CSCI 4145 / 5409 – Summer 2020

Assignment 6 - Team Assignment

2020-06-30: Amendments

- DB table *Search* (of company Z) does not have the *userId* attribute as user is not authenticated until ordering
- Specs are silent on the UI for the tables *Search* and *JobParts* (for the company Z) and *partOrders* table on each of the companies X and Y. UI that would satisfy the basic requirements includes a search given the name of the job; however, information retrieved should be displayed in an ordered fashion (ordered first by the *jobName*, within a job next ordered by *userId*, and then ordered by *partId*).

2020-06-29 -- Please note the following clarifications and amendments:

- DB table *Jobs (jobName, jobName, partId)* is such that *partId* is a part of primary key
- On page 3, paragraph that starts with "Another company, Z, prepares a web-application that offers", the first two lines should have the word part replaced with the word job so that it would be:
  "Another company, Z, prepares a web-application that offers the client users to select a particular job (either by selecting a job name from a list of job-names (obtained from the company X) or by doing a search for a **job** with a particular name"
- Clarifications on the assignment were also posted on the Brightspace discussion board *FAQs* and Assignment Questions. However, as these discussion boards are inaccessible at the point of writing these amendments, they were also posted on the module *Assignments* (on Brightspace)

This is the first of two assignments involving teamwork. As a team you will be required to utilize agile software development method based on sprints and scrums as described in The Scrum Guide™, to which you were introduced already. Furthermore, you are also required to use the following collaborative software and to keep minutes of your meetings:

- FCS *GitLab* to be used for software development.
- *MS Teams* software for collaboration and coordination.
- *Trello* software to keep track of tasks and their progress – regular snapshots of your Trello need to be taken to show progress in your software development and also to record who-did-what. For that purpose, each task must include estimated hours and once the task moves through development sprints, also actual hours and who performed the task.
- You must keep minutes of your meetings. Amongst other things, your minutes must contain date/time, meeting software used (zoom, MS Teams, …), and who was present. Your minutes must include items discussed and decisions made (all brief and to the point). Your minutes need to be stored and available on Slack in a dedicated Slack channel. Keep in mind that your

Recall that at the end of this assignment and the course, you will have to fill out forms in which you provide information on various aspects of your teamwork, information that will also include major tasks you performed and tasks that other team members performed – this information should be consistent with information contained in your Gitlab, Slack, Trello board, and minutes of meetings – teamwork is a substantial part of your evaluation.

Additionally, you will need to provide a document, *Summary* (that is, summary of your activities for evaluation purposes), that provides a narrative about your decisions, plans, and activities in completing your assignment. The narrative will be based on and supported by your minutes of meetings and content of your repositories -- in your code repo, MS Teams Chat and Files, and Trello boards (at particular points in time). Collectively, these repositories and the minutes of the meetings will be referred to as the ***C**ollaboration **I**nformation (**CI**)*. The intention/objectives of the document is to provide guidance to your evaluator/marker in reviewing your activities and contributions as they appear in your CI. Although CI contains data on what was done, when, and why, it may lack some information that was used to determine your actions and decisions. Furthermore, as the CI contains data/information that is related but stored in different repositories, forming the whole picture, of how your team functioned and tackled various issues that arose and made necessary adjustments, is difficult to form while viewing the data in the various CI repos – and that is the job your *Summary* document. Your summary should be at a sufficient level of abstraction to highlight the major decisions and the reasons why – that way your evaluator will not get "bogged down" in details in forming the overall picture of your team's functioning.

PART A – TEAM CHARTER

As a team, you will depend on each other to do her/his assigned work, to be present for meetings, to work on tasks, etc. However, things will go wrong. How are you going to react when person X doesn't get their work done? Or they don't come to class/meetings? What happens when person X has an argument with person Y?

In any group setting, events like these are bound to occur. You and your team must be aware of this and agree on ground rules from the beginning. Trying to handle problems when they occur will only increase the severity of the problem. To be prepared for such events, you will be required to create your team/group charter that is based on the team charter exercise from Marchewka (2015)[1], pp. 21-22. Do not take this task lightly. The information in this charter document will be used to settle disputes and/or problems as they arise.

Create a professional-looking document that includes the following:

- Team Name -- Come up with a name for your team to give yourselves an identity.

- Team Members -- Provide the following information:

  - *Contact Information* -- List the names of your team members, their phone numbers and Dalhousie email addresses.
  - *Skills and Knowledge Inventory* -- List the specific knowledge and/or skills each team member can contribute to the project. For example, specific technical knowledge, communication skills, leadership skills, etc.
  - *Roles and Responsibilities* -- Based upon your team's skills inventory, define some initial roles and responsibilities for each member of your team. These roles may change over the life of the project.

- Agreed Upon Meeting Times -- Compare schedules and choose times your group can meet to work on the project outside of class time. Of course, in our current COVID situation, your meetings will be online using MS Teams.

- Team Communication -- Decide how the members of your team will communicate and share information using MS Teams. How will your MS Teams communication and data storage be used? Be specific, for instance, state where your minutes will be stored and who will produce them.

- Team Rules and Expectations -- You probably have experience working in a team before. Share some of those experiences and discuss whether those experiences were positive or negative. Based upon your discussion, define:

  - *Team Goal* -- What do you want to achieve as a team? For example, do you want provide your instructor with work that meets the minimum specifications and get a very good grade or are you all willing to go above and beyond in creating the deliverable in order for your work be excellent?
  - *Set of Rules and Expectations* -- What are the team's rules? For example, how will the team make decisions? How will the team resolve conflicts? What happens if someone misses a meeting? Two

[1] Marchewka, J. (2015). Information technology project management (5th ed.). Hoboken, NJ: John Wiley & Sons you need to develop your Team Charter

meetings? Three? How will the team deal with someone who does not contribute equally? With someone who tries to do everything themselves because they "can do it better themselves" (and don't really trust their teammates)?
    - *Charter Changes* -- How are you as a group going to make changes to this charter if you think it is needed later on.

- Signatures – In normal circumstances, each member of the team would sign the team charter to indicate that they have read, but more importantly understood, and agreed to the rules and expectations of the team. However, in the current situation, how will you obtain confirmation from all team members that they read and agreed to the final version of the charter? A reasonable way is to obtain an email from each member that states that they read and agreed to the terms of the charter.

PART B

In this assignment, you need to provide management of information about jobs, as in previous assignments; however, you also need to provide for management of information on parts and to provide a mechanism for finding/matching parts with jobs. Consider the following the tables *Jobs* and *Parts,* where the table *Jobs* is as in previous assignments:

- *Jobs (jobName ... String; partId ... integer; qty ... integer)*, where
    - *jobName* … string, primary key
    - *partId* … integer/number, ~~foreign key,~~ **primary key**
    - *qty* … is the quantity of parts required for the job

  As an example, triplet ("Job1", 3, 55) represents information that the job, with the name "Job1", requires a quantity of 55 of parts with the part number being 3. Note that the combination of *jobName* and *partId* is unique in that there are no two triplets with the same combination of *jobName* and *partId* (i.e., in a DB table, the composite attribute (*jobName, partId*) would be the primary key). Of course, a job could require different quantities of parts that have different part-numbers.

- *Parts (partId ... integer; partName ... string; qoh ... intetger)*, where
    - *partId (primary key)* and *partName* … are self-explanatory
    - *qoh* … is quantity-on-hand of parts (number of parts in stock)

The table *Jobs* is managed by the company X, while the table *Parts* is managed by company Y. Providing management of parts information implies provision for the CRUD operations on the *Parts* table. Of course, web browser-based UI will be required for this purpose plus the CRUD operations should be supported by a web service with endpoints similar to those you created for the management of jobs in the previous assignments. Information about parts, managed by the company Y, is similar to that of company X for jobs, namely, there is UI, for CRU[2] operations on parts (CRUD w/o Delete and wherein part number is not editable), that invokes the web services with appropriate endpoints similar to those for the web services for jobs.

Another company, Z, prepares a web-application that offers the client users to select a particular job (either by selecting a job name from a list of job-names (obtained from the company X) or by doing a search for a **job** with a particular name supplied by the user. If the job is selected/found, then the web app (web service endpoints) for the company Y will be queried for different parts (and their quantities) required by the job – once obtained, this information is shown to the user. The user is given an option to obtain/get all parts for that job. However, there is a constraint that must be enforced in that a

[2] CRU operations are CRUD operations that exclude the Delete operation and in which the Update operation does not change the part number.

client may successfully order parts for a particular job only once. If the user selects to order/get all parts for a job, the user will be authenticated first, and then the user's order is filled (of course, only if the parts are available) – quantities of parts, for that filled order, are subtracted from the corresponding parts in the table *Parts*. The user is informed of the results (filled successfully or there was a failure of some kind). Each of the three companies needs to keep track of activities in the following way:

Companies X and Y: Besides providing for the CRUD operations as a webservice and the frontend for those operations, a company also needs to keep track of each successful order for a job by a client (identified by user ID) in the following table:

PartOrders (partId, userId, jobName, qty), where

partId	… integer/number, primary key
jobName	… string, primary key
userId	… ? , primary key
qty	… integer/number

Thus, each of the companies X and Y will need to be informed by the company Z of a successful order – thus the companies X and Y will need to provide:

- Web service (endpoint) for receiving the job-parts order information (from Z)
- Frontend for the user to query about the information on the successful job (search on job name with information displayed in a sorted order, first by *jobName*, then by *userId*, and then by *partId*

Company Z: *JobParts (partId, userId, jobName, qty, date, time, result)*, where

partId	… integer/number, primary key
jobName	… string, primary key
userId	… ? , primary key
qty	… integer/number
date	… ?
time	… ?
result	... indicates the success/failure of making an order

Search (~~userId,~~ jobName, date, time) … stores information about searches
… UI to provide search by *jobName*

Of course, your team must meet and review the requirements, but clearly there will be three major tasks, each with sub-tasks:

- Management of the *Jobs* table (including UI), such that the user of the company X can perform CRUD operations on the table (and thus there will be a web service that has endpoints that support the CRUD operations) and also any operation(s) required for interacting with the company Z operations, such as receiving an information about a successful order for parts for a job.
- Management of the *Parts* table, (including UI) such that the user of the company Y can perform CRUD operations on the table (and thus there will be a web service that has endpoints that support the CRUD) and also any operation(s) required for interacting with the company Z operations, such as receiving an information about a successful order for parts for a job.
- Management of a jobs-parts request for a user to find parts for a selected job that updates the tables *Jobs* and *Parts.*

Of course, there is an implied constraint that the company Z does not have direct access to the tables *Jobs* and *Parts* and hence must interact with the software of companies X and Y. Similarly, companies X and Y have their separate tables to keep track their info.

Additional constraints:

- You need to use at least two different clouds, e.g., one for the company X and the other for companies Y and Z
- Software for each company must be in a unique student account
- Your deployment needs to use containers
- Your software and DBs need to be on clouds (they need not be managed DBs).

Simplifying assumptions:

- User authentication may be by simple username and password stored in a table. You need not provide CRUD operations to manage the user information.
- You may use one DB to store all the tables under the constraint that any table will be accessed by one company only (owner of the table).

Submission Requirements

Note that the submission deadlines are posted elsewhere. Any file that you submit should have a name:

"A6X-GN-Y.eee" where

- N … is a two-digit ID/number of your group/team
- X … is the letter identifying the assignment part (one of A, B, …)
- Y … is a brief (max 8 letters) description of what is contained in the file, e.g., "compX", or "jobs", or "Parts", or "JobPart", or …
- eee … is a file extension

PART A

Submit a PDF copy of your charter document on Brightspace.

PART B

Content ... Description of what is submitted (in which files that you include as a part of your submission).
- It should also include, in a separate section, description of any deficiencies related to the assignment's functional requirements.
- Optionally, in a separate section, brief description of what you did in excess of the basic requirements. Use

this to highlight what you did particularly well or in excess of the requirements. In particular, if you are proud of your work and feel it deserves more than satisfying requirements (B+), highlight here why.

Summary ... PDF document with content as described above

Minutes of Meetings … Document that summarizes information contained in your minutes of meetings. The summary contains, for each meeting, date/time, and major items discussed

Trello boards … Snapshots of the content of your Trello boards at major points in your development effort

Software organization … high level overview with references to your code repository, including DB(s) used

DB Scripts … Script files containing data with which you initially seeded your DB

Access Information … Contains information on how to access your repositories (note that the document should not contain any passwords … those should be communicated privately, if need be). Your TA needs to have access to your repositories

A6-Tasks-Form-Team.pdf ... Download the form *Tasks-Form-Team.docx* (it is attached to the assignment box), fill it out, produce and submit a pdf version that should be called: *A6-TasksForm-Team-GXX.pdf,* where XX is your group number. The form is filled out by the team and submitted on behalf of the team.

A6-Self-Peer-Form-Ind.pdf … ... Each member needs to download the form *Self-Peer-Ind-YourName.docx* (it is attached to the assignment box), fill it out, produce and submit a pdf version that should be called: *A6-Self-Peer-Ind-XXXX.pdf,* where XXXX is your last name as it appears on Brightspace. The form is filled out and submitted individually by each team member in the assignment A6-Self-Peer-Forms.

Optional Items

Video ... Reference to a video that can be viewed by your evaluator to be introduced to your software when it executes – the intention is to provide your evaluator with an introduction to the software UI for each of the three companies. The video is not intended to be used as a proof that your software works, i.e., it is not intended to show the results of all possible cases that can be reached through UI.

…