SIVA HARI A S (2018503060)

A **RESTful API** was built which performs **CRUD**(Create, Replace, Update and Delete) operations on employee database

REST API

- ➔ A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services
- ➔ When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint.
- ➔ This information, or representation, is delivered in one of several formats via **HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text**

Tools used : ExpressJs, Postman

ExpressJs

- ➔ ExpressJs, or simply Express, is a back end web application framework for Node.js
- ➔ It is designed for building web applications and APIs
- ➔ It allows specific handling for different **HTTP verbs (e.g. GET, POST, DELETE, etc.),** separately handle requests at different URL paths ("routes"), serve static files, or use templates to dynamically create the response

Postman

- ➔ Postman is a popular API client that makes it easy for developers to create, share, test and document APIs
- ➔ This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses

Server.js

```
const express = require('express')
const bodyParser = require('body-parser')
const usersRoutes = require('./routes/users.js')

const app = express()
const PORT = 5000

app.use(bodyParser.json())
app.use('/users', usersRoutes)

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

```
    res.send(`Hello from home page`)
})

app.listen(PORT, () => {
    console.log(`Server running on port: http://localhost:${PORT}`)
})
```

Routes.js

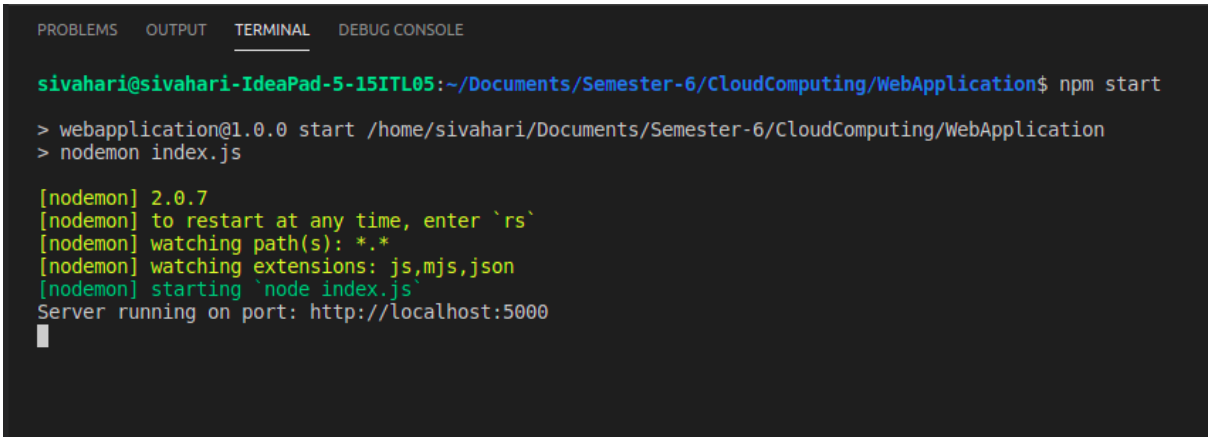
```
const express = require('express')
const {v1: uuid} = require('uuid');

const router = express.Router()

// all routes in here are starting with /users
// mockDatabase
let users = [
    {
        firstName: "Ryan",
        lastName: "Harris",
        age: 37,
        userId: uuid()
    },
    {
        firstName: "Marshal",
        lastName: "Doe",
        age: 24,
        userId: uuid()
    }
]
router.get('/', (req,res) => {
    res.send(users)
})

router.post('/', (req,res) => {
    const user = {
        firstName: req.body.firstName,
        lastName: req.body.lastName,
        age: req.body.age,
        userId: uuid()
```

```
    }
    users.push(user)
    res.send(`User with name ${req.body.firstName} added`)
})

// /users/2 => req.params { id: 2}

router.get('/:id', (req,res) => {
    const id = req.params.id.toString()
    const foundUser = users.find((user) => user.userId === id)
    res.send(foundUser)

})

router.delete('/:id', (req,res) => {
    const id = req.params.id
    users = users.filter((user) => user.userId !== id)
    res.send(`userId: ${id} successfully deleted!!`)
})

router.patch('/:id', (req,res) => {
    const id = req.params.id.toString()
    const {firstName, lastName, age} = req.body
    const user = users.find((user) => user.userId === id)
    //console.log(user)
    if(firstName) {
        user.firstName = firstName
    }
    if(lastName) {
        user.lastName = lastName
    }
    if(age) {
        user.age = age
    }
    res.send(`userId: ${id} updated successfully!!`)
})

module.exports = router
```

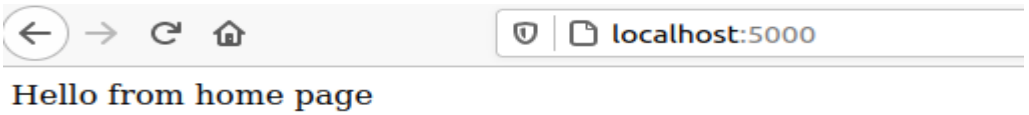
SIVA HARI A S (2018503060)

Getting details about employees:

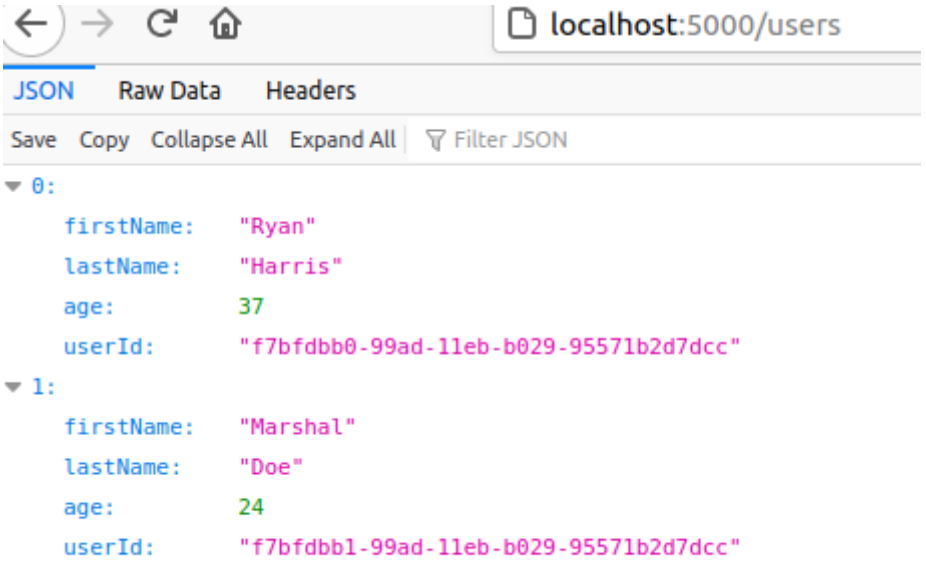
➔ Starting the server



➔ Opening the port 5000 on browser : **http://localhost:5000**

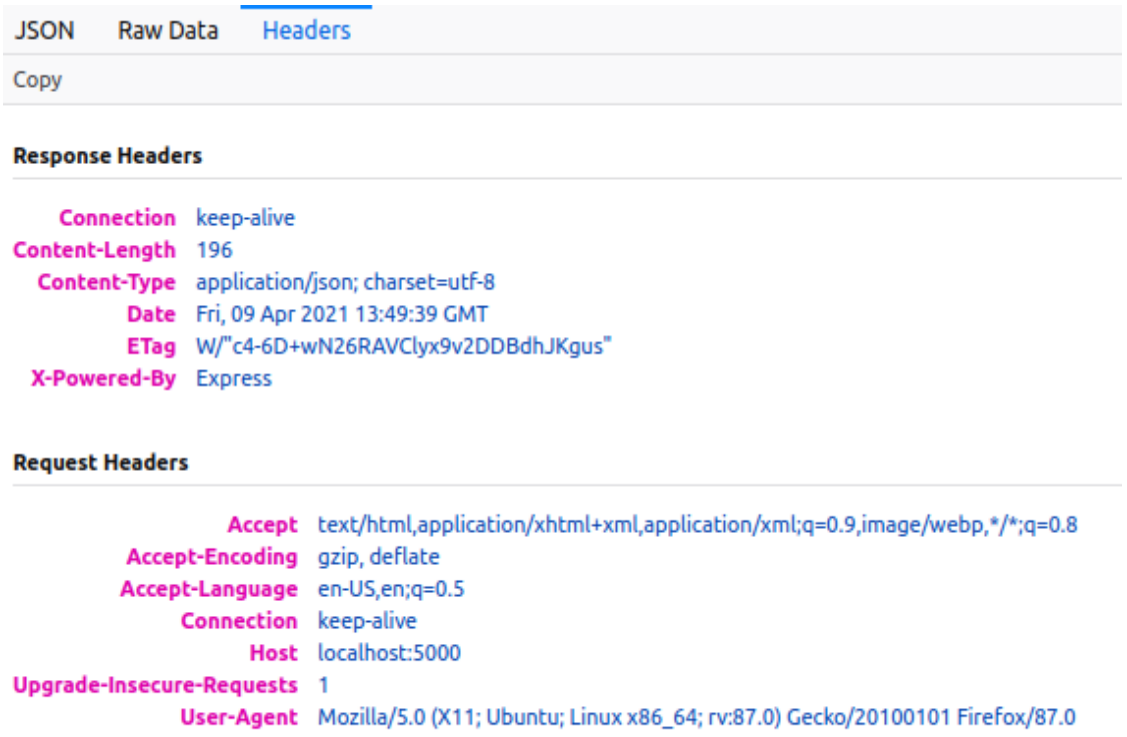


➔ Fetching the employee data by **HTTP-GET** request from the server, we hit the end point: **http://localhost:5000/users**



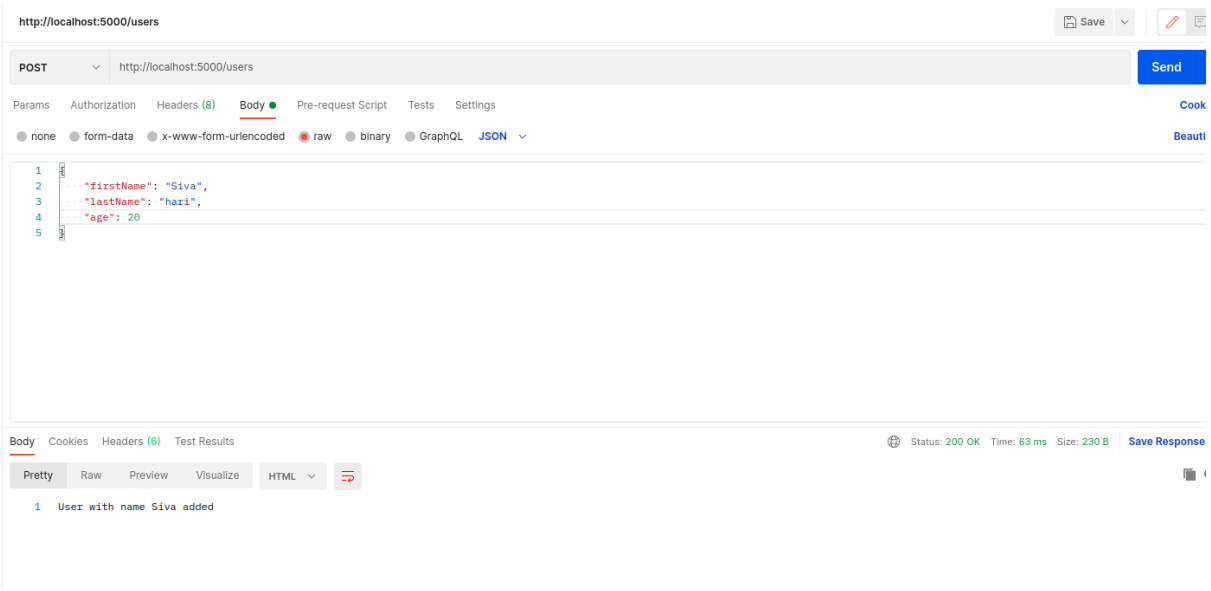
➔ We take a look at the headers of the request and response

SIVA HARI A S (2018503060)



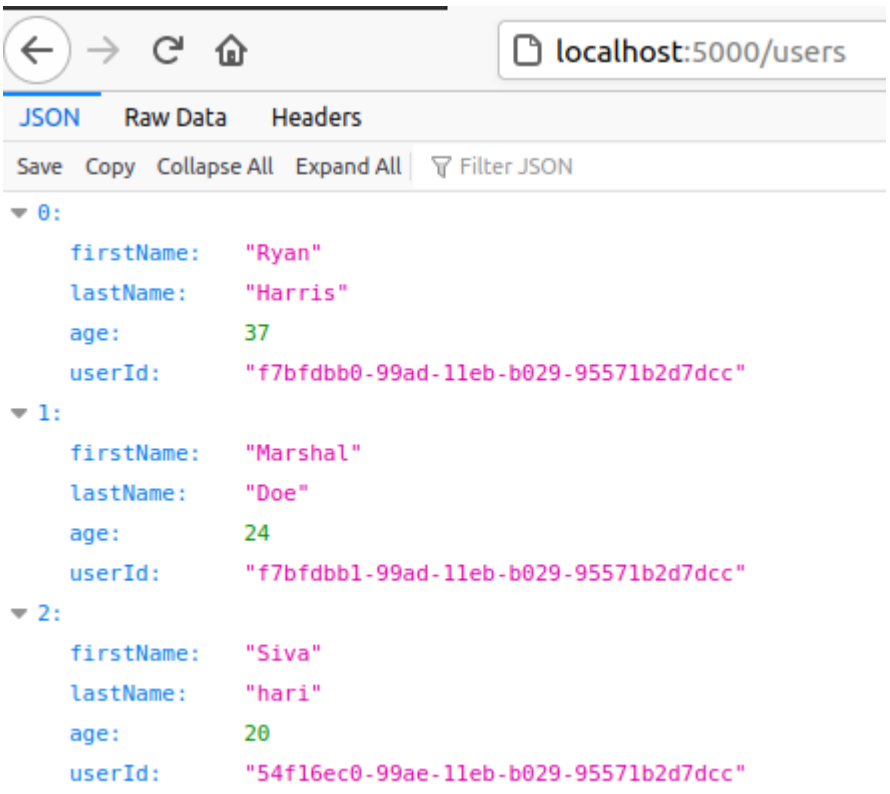
Creating a new employee:

➔ We give a post request to the server using postman to the server with all the details of the new employee



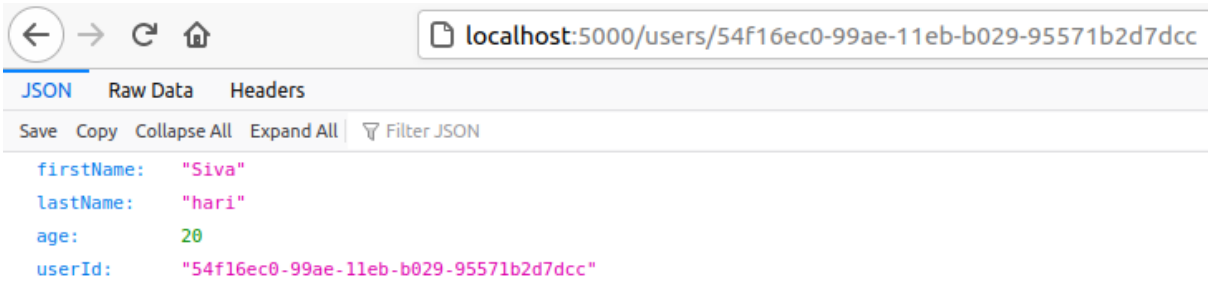
➔ We go to the localhost and fetch all the users by sending a HTTP-GET request to the server to check if the new employee was added to database or not

SIVA HARI A S (2018503060)



Getting info about a specific employee using UUID(unique id assigned to each employee):

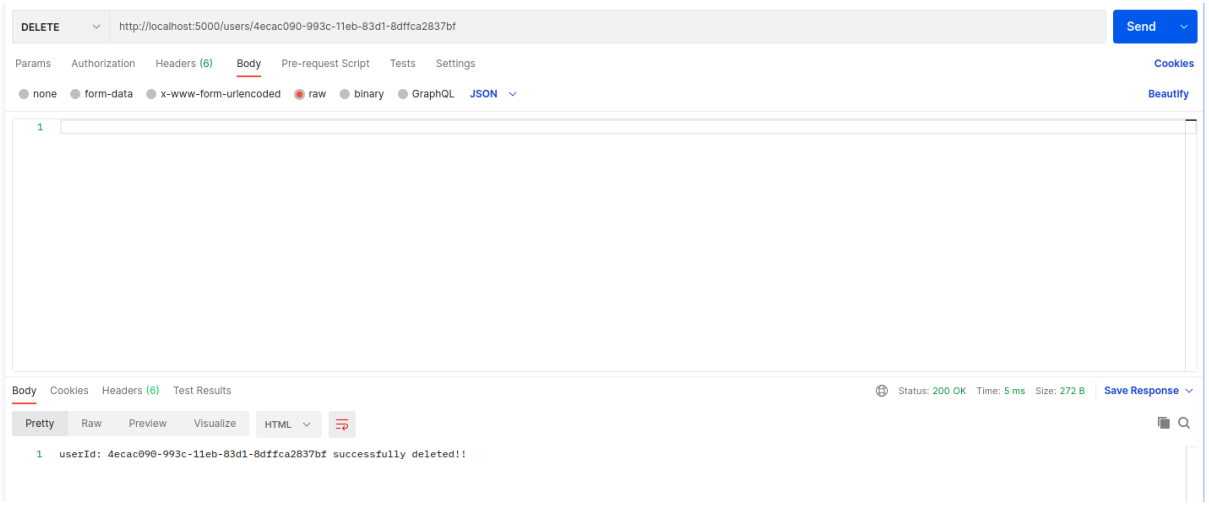
- ➔ We give a HTTP-GET request to the server with url: http://localhost:5000/users/(uuid of employee)



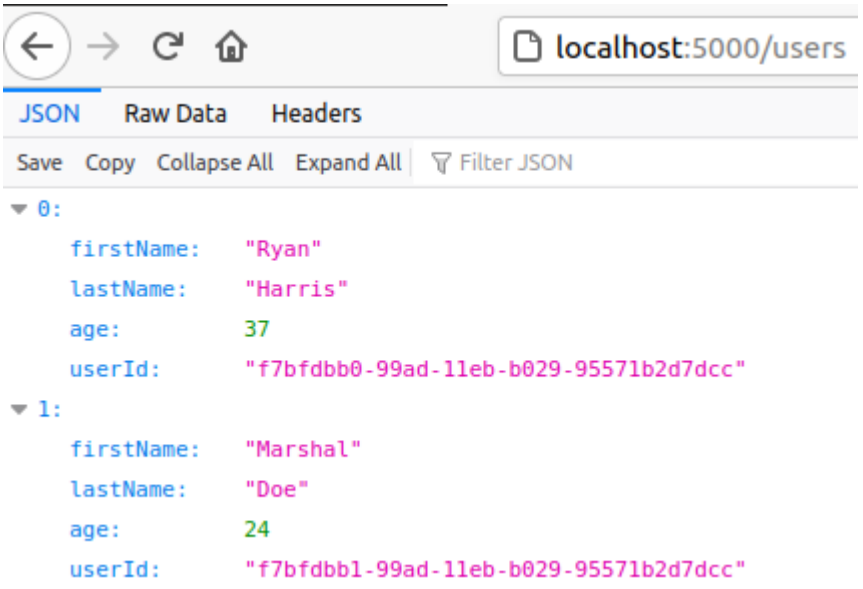
Deleting a particular employee:

- ➔ We give a HTTP-DELETE request from the browser with the unique uuid to delete that particular employee

SIVA HARI A S (2018503060)



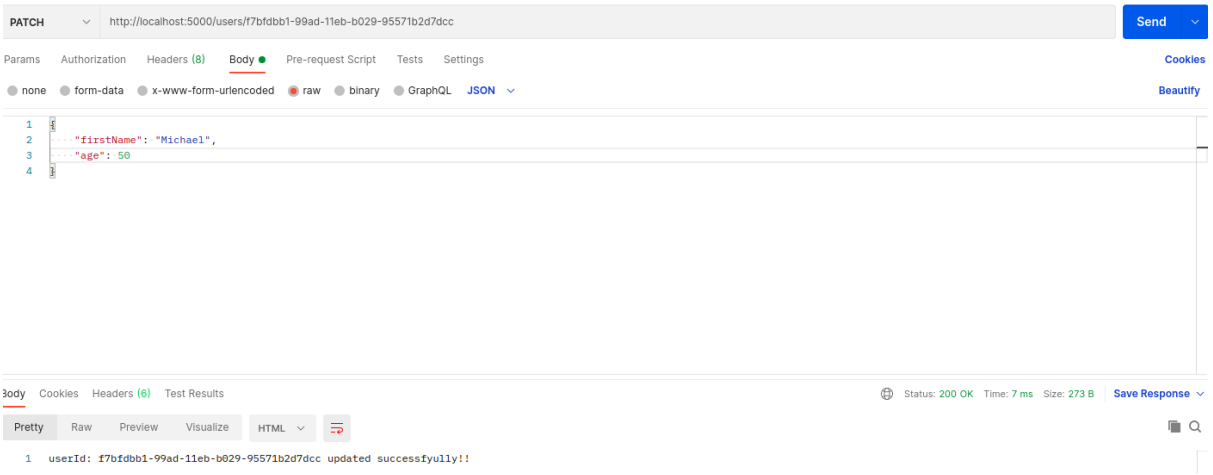
➔ We fetch all the users by sending HTTP-GET request from the server to check whether the employee was deleted or not



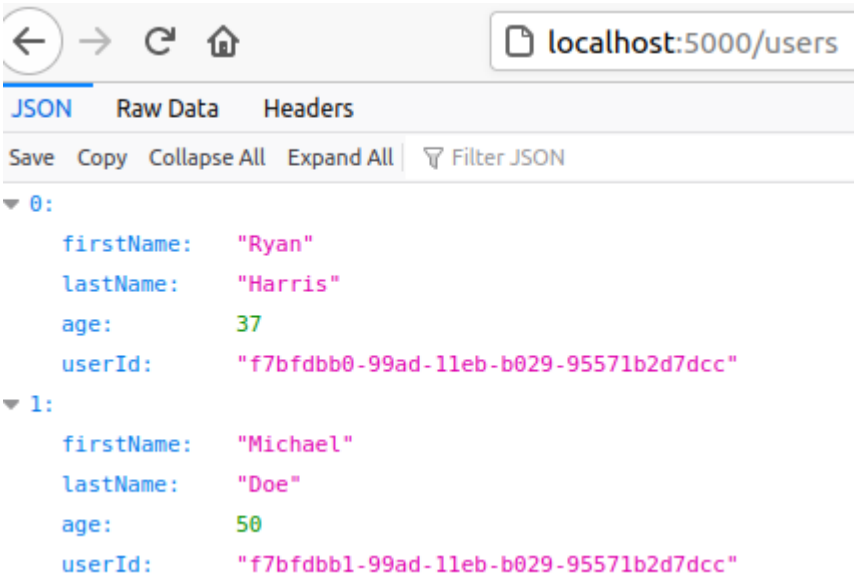
Updating the details of a particular employee:

➔ We give a HTTP-PATCH request from postman with the UUID and the details to be modified

SIVA HARI A S (2018503060)



➔ Fetching the employees by giving HTTP-GET request to the server and seeing if the details were updated or not



VIDEO DEMONSTRATION OF THE ASSIGNMENT

https://drive.google.com/file/d/1WoU33mbPhV_BAoudd4zFGNdLa6tG15K0/view?usp=sharing

---