CSU33012 - Assignment 1 - Microservices and Spring Boot

Version 1. This assignment is worth 15% of your grade.

Conventions

All project **must**

- be version controlled in git
  - on gitlab.scss.tcd.ie
  - in **private** repositories
  - commit incremental changes, do not just commit the final version
- Use the given repository name
- Use the given project name, e.g., `helloworld`
  - as artifactId in Spring Initializr/pom.xml
  - as project name (root folder of the project structure).
    Note that this is different from the repository name.
- Use the namespace `ie.tcd.scss`
  - Use this as group ID in pom.xml and base package in Java
- Use Spring Boot 3.1.4 or newer
  - Stable version of Spring 3. No snapshots or milestones.
- Use Maven as a build system
- Use the Maven standard file structure
- Use Java 17
- Include at least the following dependencies in pom.xml, you might have to add more dependencies depending on the given task.

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-test</artifactId>
   <scope>test</scope>
</dependency>
```

Submission (to be updated)

- Make sure to use a repository on gitlab.scss.tcd.ie with the specified name.
- Commit regular progress, do not just commit the whole project in the end.
- The content of your project should be in the **main** branch.
- More details on the submission to be announced.

Task 1.1

(Weight 2%)

- Repository `helloworld`
- Project name `helloworld`
- Root Package `ie.tcd.scss.helloworld`
- Implement a Spring Boot application which provides a microservice `ie.tcd.scss.helloworld.controller.HelloController` that follows the following specification

Request			**Response**		
Method	**URL**	**Body (example)**	**Status Code**	**Body (example)**	**Notes**
`GET`	`/hello`	`Not provided`	`200`	`Hello, World!!!`	—
`GET`	`/howareyou`	`Not provided`	`200`	`Grand. How are you?`	—

Task 1.2

(Weight 4%)

- Repository `namelist`
- Project name `namelist`
- Root Package `ie.tcd.scss.namelist`
- Implement a Spring Boot application which provides a microservice `ie.tcd.scss.namelist.controller.NamelistController` that follows the specification below.
- For this task, the data should be stored in a simple list of Strings in the controller. There is no need to implement additional services, repositories or similar.
- You will need a separate class for accepting data handed to you in the request body of the during the POST operation (a data transfer object). For instance, something like

```
public static class NameDto {
   private String name;
   // add getter/setter
}
```

Request			**Response**		
Method	**URL**	**Body (example)**	**Status Code**	**Body (example)**	**Notes**
`GET`	`/reset`	`Not provided`	`200`		Resets the list to an empty list.
`POST`	`/names`	`{ "name":"Robin" }`	`201`		**If list does not contain name:** Stores name.
			`200`		**If list does not contain name:** Does not change list.
		`{ "name":"" }`	`403`		Empty names are not allowed and rejected. Does not change list.
`GET`	`/names`	`Not provided`	`200`	`Mary, Parker, Robin`	**If the list is not empty:** List of names that were stored earlier in alphabetical order. Separated by comma, no comma after last name.
			`200`	`(List of names is empty)`	**If the list of names is empty:** Return “(List of names is empty)”

Task 1.3 (Weight 9%)

- Repository `social`
- Project name `social`
- Root Package `ie.tcd.scss.social`
- Implement a Spring Boot application that provides
  - An application
    `ie.tcd.scss.social.SocialApplication`
    which has exactly one `MessageRepository` and `MessageService` (see below).
  - A domain entity (handled as a JPA entity)
    `ie.tcd.scss.social.domain.Message`
    with attributes
    `String messageKey` (which is the ID, but not autogenerated)
    `String title`
    `String body` (up to 500 characters).
    with getters and setters for all attributes
  - A repository
    `ie.tcd.scss.social.repo.MessageRepository`
    that extends CrudRepository and serves entities of type `Message` with key `key`
  - A service
    `ie.tcd.scss.social.service.MessageService`
    that provides a function
    `createMessage(String str)`
    that will generate a test message with
    key = <str>
    title = 5 x <str>
    body = 10 x <str>
    So for instance
    - createMessage("a") will generate a message with the attributes
      `key=a, title=aaaaa, body=aaaaaaaaaa`
    - createMessage("ab") will generate a message with the attributes
      `key=ab, title=abababababab, body=abababababababababab`
  - Then, the Service stores the message in the repository. If the repository already contains a message with the same key then the function does nothing.
- The repository should use the H2 database to store data.
- Initialise the database when the application is started by calling

```
messageService.createMessage("a");
messageService.createMessage("ab");
messageService.createMessage("xyz");
```