PROJECT INFO

Title: *Scale up your app with distributed systems*

Estimated duration: *60 hours (courses + project)*

Target skills:

* *Produce technical and functional documentation for the application*
* *Fix errors reported by the customer on an application*
* *Improve an application as per a customer request*
* *Complete a Unit and Integration test suite to account for changes*

Recommended courses:

* *Scale up your code with Java concurrency (10 hours)*
* *Decouple (refactor) your web architecture (6 hours)*
* *Docker (8 hours)*

PROJECT BRIEF (Student facing)

# Scenario

You are the senior software developer for Future Technologies, a global IT consultancy firm. Chris, your CTO, has just come back from a client meeting with Californian Health, a leading private healthcare provider with a network of hospitals, outpatient centres and clinics.   
  
Californian Health have recently conducted an audit of their web platform and the issues that have been stemming from users and have documented their findings which Chris had stored this into the company’s git account [**here**](https://github.com/OpenClassrooms-Student-Center/.NET_Project_8/blob/master/Supporting%20Document%20-%20SDD.docx).

You now have a good idea of Californian Health’s website and portal.

As a follow up to this, Linda had emailed Chris with a list of current issues which he has forwarded onto you as below.

## Email from Linda

|  |
| --- |
| **From**: [Linda.McCarthy@californianhealth.com](mailto:Linda.McCarthy@californianhealth.com) **Subject**: FW: Recap of our meeting  Hi Chris,  It was good speaking to you earlier today.  I wanted to highlight the current issues facing Californian Health.   1. A recent marketing drive to promote the company had slowed down access to the main site, thereby causing access issues to members who tried to make appointment bookings. 2. The website is not purely static in that it also displays all the consultant names and speciality information from the portal database. 3. Members have also recently started noticing that the consultant booking system has not been functioning well. It slows down often resulting in appointments not being made or the consultant system’s not being updated with the information sent through by users. This has often resulted in angry complaints from users so much so, that they are now threatening to sign out from the hospital’s membership 4. Members have also complained that they are unable to see the correct information on the online calendar displaying consultant availability. It is also not real time, resulting in member’s making bookings on slots already filled 5. Recently, your developers had changed the way the search works to include intelligent searching based on the logged in member’s past visits. When deploying the code, your developers had realised that they had to deploy the entire solution which meant your services had to be offline, including your main website. This was unacceptable as the downtime to your service affected business. You anticipate more changes soon and you would like a solution that doesn’t require the whole system being deployed if there is a change in functionality in one area.   We now require the following from you to help us fix the crucial issues:   1. Have a look at Californian Health’s current architecture and come up with a solution to best host the components so that it reduces stress on the site. Mainly think of whether we can move any of the components to containers so that the application can invoke the services as needed. You want us to document the findings in a technical and functional document. 2. An immediate problem seems to be the unresponsive online calendar / booking system. Come up with a way of ensuring that members do not have to wait endlessly for the system to respond on consultant availability. More importantly, think about how members don’t double book on slots that have been filled already. 3. Refactor the code used to display the consultant calendar. Ensure that the user can have a real time view of the consultant’s availability. You had also wondered whether the user can have an on-screen confirmation of whether the booking is confirmed. 4. Finally, you feel that the test cases written to test the use of the online booking by concurrent users is not working well. You wanted these to rewrite so that it could be tested for a load of 3000 concurrent users at any point.   If you could confirm the above via return email, we will get the work started ASAP!  Thanks & Regards,  Linda |

With the email in your hand, its time to get started!

# Deliverables

* Completed functional and technical documentation with a revised architecture of the system
* Updated codebase on github

# Presentation

You’ll do an oral presentation of your project with an assessor in order to imitate real-life conditions. Your assessor, in this case, will play the role of Linda, who will be doing a code review. You will explain your process for developing and testing the refactored version of Californian Health’s architecture.

The presentation will be structured as follows:

* Explanation of deliverables (15 minutes)
  + Give a short-guided tour of your deliverables.
  + Explain:
    - How did you implement the required changes? Why?
    - What tests did you run? Why? How did you implement them? Explain your choices.
    - What constraints or difficulties did you encounter? How did you address them?
    - What would you have changed or done differently?
* Discussion (10 min)
  + The assessor will ask you questions about your methodology and your deliverables.
  + Be prepared to defend your work, as the validator will challenge your decisions.

The assessor will stop playing the role of the manager 5 minutes before the end of the session so that you can debrief together.

MENTOR GUIDE (mentor- and assessor-facing)

## 👥 Mentoring support notes

Students should find a natural way to divide the Monolithic application into microservices mainly delineated around the following functional boundaries:

Online booking functionality with messaging queue  
Consultant Calendar  
The central web service functionality that deals with consultant search.

Students will need to implement concurrency to solve the performance problem surrounding the online booking of the consultant calendar and the availability of the consultant calendar. These problems are tangled together, as once a consultant’s calendar availability is returned, the online booking needs to be made and the consultant calendar updated immediately. To do this, students may use a message queuing service.

The project contains unit tests that the students will need to use to test the application with 3,000 users. The students will need to modify these unit tests to match their concurrent solutions, but they should preserve the required time limit assertions that only pass the tests if they complete within 15 and 20 minutes.  
  
It is easy for students to edit the unit tests to change the number of users. This is meant to allow them to test 100 users, then 1000, and so on.

## 🕵️‍ Evaluation rubric

|  |
| --- |
| If you are a validator on this project, your role will be to inspect the deliverables in order to validate (or not) whether the student has acquired the skills worked on in the project.  To help you in this work and clarify the expectations, we’ve provided you with this evaluation rubric below. The listed criteria should help reduce friction and differences of opinion between the mentors, validators and the OpenClassrooms Education team.  Even though the criteria have been written out to be as precise, objective and complete as possible, they don’t cover everything. Consider those that have been identified as minimum criteria to validate the associated skill. If you have other criteria in mind or disagree, feel free to add suggestions as comments directly in this document. |

**Complete** = All the elements I asked for, are they included here? (*e.g. I asked for a test report, did they provide it?)*

**Relevant** = Do each of the elements do what I asked? *(e.g. I asked them to test specific things, did they test them? I asked them to do it using X technique, did they do it that way?)*

**Presentable** = Is it organized based on industry standards *(e.g. did they use the industry conventions I specified? Did they export the test report in the correct format?)*

|  |  |
| --- | --- |
| **🎯Skill** | **🔍 Criteria** |
| ***Produce technical and functional documentation for the application***  **Deliverable:** Documentation template (includes architectural diagram and rationale) | Documentation is **complete** when:   * The document is delivered with an Architectural diagram.   Document is **presentable** when:   * The Architectural diagram uses industry standard methods to convey application design. * Open standards such as HTTP/HTTPS, XML and /or JSON is used * Message queues and microservices are used. |
|
|

|  |  |
| --- | --- |
| ***Fix errors reported by the customer on an application***  **Deliverable:** zip file and github links for Complete code base | The modifications to the application are **complete** when:   * There is a zip file and code checked into multiple Github repositories.   The modifications to the application are **relevant** when:   * The application is split into multiple GitHub repositories such as:   + the front-end application to render the code   + the API layer with microservices for individual components for online booking and calendar availability. * The GitHub repositories include working Docker files. * The disparate components of the application (as delineated by the separate Github repositories) communicate with each other without error.   The modifications to the application are **presentable** when:   * The repositories have a granular commit history. * The commit messages are meaningful - describing the code change committed. * The code base has different Github projects for the varied components |
|
|

|  |  |
| --- | --- |
| ***Improve an application as per a customer request***  **Deliverable:** zip file and github links for Complete code base | The modifications to the application are **complete** when:   * There is a zip file and code checked into multiple Github repositories.   The modifications to the application are **relevant** when:   * Microservices for the online booking and consultant calendar are added to componentize the functionalities * Message queues are used for online booking to ensure that concurrent users are able to book without any issues.   The modifications to the application are **presentable** when:   * The code has correct indentation.   + Indentation for if statements   + Indentation for for loops   + Indentation for method and class signatures * Class variables are named with Camel Case. |
|
|

|  |  |
| --- | --- |
| **Complete a Unit and Integration test suite to account for changes**  **Deliverable:** Unit and Integration tests | Tests are **complete** when:   * The build runs unit tests successfully.   Test coverage report is **relevant** when:   * The 3,000 concurrency users test case is created using TaskScheduler |
|
|