WEB UI Forms

Description:

This project is a built using Angular framework (Angular CLI: 16.2.3) and MongoDB is used for database. NodeJS (v20.5.0) and Express (9.8.1) are used for server-side scripting. The functionality of the web app is interacting with the database.

How to run:

- Complete the installation process.
- Open command prompt and set path to project folder(Frontend and Backend).
- For Frontend, run ng serve.
- For backend, run node server.js.

Installation:

1. **NodeJS:**
   i. Download and install the latest version of NodeJS from the browser.

```
C:\Users\arunk>node -v
v20.5.0
```

2. **Express:**
   i. Open the command prompt, install express using npm(node package manager).

```
npm install express
```

   ii. Latest version of express will be installed.

3. **MongoDB:**
   i. Download and install MongoDB from the edge or chrome.
   ii. Install MongoDB Compass.

4. **Angular:**
   i. Open command prompt with the path of the project folder.
   ii. Enter the following command in cmd.

```
npm install -g @angular/cli
```

iii. Angular is installed globally.

5. **Postman**
   i. Install Postman from the edge or chrome.

Code Walkthrough:

a) Back-end

i. Create a javascript file (server.js) and import the necessary modules (express, cors, mongoose, and ./routes/routes)

```
var express = require('express');
var server = express();
var routes = require('./routes/routes');
const cors = require('cors');
var mongoose = require('mongoose');
```

ii. Create a server connection of port: 8000.

```
server.listen(8000,
    function check(error)
    {
        if(error)
        {
            console.log(error);
        }
        else
        {
            console.log("Server established!");
        }
    } );
```

iii. In the command prompt, run server.js file.

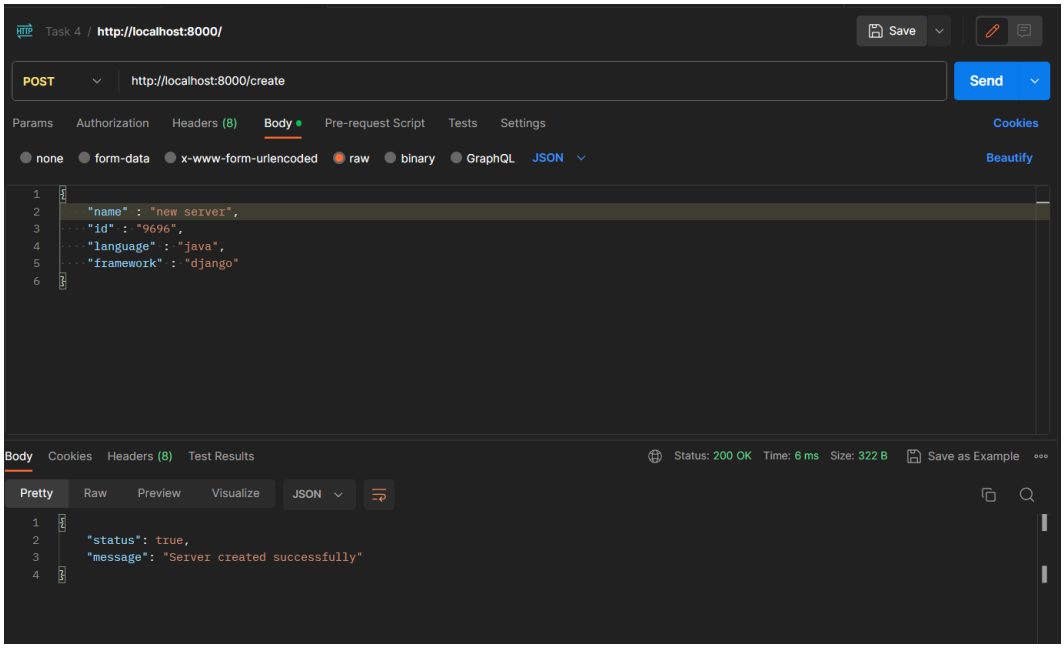
```
E:\Projects\Task 4\BackEnd>node server.js
Server established!
Connected to MongoDB
```

iv. Create a folder named router and create a router.js file in it.
v. In router.js file, create routing paths for create, read, update, delete.

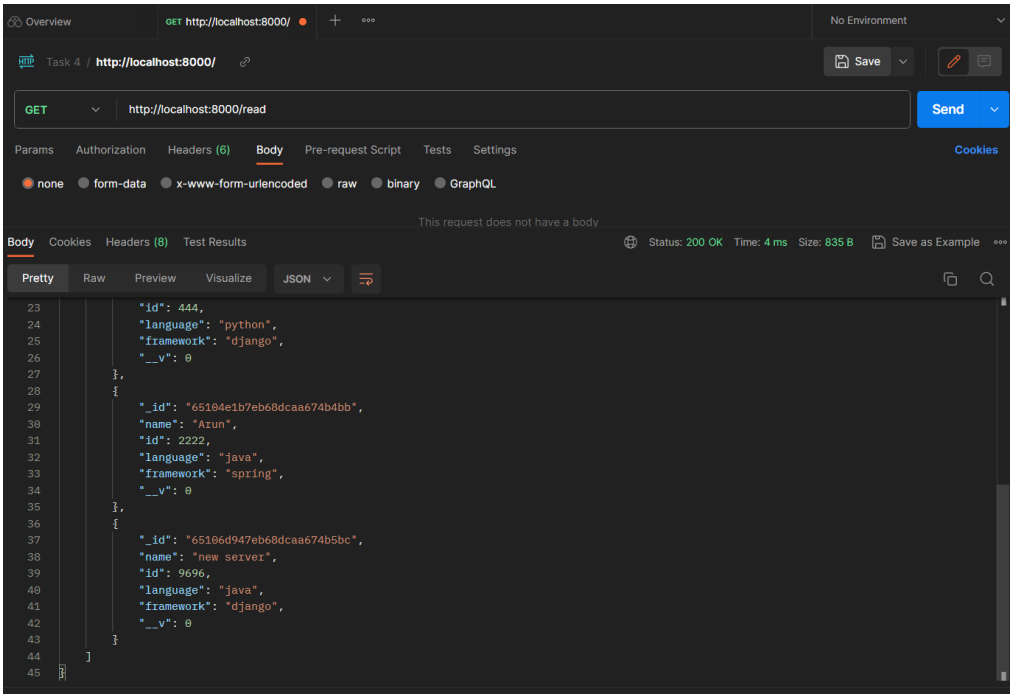
```
router.route('/read').get(controller.readDataControl);

router.route('/read/:name').get(controller.readDataByNameControl);

router.route('/create').post(controller.createDataControl);

router.route('/update/:id').patch(controller.updateDataControl);

router.route('/delete/:id').delete(controller.deleteDataControl);
```

b) Backend testing with postman

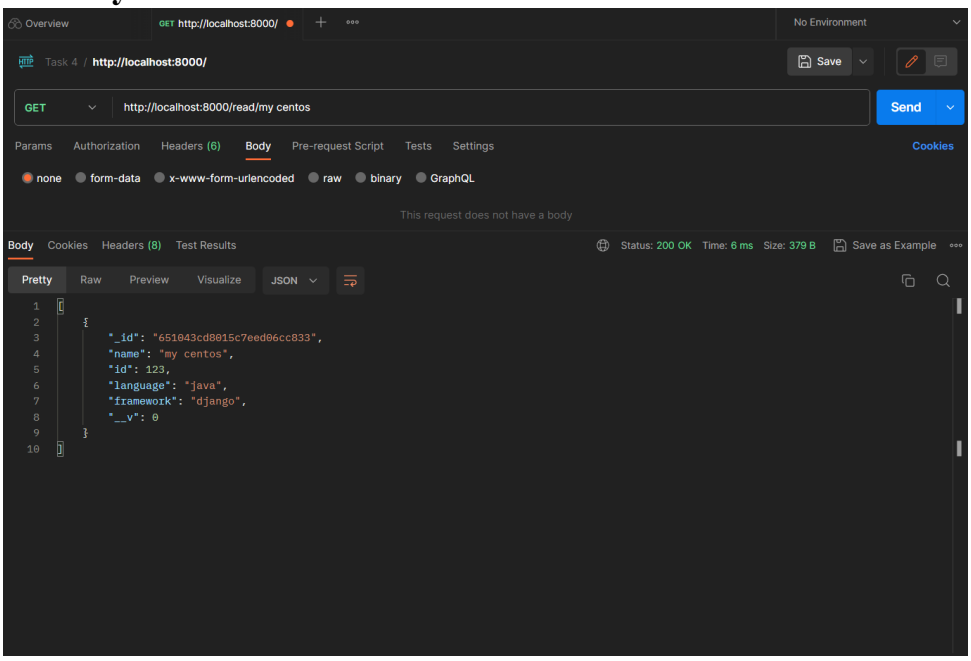
- **POST**



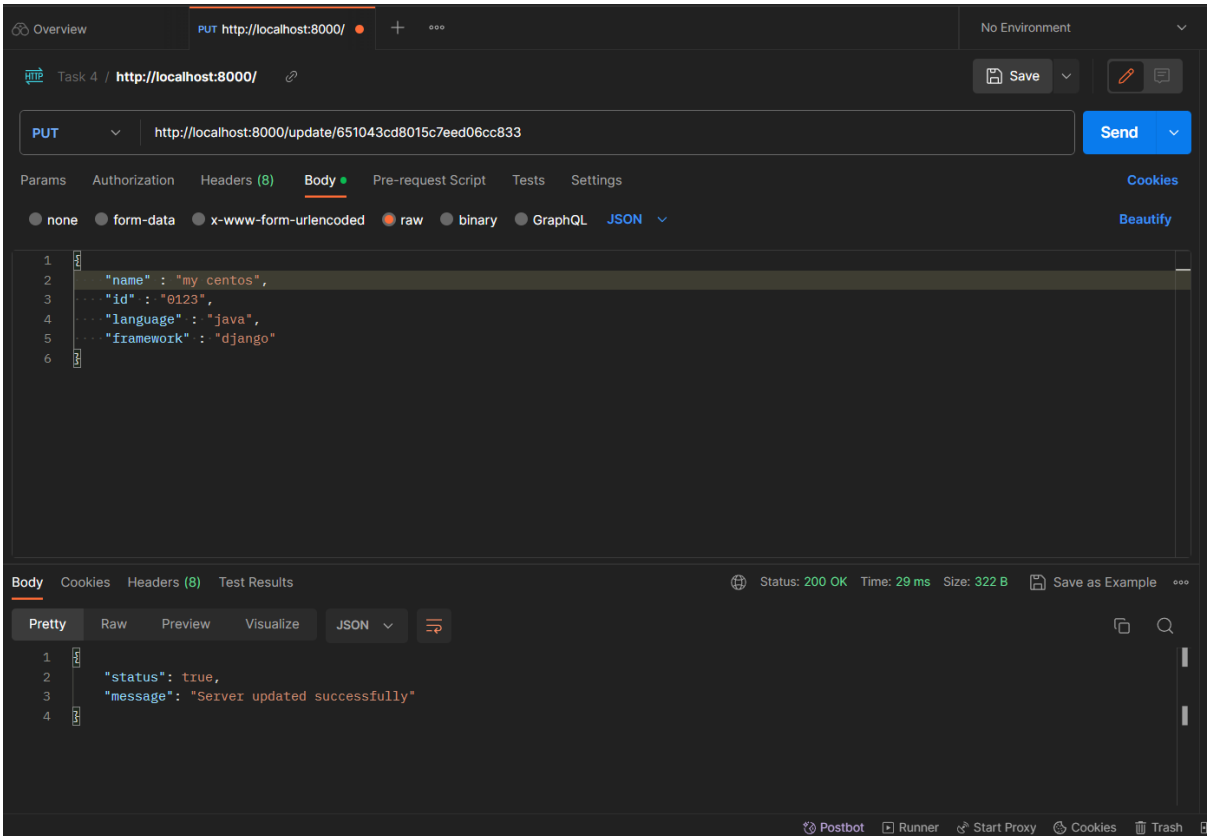
- **GET**



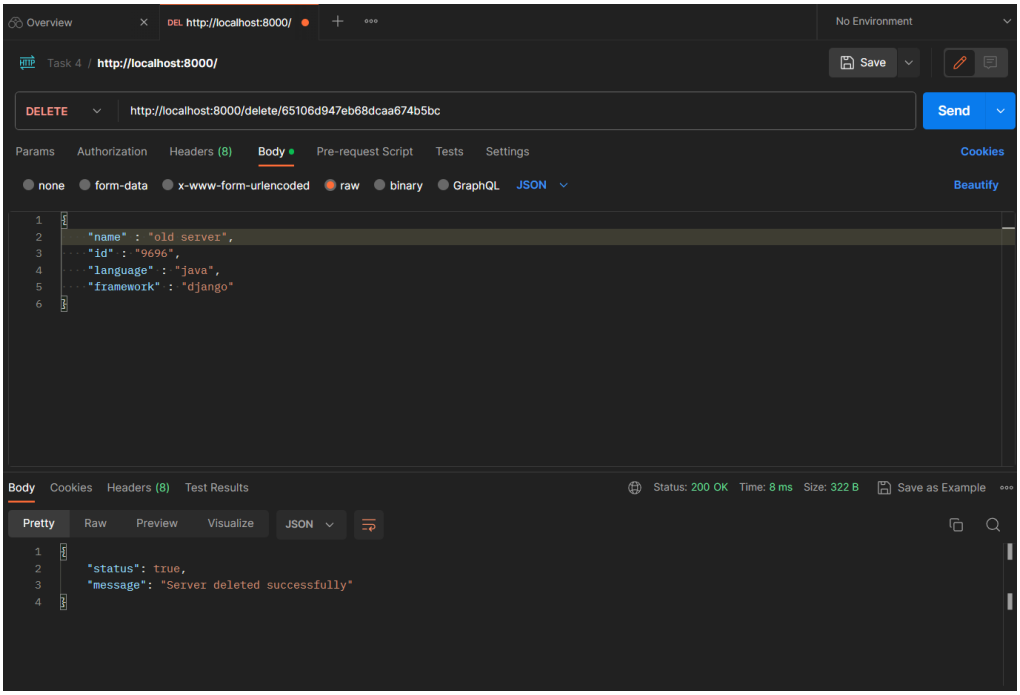
- **GET by name**



- **PUT**


Overview
PUT http://localhost:8000/
No Environment
Task 4 / http://localhost:8000/
Save
PUT
http://localhost:8000/update/651043cd8015c7eed06cc833
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"name" : "my centos",
"id" : "0123",
"language" : "java",
"framework" : "django"
}
Body Cookies Headers (8) Test Results
Status: 200 OK Time: 29 ms Size: 322 B
Save as Example
Pretty Raw Preview Visualize JSON
{
"status": true,
"message": "Server updated successfully"
}
Postbot Runner Start Proxy Cookies Trash

- **DELETE**


Overview
DEL http://localhost:8000/
No Environment
Task 4 / http://localhost:8000/
Save
DELETE
http://localhost:8000/delete/65106d947eb68dcaa674b5bc
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"name" : "old server",
"id" : "9696",
"language" : "java",
"framework" : "django"
}
Body Cookies Headers (8) Test Results
Status: 200 OK Time: 8 ms Size: 322 B
Save as Example
Pretty Raw Preview Visualize JSON
{
"status": true,
"message": "Server deleted successfully"
}

c) Front-end

i. Open command prompt in the respective folder and run the command to create new Angular project.

```
ng new my-first-project
cd my-first-project
ng serve
```

ii. Generate new components ‘add’ and ‘home’.

```
E:\Projects\Task 4\FrontEnd>ng g c add
```

```
E:\Projects\Task 4\FrontEnd>ng g c home
```

iii. Configure paths in app-routing-module

```
const routes: Routes = [
  { path:"", component: HomeComponent },
  { path:"add", component: AddComponent}
];
```

d) Database

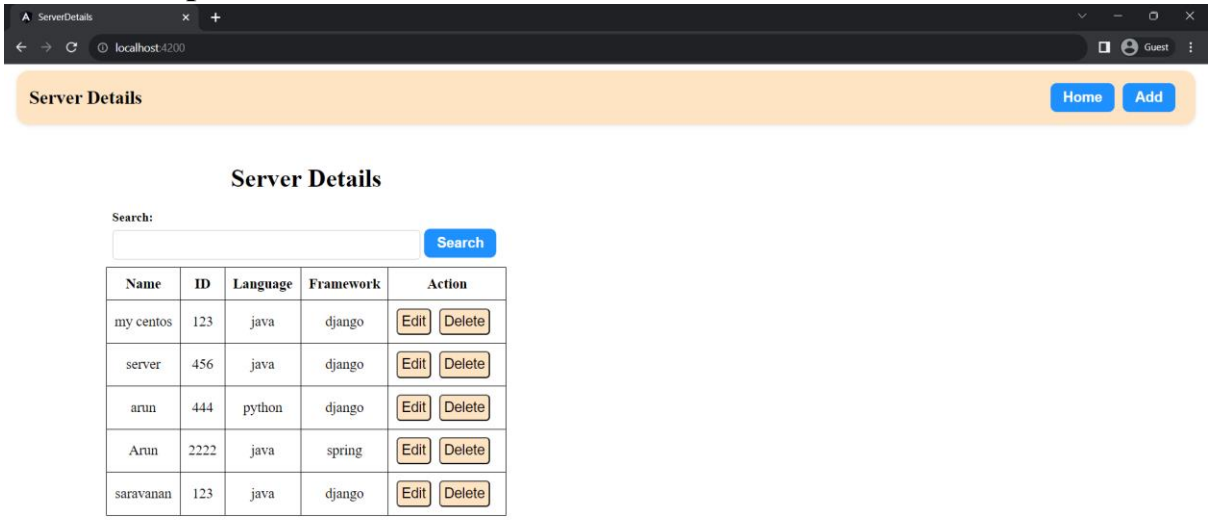
i. Open MongoDB compass and create a collection (project_details). Copy the connection string of the collection created. (Connection string - mongodb://localhost:27017)

ii. Establish database connection with MongoDB.

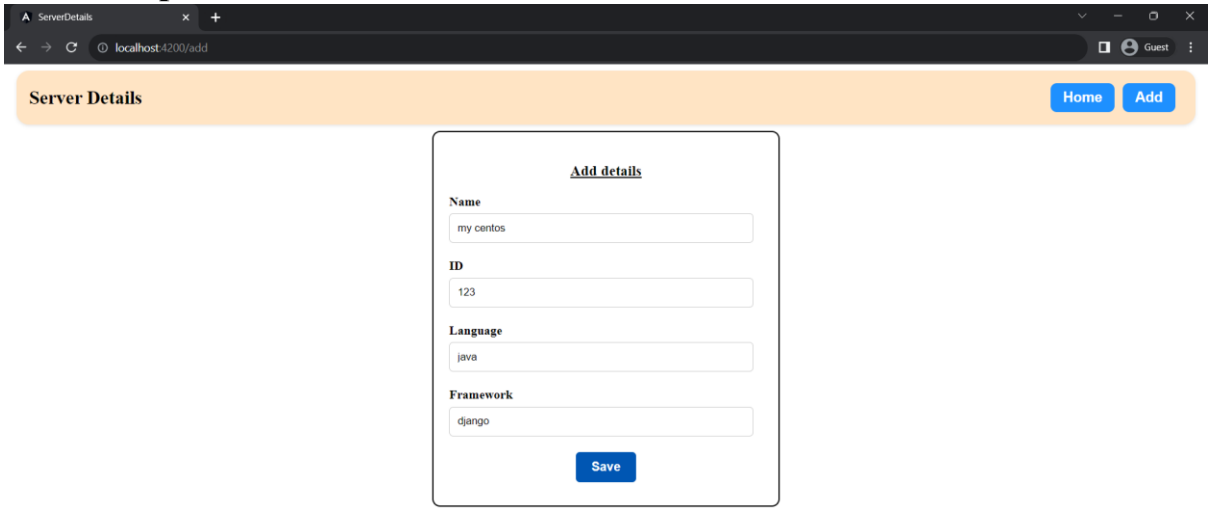
```
mongoose.connect("mongodb://localhost:27017/project_details",{ useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => {
    console.log('Connected to MongoDB');
  })
  .catch(error => {
    console.error('Error connecting with MongoDB', error);
  });
```

Output Screenshots:

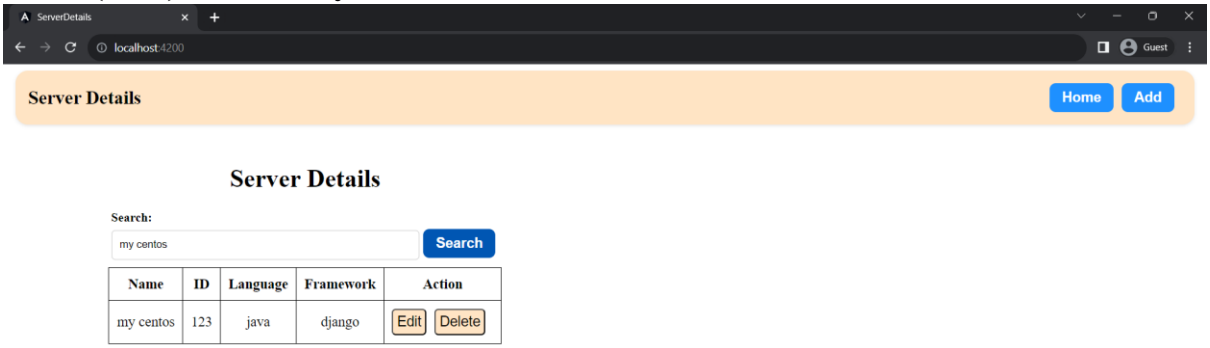
Home Component – GET all servers



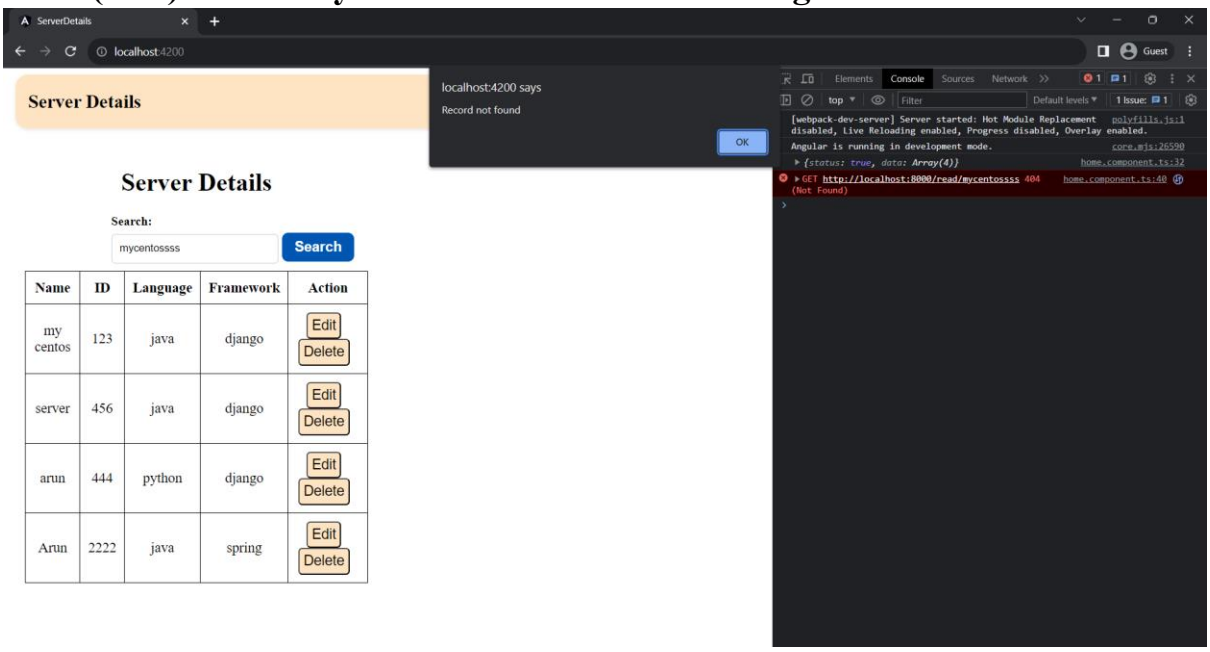
Add Component – PUT a server



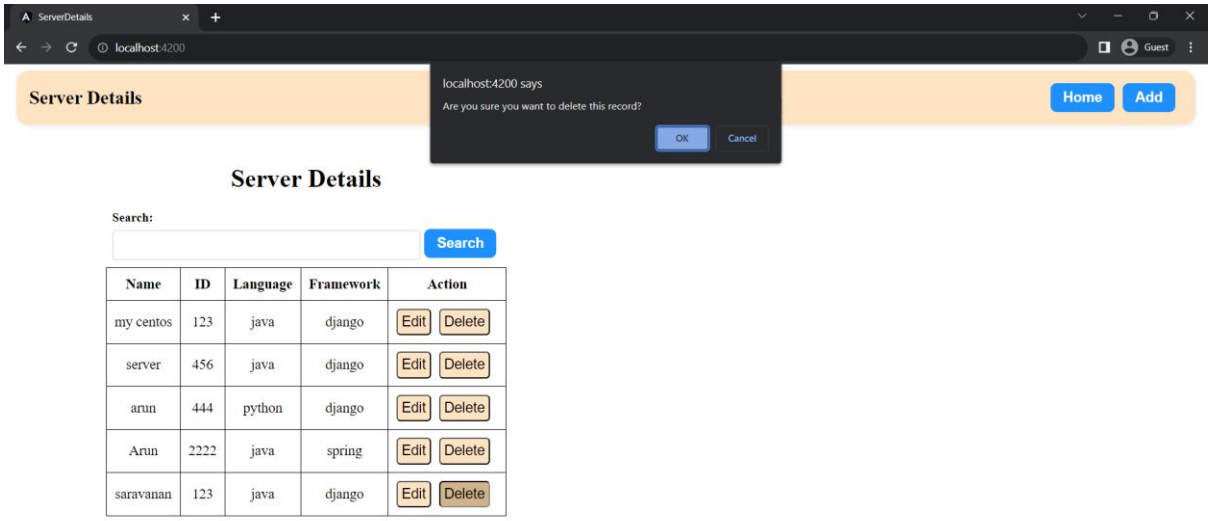
GET (find) servers by name



GET (find) servers by name. Return 404 if nothing is found



DELETE server



Server deleted