**KAIBURR ASSESSMENT**

**NAME : AAMEER ARSHATH A**

**REG NO : 22MCA0011**

**Task 1. Java REST API example.**

**Implement an application in java which provides a REST API with endpoints for searching,**

**creating and deleting “server” objects:**

**● GET servers. Should return all the servers if no parameters are passed. When server id**

**is passed as a parameter - return a single server or 404 if there’s no such a server.**

**● PUT a server. The server object is passed as a json-encoded message body. Here’s an**

**example:**

**{**

**“name”: ”my centos”,**

**“id”: “123”,**

**“language”:”java”,**

**“framework”:”django”**

**}**

**● DELETE a server. The parameter is a server ID.**

**● GET (find) servers by name. The parameter is a string. Must check if a server name**

**contains this string and return one or more servers found. Return 404 if nothing is found.**

**“Server” objects should be stored in MongoDB database.**

**Be sure that you can show how your application responds to requests using postman, curl or**

**any other HTTP client.**

**ADD DEPENDENCIES:**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.mongodb</groupId>

<artifactId>mongodb-driver-sync</artifactId>

</dependency>

</dependencies>

**CREATE A SERVER ENTITY:**

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "servers")

public class Server {

@Id

private String id;

private String name;

private String language;

private String framework;

// Constructors, getters, and setters

}

**CREATE AN SERVER REPOSITORY:**

import org.springframework.data.mongodb.repository.MongoRepository;

import java.util.List;

public interface ServerRepository extends MongoRepository<Server, String> {

List<Server> findByNameContaining(String name);

}

**CREATE AN REST CONTROLLER:**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/servers")

public class ServerController {

@Autowired

private ServerRepository serverRepository;

@GetMapping

public List<Server> getServers(@RequestParam(required = false) String name) {

if (name != null) {

return serverRepository.findByNameContaining(name);

} else {

return serverRepository.findAll();

}

}

@GetMapping("/{id}")

public Server getServerById(@PathVariable String id) {

return serverRepository.findById(id).orElse(null);

}

@PostMapping

public Server createServer(@RequestBody Server server) {

return serverRepository.save(server);

}

@PutMapping("/{id}")

public Server updateServer(@PathVariable String id, @RequestBody Server server) {

server.setId(id);

return serverRepository.save(server);

}

@DeleteMapping("/{id}")

public void deleteServer(@PathVariable String id) {

serverRepository.deleteById(id);

}

}

**CONFIGURE MONGODB:**

spring.data.mongodb.host=localhost

spring.data.mongodb.port=27017

spring.data.mongodb.database=mydb