Practical session in week 7 - Unit testing with Karma & Jasmine

In this assignment, you will write a very simple unit test for an Angular application.

1. Create a simple Angular project

Create a simple Angular project using angular-cli with a single Login component, similar to the one you did in the second hand-in assignment (i.e. where there is a `LoginComponent` which talks to a `ChatService` dependency). You may also just use your solution to the second hand-in assignment as a base.

2. Write Unit tests for `LoginComponent`

A `.spec.ts` file should exist in the folder for the `LoginComponent`. In there you will write unit tests for the method where the user types something into the username textbox and presses the "Login" button. There are two cases we want to test:

- When the user selects a username which is available, (s)he should be redirected to the "/rooms" route.
- When the user selects a username already in use, (s)he should stay on the same page, and an error message should appear.

Obviously, we cannot reliably test this using the real `ChatService` - because that service relies on an already running instance of a server. This would be hard to set up properly in a testing environment. Instead, we will mock the service, providing a dummy instance which doesn't connect to the server at all, but instead will return hardcoded values. In our case, the mock service will either return `true` (the username is free), or `false` (the username is taken), and we will tell it which case to use. See example in the slides about Angular testing on how to create that mock class.

Otherwise, you should write two `it()` functions, one for each case:

a) when the user selects an available username:

> Ensure that your `mockService` is configured to always return true.
>
> In the `Assert` part, use `expect()` to verify that the `navigate()` function has been called on the router

b) when the user selects an already taken username:

> Ensure that the `mockService` is configured to always return false.
>
> In the `Assert` part, ensure that the `navigate()` method has NOT been called. Also, ensure that the HTML element which displays the error message is visible.

There are many errors you might encounter when running the tests. For instance, the router-outlet element in `AppComponent` might cause problems. Here is one method to fix it:

http://stackoverflow.com/questions/39792753/testing-router-outlet-component-in-angular2.

There will no doubt be other errors, feel free to share with others if you find solutions for such errors.

Handin

Handin your `LoginComponent.spec.ts` file.