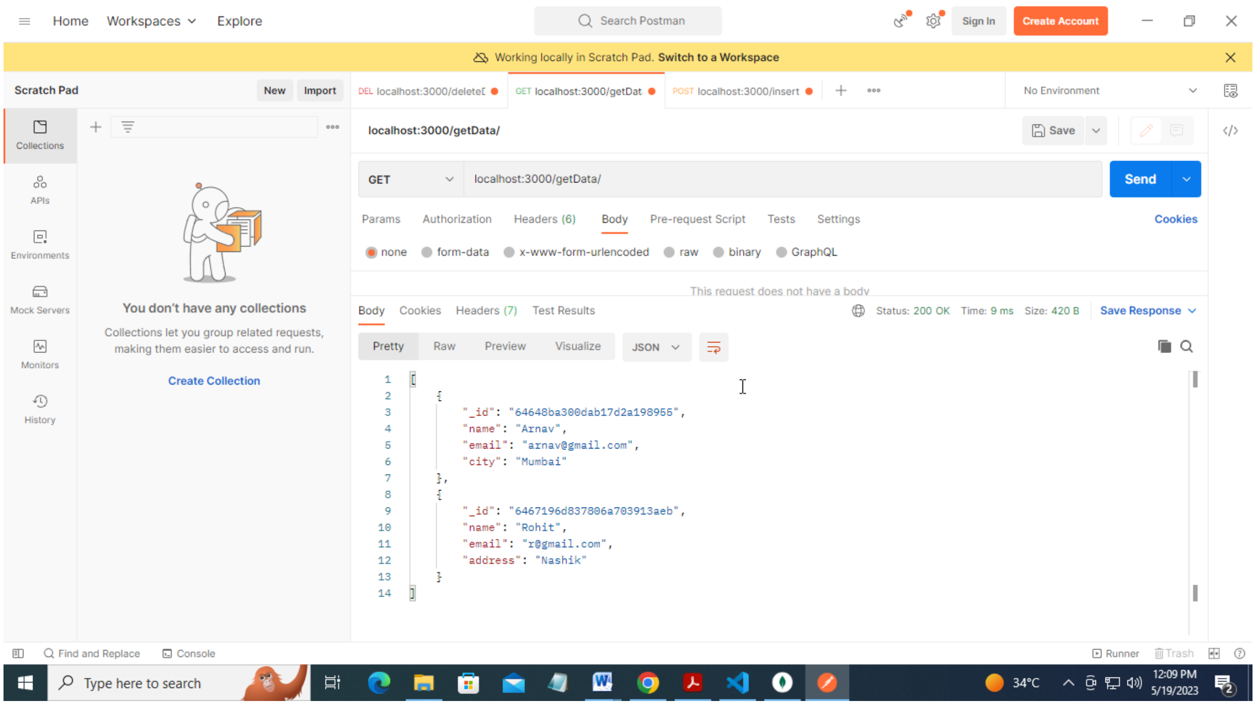
POST localhost:3000/insertData

```
{
    "name":"Rohit",
    "email":"r@gmail.com",
    "address":"Nashik"
}
```

Status: 200 OK Time: 86 ms Size: 254 B

```
Data Inserted Successfully
```

Retrieving all Todos using GET /getData API



localhost:3000/getData/

GET localhost:3000/getData/

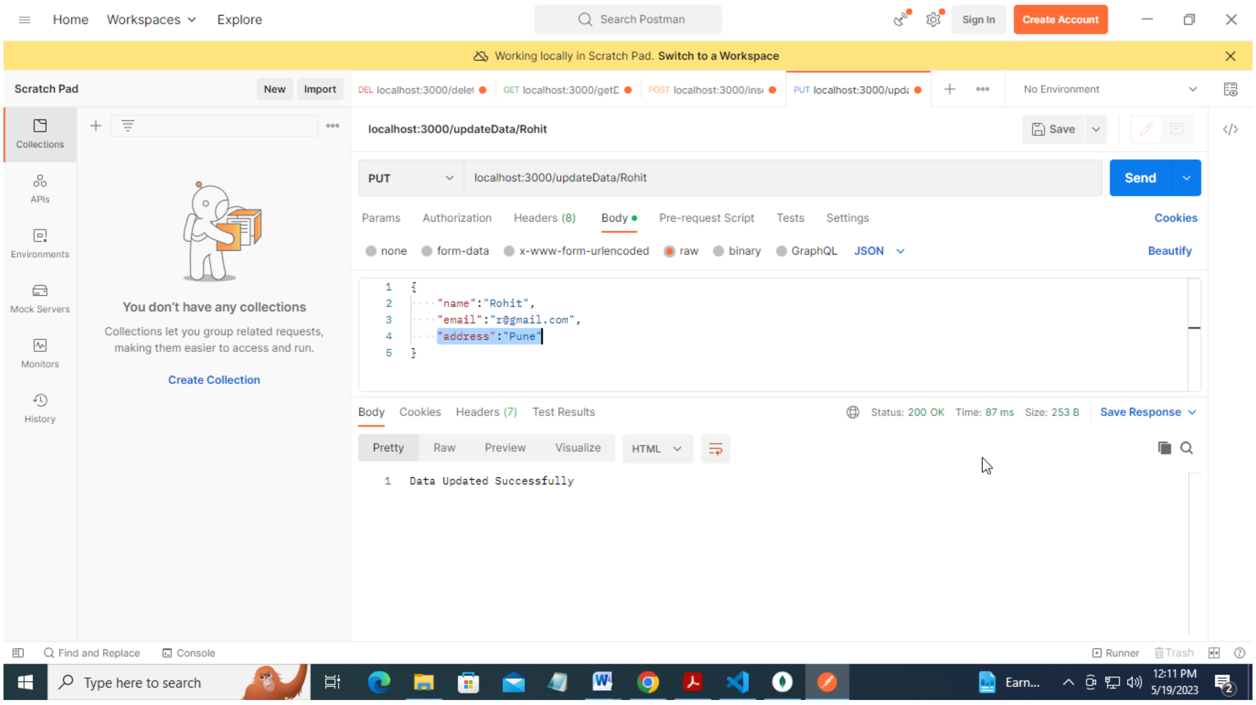
This request does not have a body

Status: 200 OK Time: 9 ms Size: 420 B

```
[
    {
        "_id": "64648ba300dab17d2a198955",
        "name": "Arnav",
        "email": "arnav@gmail.com",
        "city": "Mumbai"
    },
    {
        "_id": "6467196d837806a703913aeb",
        "name": "Rohit",
        "email": "r@gmail.com",
        "address": "Nashik"
    }
]
```

CM/ADL-D-11	Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations	Page	07/08
Experiment No.: 03-B	Semester – II	Rev.: 00	Date: 15-06-17

Updating a Todo using PUT localhost:3000/updateData/Rohit API



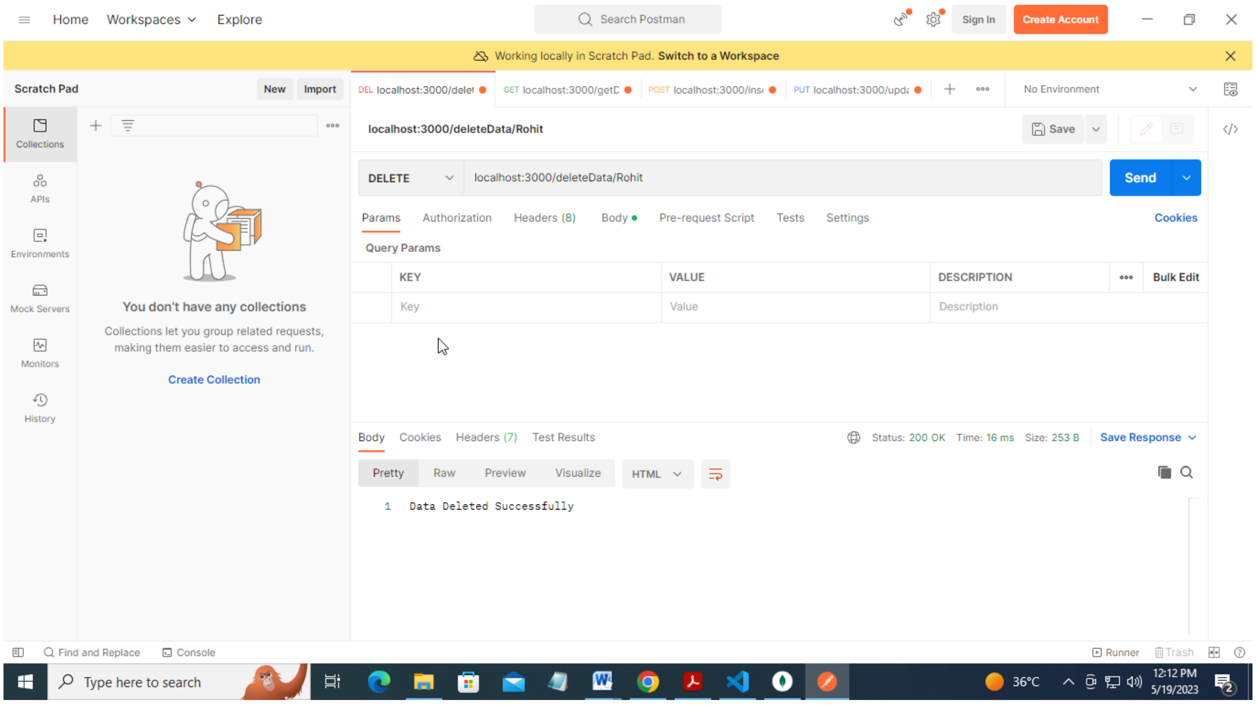
PUT localhost:3000/updateData/Rohit

```
{
    "name":"Rohit",
    "email":"r@gmail.com",
    "address":"Pune"
}
```

Status: 200 OK Time: 87 ms Size: 253 B

```
Data Updated Successfully
```

Deleting a Todo using DELETE localhost:3000/deleteData/Rohit API



DELETE localhost:3000/deleteData/Rohit

Status: 200 OK Time: 16 ms Size: 253 B

```
Data Deleted Successfully
```

• Conclusion

In this tutorial, we have developed a RESTful CRUD (Create, Retrieve, Update, Delete) API with **Node.js, Express**, and **MongoDB**.