CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

SOEN 342 – Sections H and II: Software Requirements and Deployment

Fall 2024

Instructor: Dr. Constantinos Constantinides, *P.Eng.*

Project description

Date posted: 26 September, 2024

INTRODUCTION

In this project you will work in pairs. The project weighs a total of 20% of your overall mark. Please read the entire document very carefully.

HOW TO DO WELL IN THE PROJECT

This exercise is how you go about developing a software system. A software system is called a "system" for a reason. It is a collection of technical artifacts and as a collection it must be consistent. Your design artifacts must align with analysis artifacts and your implementation artifacts must align with design artifacts.

ORGANIZATION

The preliminary iteration (see later) should give you enough time set up your environment.

SYSTEM DESCRIPTION

The system under development would support an organization to offer group or private lessons (of various types, such as yoga, swimming, etc.) to clients.

The organization owns or rents space (gyms, rooms, swimming pools) in various locations in possibly different cities across the province and each location is made available over some given schedule, where a schedule is essentially a sequence of day-time slots. As an example, we may have the following:

> "The EV-Building gym Room 7, in Montreal, is available for Judo classes on Sundays from 12 noon to 3PM, from September 1st to November 30th, 2024."

The length of a time slot is not fixed (and you are free to create schedules as you see fit). For example, a swimming lesson would normally be for half hour, but a Judo class would normally be for one hour.

A given location can only accommodate one lesson at any given time slot on a given day. The same type of lesson can be offered in both modes (private, group). For example, at the same location the organization would offer private swimming lessons and group swimming lessons.

The organization does not have permanent instructors, but it hires seasonal instructors of various specializations. An instructor would register with the system by entering their credentials (name and phone number) and specialization as well as to register their availability in one or possibly several cities.

For example,

> “Grace (514 - …) is a swim instructor, available to work in Montreal and Laval.”

Once registered, an instructor can subsequently take on possibly several offerings that the organization makes available. All offerings are made available to potential instructors, but only those that have been taken by instructors are made available to the public in order to attract clients.

For example,

> “We offer private and group Judo classes in EV-Building on Sundays from 12PM to 3PM from 1.Sep to 30.Nov as follows:
>
> 12:00 – 13:00. Group. Instructor: …
>
> 13:00 – 14:00. Group. …
>
> 14:00 – 14:30. Private. …
>
> 14:30 – 15:00. Private. …
>
> We offer swimming classes at …
>
> …”

The public can view the organization’s available offerings. However, in order to make a booking, one must first become a client, i.e. they must register with the system and only then they may proceed to select one or possibly several offerings. For each available offering of interest, a client may proceed to make a booking. Once a booking is made, then the corresponding offering continues to appear publicly, but it is annotated as non-available.

For example:

> "EV-Bld. 14:00 – 14:30. Not available."

Booking details are only available to the corresponding clients (and to the administrator of the system) who can view their bookings, or cancel a booking.

For someone who is underage (i.e. less than 18 yrs old), they would need an adult who will be acting as their legal guardian who can handle their bookings.

For example,

> "James Russo registered with the system and made a booking for his daughter Lisa (who is 14 yrs old) … "

The system has one administrator who has full Read/Write access to all records. The administrator enters the organization's offerings, but the administrator does not register instructors or clients. An administrator, however, may delete an account of an instructor or a client.

We have two types of Actors: A 'Writer' is one who modifies state. A 'Reader' is one who accesses but does not modify state. The system supports concurrency with the following restrictions: Writers operate in self-exclusion. Writers and Readers operate in mutual exclusion. The system can allow multiple Readers.

The system must be developed iteratively as follows:

	DATES	REQUIREMENTS	ARTIFACTS TO PRODUCE/ DELIVERABLES
1	**27.09 – 04.10**	• Set up team. • Set up GitHub platform. • Send email to Instructor with names of your team. Include Course Number and Section in Subject, and names with id's in the body. • Communicate credentials to your marker (Instructions will be posted). • Construct a UML Use Case Diagram.	
2	**04.10 – 18.10**	**USE CASE 1:** PROCESS OFFERINGS. (Actors: Administrator, Instructor) Organization (through the Administrator) makes offerings available; Instructors select lessons; Public can view offerings.	• UML domain model [+ package diagram] • System sequence diagram(s) to capture success and failure scenarios. • Identification of system operations and operation contracts. • UML interaction diagrams. • UML class diagram. • Implementation.
3	**18.10 – 01.11**	**USE CASE 2**: PROCESS BOOKINGS. (Actor: Client)	Add new / refine existing artifacts.
4	**01.11 – 15.11**	• Add Persistence, and • Formalisms (See 'Formal Specifications').	• Relational data model. • Add new / refine existing artifacts. • OCL expressions. • 5 min video demonstrating entire system functionality.

FORMAL SPECIFICATIONS

Certain requirements cannot be fully captured by the UML. Consider the following requirements which you must specify in the *Object Constraint Language*.

1. *"Offerings are unique. In other words, multiple offerings on the same day and time slot must be offered at a different location."*

2. *"Any client who is underage must necessarily be accompanied by an adult who acts as their guardian."*

3. *"The city associated with an offering must be one the city's that the instructor has indicated in their availabilities."*

4. *"A client does not have multiple bookings on the same day and time slot."* (for simplicity we consider only identical day and time slots, even though in reality a booking on Monday 3pm – 4pm and another also on Monday 3:30pm – 4:30pm should not be acceptable.)

TOOLS AND TECHNOLOGIES

- Platform: The core architecture of the system must be object-oriented. Feel free to use any programming language or combination of languages.
- Framework(s): You may only use a framework to handle persistence and concurrency (even though you will still need to produce a relational data model).
- User interface: Feel free to create *any* interface you see fit. You will not be assessed on this component.
- GitHub: In your GitHub account, please contain a README file that partners' names and their id's, email information as well as your course section.

Feel free to use any UML drawing tool. Some popular tools are the following:

Smartdraw. URL: https://www.smartdraw.com/uml-diagram/uml-diagram-tool.htm

Lucidchart. URL: https://www.lucidchart.com/pages/

Visual Paradigm. URL: https://online.visual-paradigm.com/diagrams/features/uml-tool/

Canva. URL: https://www.canva.com/graphs/uml-diagrams/

draw.io. URL: https://app.diagrams.net/

You can find more resources here:

Top online UML modeling tools (also including web-based tools for ER and BPMN diagrams).
URL: https://modeling-languages.com/web-based-modeling-tools-uml-er-bpmn/

20+ JavaScript libraries to draw your own diagrams (2024 edition).
URL: https://modeling-languages.com/javascript-drawing-libraries-diagrams/

From Text to Models: A Comprehensive Guide to Textual Modeling and Diagrams as Code Tools in 2024.
URL: https://modeling-languages.com/text-uml-tools-complete-list/

ASSESSMENT

Note that even though this is a team project, marking is not assigned collectively. Based on the criteria described in this document, individual penalties will result in different partners possibly receiving a different mark. As this is an academic exercise (as opposed to an industrial assignment) where the objective is to learn, you are both expected to participate in all activities (requirements analysis, design, implementation, testing) on a relatively equal weight. Failure to do so will result in penalties.

We will monitor your GitHub account throughout the process. Please note that the entire project is locked only at the end of the fourth (4^{th}) iteration, where we will obtain the final image of your workspace. This implies that you are free to modify artifacts developed prior to the final deadline of 15.11. Please do not submit either a hard copy or an electronic copy of your project.

For both partners of each team we will monitor the following: a) number of commits, b) number of assigned tasks, c) the total number of tasks, and d) the degree of importance (of committed tasks), where the latter is defined as follows:

0: Unimportant commits, e.g. comments.
1: Minor commits, e.g. changing the names of functions or variables.
2: Important commits, e.g. adding new functions into classes.
3: Very important commits, e.g. such as adding a class.

In other words, the notion of "relatively equal weight" of workload does not exclusively depend on the number of your commits, but on all four factors described above.

CREATING A VIDEO

During iteration 4 and once you are ready to present your project, you must create and post a video (max duration: 5 mins), demonstrating the functionality of your system.

ISSUES

You must notify me once you encounter an issue that you cannot easily resolve. Accommodating a partner who is not contributing their weight to the project will result in a penalty to both of you.

END OF PROJECT DESCRIPTION
