Mini project: Hotel booking

Part 1: Unit testing

Background

A hotel has decided to develop a web site, where people can book their hotel rooms. A hotel room can be booked for a period (start date – end date) in the future provided that it is not already booked for one or more days during the desired period. When the customer checks in at the start date, the room should be marked as occupied. If the customer does not check in at the start date, the reservation should be cancelled. When the customer checks out, the room should be marked as free. A customer can cancel or change a reservation before the start date occurs.

It should be easy for the user of the web site to find out whether any rooms are available during a given period. Therefore, the web site should include a calendar-like view which marks all the dates - on a month per month basis - where all rooms are reserved, with red background color. To support this feature, a method which finds and returns the fully occupied dates for a given period, is required.

A minimal solution has already been made. It implements the following two features:

1. A hotel room can be booked for a period (start date – end date) in the future provided it is not already booked for one or more days during the desired period.
2. It should be easy for the user of the web site to find out whether any rooms are available during a given period.

You job is to implement different types of testing, covering the above mentioned two features, during the course.

I have published two partial ASP.NET Core Web API solutions on GitHub, and you can use one of these as a starting point. Both solutions implement the same, but one of them uses asynchronous methods extensively. Here are links to the two partial solutions:

https://github.com/hkeasv/HotelBooking_Clean

https://github.com/hkeasv/HotelBooking_Clean_Async

I recommend that you use the asynchronous version as a starting point, at least if you are already familiar with asynchronous programming in C#. The solutions contain a simple ASP.NET MVC application as well as an ASP.NET Web API. If you are unfamiliar with ASP.NET MVC, then just ignore it, and use the ASP.NET Web API instead. The purpose of the ASP.NET MVC application is just to provide a simple user interface.

You are free to choose another programming language. However, if you do so, you should expect to spend considerably more time to develop your solution. You should then develop the initial solution from scratch and find alternatives to the testing tools taught in this course.

Overall plan

Part 1	Handed out Week 5 Goals: • Formation of teams for further work on the mini project. Please send an email to me (hk@easv.dk) with a list of your team members. • Minimal unit testing of booking feature using data-driven unit testing and a mocking framework. **Presentation: Week 9**
Part 2	Handed out Week 10 Goals: • Test cases for the booking feature have been derived using black-box test techniques. • Functional tests of selected user stories have been implemented using Cucumber. • Automated API tests for the booking feature have been implemented using Postman. **Presentation: Week 14**
Part 3	Handed out Week 15 Goals: • Test cases for the booking feature have been derived using white-box test techniques. **Presentation: Week 18**

General terms and conditions

The mini project is team work. A team size of 2-3 members is recommended.

The mini project is a compulsory assignment. This means that your solution must be approved by the teacher before you can attend exam.

You are not required to hand in your solution for approval. Instead, each team should present their solution for the teacher. During presentations, every team member must actively participate. It will be specified in each part of the mini project what should be presented.

If a team is unable to present during one of the scheduled weeks, it can contact me in good time to have the presentation re-scheduled to one week earlier or later.

Requirements for "Mini project part 1"

There are two goals of this part of the mini project:

1. **Unit testing**: Write unit tests for all business logic inside the *HotelBooking.Core* project. You must demonstrate use of data-driven unit testing as well as use of a mocking framework (e.g. Moq). You should use the basic naming conventions introduced in the course.

2. **Design for testability (optional)**: Improve the design of the minimal implementation to maximize its testability (this may be hard to do).

Terms and conditions for “Mini project part 1”

Each team should demonstrate their solution to this part of the mini project during Week 9. A presentation schedule will be published the week before.

You should explain how you have implemented unit testing of the core business logic of the hotel booking solution. The presentation must include one example of data-driven unit testing as well as the use of a mocking framework. Each presentation can last up to 10 minutes.