CSC2063: Group Coursework (Design Part)

Service-Oriented System for Student Information Management

Your brief is to deliver the service-oriented design of a software system that stores and manages information related to the students and the academic staff members of a university school. The suggested work plan for your project, the submission details, the due date of your project, and the technical description of the software system that you need to design are specified in the sections of this document.

Suggested Work Plan

Please note that the first/design part of the coursework covers the material that was taught in the first three weeks of the module. A suggested work plan includes the following steps:

1. Design/develop a skeleton of the Java classes of the system, along with their relationships based on the system description below (**Week 4**)
2. Identify candidate web services that contain the Java classes of your system. You can manually identify your web services or you can identify your Web services by using the algorithm presented in the corresponding lecture here. If you apply the algorithm, we can give as input to the algorithm only one service that contains all the Java classes (**Week 5-6**)
3. Specify in XML format the description of the Web services you identified and submit it via Canvas (**Weeks 6**).

Identified Services

The services that you will identify must have two attributes based on the corresponding lecture about service identification that is available here: (i) the services should have the highest possible average cohesion vale (higher than the cut-off point - we assume the cut-off point in this project of the lowest acceptable cohesion value is 0.8); (ii) there should not exist services that use classes of each other.

Q&A Sessions

- You can ask your questions/get help on your project by attending the in-person practical sessions on Tuesdays (location: CSB/0G/028), where you can ask for having 1-to-1 meetings during the practical sessions with the student support officer, Mr Leo McHugh, l.mchugh@qub.ac.uk (Tuesdays, 11-1pm).
- You can join the online meetings of my weekly office-hours to possibly ask your questions about the coursework (Thursdays, 3-4.30pm).

Canvas Submission and Due Date (Details/Restrictions)

Each group member should submit via Canvas at this page the following file:

- The XML file of the description of the services of your system that
    - must describe all the services of the system (not the services that one group member identified)
    - must be named by applying the following pattern (tokens separated by underscore):

teamNumber_StudentID_StudentName.xml
E.g., team12_4000000_TomLeeMcallen.xml

- must follow the syntax of the XML service description described and exemplified in this lecture. You can manually create the content of the XML file if you prefer. In case you want to automatically generate the file of your service description, you can use the generator that is available here. You can download from this link the "ServiceDescriptionGenerator.jar" file. To execute the generator, then you need to use a terminal (in Linux) or a PowerShell (in Windows) for executing the .jar file as follows: "java -jar ServiceDescriptionGenerator.jar". If your operating system is windows, you can further download the "ServiceDescriptionGenerator.bat" file from the link above that facilitates the execution of the jar file. In particular, you can just double click on the .bat file to execute the generator. In both cases of operating systems, the generator will guide you to insert the information about your service description[1]. Finally, the generator will create the .xml file and will store it in the folder where you downloaded the generator. The ".xml" file is the only file you need to upload on Canvas.

- Submission deadline: 20 February 2022
- Feedback release: around 2 weeks after the submission deadline.

Note: The mark of the first/design part of the coursework is an individual mark for each student. To mark the design part of the coursework, we will compare your work against all the requirements that are specified in the rubric of the first part of the coursework that is available here. To this end, you should deliver a file that meets all the requirements that are specified in the rubric even if you have chosen to work individually or your group has ended up having less than three members.

System Description

Functional Requirements

The system manages information about academic staff members and students. In detail, the system manages the following information:

- *Role*: a role has one of the following predefined values: ACADEMIC_STAFF_MEMBER or STUDENT
- *Students*: an array of students is maintained (max array length: 1000)
- *Student*: the information that is maintained for each student is the following: an ID (an integer number) and an array of modules for students (max array length: 20)
- *Module*: a module is characterized by a code
- *Module code*: a module code can equal one of the following predefined values: CSC1022, CSC1023, CSC1024, CSC1025, CSC1026, CSC1027, CSC1028, CSC1029, CSC1030, CSC1031
- *Academic staff members*: an array of staff members is maintained (max array length: 100)

[1] Please note the generator does not make logical checks on whether the paths of the relationships (e.g., the service name in a relationship is an existing service name) or the names of the services, packages, and classes are correct names in your service description. The generator only checks whether the syntax of your file follows the required XML syntax.

- *Academic staff member*: an academic staff member is characterized by the following information: an ID and an array of modules for academic staff members (max array length: 2).

A requirement for the information that is described above is that each bullet point should correspond to a separated object-oriented (Java) class. Moreover, the relationships between the classes are specified in the points above. You sound not include extra classes or relationships between the classes.

When the system starts up, the CLI of the system interacts with the Controller of the system to retrieve or insert information about students or academic staff members as follows.

- The CLI can use the following information: the role of an end-user, the ID of an end-user, the code of a module, and the Controller of the system.
- The Controller can use the following information: the role of an end-user, the ID of an end-user, the code of a module, the array of academic staff members, and the array of students.

Non-Functional Requirements

1. *User interface of the system*: An end-user cannot directly interact with the system. In other words, the system should not use the standard input (e.g., Scanner Java class) to provide information to the system. All the input information to the system should be hard-coded inside the “main” function of the system. The system includes a single “main” function for all the end-user’s roles. The system does not provide a graphical user interface.
2. *Storing data*: We assume for the first/design part of the coursework that the system does not store information about students and academic staff members.
3. *Service interfaces*: You don’t need to define Java/Web interface for the services that you will identify. Moreover, you don’t need to apply the algorithm for (re-)identifying multiple interfaces for your services. The first part of the coursework focuses on identifying the Java classes of the services of your system.
4. *Packages of services*: You can assume that each service contains only one package. Moreover, you can assume that the skeleton of the monolithic version of the system contains only one package too.