Dalhousie University, Faculty of Computer Science

CSCI 4145 - Cloud Computing – Winter 2018

Assigned: March 12, 2018
Demo: April 10, 2018, at the scheduled time slot.
Final submission: April 10, 2018, 23:59

TEAM PROJECT

Workflow Coordination with Security Features

In the project, you can work in groups of two or three people. If you want to work alone, although it is not preferred, it is OK but the requirements will not be reduced for those you will work alone. Similarly, the requirements are the same for 2 or 3 people groups. You have to form your groups by March 19, 2018, noon and need to inform Dr. Levi and your TA Yichuan Xu via email (CCed to all group members).

In the project, you will further elaborate on Assignment 4 with some extra workflow requirements and also you will have some additional security features. The details are below.

Consider the following scenario. A customer, called *Bob*, is considering purchasing a house and is applying for a mortgage from a mortgage broker *MBR*. The company has a web portal that *Bob* enters and fills out appropriate information on the form. He receives a confirmation that all information *Bob* filled out on the form was received but also let *Bob* know that to obtain a mortgage he needs to get confirmation from his employer *EMP* about his employment, confirmation from an insurance company *INSinc* that the house is insurable and for how much, and from the bank *BigBank*, information about his banking. Fortunately, all is automated, and *Bob* simply uses the respective portals of organizations to ask them to provide appropriate information. To obtain insurance, he also needs to contact his real estate broker *RE* who will use an appraiser to appraise the property and submit the information to the insurance company. Here is the scenario describing the work flow using very simplified information that is exchanged.

Bob enters the *MBR*'s portal and on a form requesting a mortgage he enters his name, mortgage value of $*X*, and identifies the house in which he is interested by the MlsID (id that is recognized by the various actors). Once the information is entered on the form, the form invokes the *MBR*'s web service to store the supplied information in their repository and wait for the rest of the information. *Bob* also receives a mortgage application identifier MortID.

Bob then enters the portal of his employer *EMP*. He is first authenticated and then selects an appropriate form to request that his employment information be provided to the company *MBR* by using the *MBR*'s web service while using his mortgage application MortID. Once he submits his request, the form invokes the *EMP*'s web service that retrieves appropriate information about *Bob* (Bob's name, salary, and start of employment) and provides it as input to the *MBR*'s web service it invokes. Once the *MBR*'s web service responds that the submitted information is accepted, the *EMP*'s web service informs *Bob* that his information has been properly forwarded to *MBR*.

Bob then enters the portal of the real estate broker *RE* and selects an appropriate form to request an appraisal of the property MlsID. He also supplies the MortID and, of course, his name. Once the form is submitted, the *RE* organizes the appraisal of the property MortID and then it submits the appraisal to the *INSinc* by invoking the *INSinc*'s web service with parameters MortID and appraised $ value. In addition, it also invokes a web service of the municipality *MUN* to supply to the insurance company *INSinc* confirmation on the municipality services (policing, schools, water, sewage, etc) available in the area in which MortID property is located. The municipality web service looks up the code that identifies such services and invokes the *MBR*'s web services to pass it MortID and the municipality-services-code it looked-up.

Once the *INSinc* receives information on the property MlsID from the *RE* and *MUN*, it constructs an insurance quote on the property MlsID and submits it to the *MBR* by invoking the *MBR*'s web service and providing it with confirmation of insurance availability - it supplies it with MlsID, insured $ value, deductible ($ value), and *Bob*'s name.

It should be noted that once *Bob* initially enters the *MBR* portal and requests the mortgage, the order in which he enters the remaining portals should not matter.

Finally, once the *MBR* receives all information, it prepares a mortgage document and lets *Bob* know that it is ready.

At any time, *Bob* can use *MBR*'s portal to query the state of his application - he is informed which information has been submitted to *MBR* and which is missing.

Notes on Implementation

Your task is to create the various web services and to control the workflow to support the above scenario. You should also **create a logger that stores information about the web service invocations**. Each web service will, as the first step, invoke the logger service to record its invocation, including information about parameters. Before a web service returns with its results, it should invoke the logger service to record that it has finished and information about the result of the web service.

Your services should be hosted on AWS and Azure clouds. For simplification, you may use one web server to host web pages and web services for all companies. Of course, two companies cannot share a web page or a web service. Furthermore, each URL must identify the company that owns it (e.g., by using the company's acronym as a part of URL).

You have to implement workflow coordination using AWS and/or Azure coordination mechanisms (AWS SWF and/or Azure Logic App).

Furthermore, each company should have its own table, hosted in a cloud, that would hold the necessary tables and data.

For top-notch marks/grades, your communication should be over secure channels and successful authentication should return a security token. Authentication may be either "home-grown" (if you are experienced in security/authentication protocol design) or may be using open-source software (preferred). If you build your own authentication server, upon login, it should ask the user for credentials and, if authenticated (all OK), it should return a token that is then presented to the web services when their services are required by an authenticated user. Some organizations, such as an employer, will require their own authentication server and as they are not able to rely on a third-party authentication. However, some organizations, such as MUN, can rely on a third-party authentication, as information they provide is not private.

We strongly recommend that you ensure that you can complete at least a basic project software that works, for instance w/o security or authentication. That means, even you cannot complete the security part, please do submit your workflow that works in insecure way.

Demo/Presentation and Submission Requirements

On the due date, your team will make a demo and presentation first. This is going to be done to the instructor and TA only (not to the entire class). The details of the demo/presentation schedule will be made later.

In the presentation, your workflow/DB/security design will be presented in 4 minutes using at most 5 PowerPoint slides. Then, you will make a live demo that demonstrates all the details of the implemented workflow in 10 minutes. After that (or during the demo and presentation), we can ask some questions. Actually, this demo will serve as testing of your project. All group members must be present during the demo and presentation (no exceptions).

The final submission will be done after the demo and presentation (on the same day until 23:59).

The Content of the Submission

You have to submit (to Brightspace) a zip file that contains:

- A document (pdf preferred but MS Word is OK) that includes:
    - Title page with the usual information (identifies course/project and your team members)
    - Table of Contents
    - URLs related to your application and a description for how the workflow to be used by a user (from end-user's point of view).

- Resources used (Hardware/software platform, including software libraries, used to develop your client, e.g., MacBook Pro with OS X10.9.5 Eclipse SDK v. 3.7.2, … and to develop your web services).
- Identification of any code from other sources that you used (e.g., code snippets form tutorials): Identify source and also where in your code you used the code.
- Description about any framework used to implement web services (like Spring, Jersey, etc.). Description of configuration of the log services, if any
- Sufficient description of your program's tests (input/output) that demonstrates that your software works. Your description needs to state what is tested/shown (e.g., which specific cases) and needs to be supported by figures showing your input/output (e.g., screen snapshots).
- A "*Design*" section that includes:
  - Web services description, including input/output, and functionality. Web services should be grouped logically and should include logger service(s).
  - Description of portals and their hosting details.
  - Workflow coordination description or annotated screenshots. By just reading this part, the reader should be able to understand the portal visits, information exchange among the entities, and any related activities. Also include technical implementation details about the workflow coordination.
  - DB maintained by each organization/entity (include DB schemas)

- Source code (in appropriate folder structure) and documentation sufficient to install and deploy the applications to the cloud and test it.
- All the files should be uploaded in "CSCI4145-Project-{Names}-{B00Numbers}.zip" format.

Grading Criteria

85%: Correct functionality (including presentation and demo)
15%: Documentation (including design and testing parts)

Good Luck

Yichuan Xu and Albert Levi