CS 6314 – Web Programming Languages
Fall 2018
Dr. Mithun Balakrishna
Course Project

A. Project Steps and Deadlines:

- **Project Group Formation**:
    - Due by **Friday, October 12th 2018, 11:59pm**
    - A maximum of three (3) students per project group
    - The group should decide on an appropriate group name
    - One group member should submit a document containing the group name and the group member information i.e. Group name and Group member names, via eLearning
        - Please name the document following the convention "ProjectGroupInfo-GROUPNAME.pdf", where GROUPNAME is your project group's name.
        - Submit the document to the "Group Information Submission" assignment inside the "Final Project" folder listed in the course home page on eLearning.
        - Students that want to work on the project individually should also submit this document
    - Students that need help to form a group should meet the Instructor on **Friday, October 12th 2017** at **8:30pm** in the classroom ECSS 2.201
        - Students that want to work on the project individually do NOT need to do this
- **Computing Resources**:
    - Deadline: **Friday, October 19th 2017, 7pm**
    - Please talk to the Instructor if your group does not have the computing resources (i.e. a laptop/desktop with internet connection and root/administrator privileges) to support implementation of this project.
- **Project Demo**:
    - Due date: **TBA**
    - Demo sign-up details: **TBA**
    - Submit your project source code and report via eLearning before your group's allocated demo session:
        - One group member should submit a single zip file containing the following via eLearning:
            - Project source code/script file(s)
            - A ReadMe file with instructions on how to access the project demo
            - Project report in PDF or MS Word document format.

        - Please name the zip archive document following the convention "ProjectFinalSubmission-GROUPNAME.zip", where GROUPNAME is your project group's name.
        - Submit the document to the "Project Final Submission" assignment inside the "Final Project" folder listed in the course home page on eLearning.
    - Please hand over a hard copy of the project report before the start of your group's demo session with the TA

B. Project Description:

Please design and implement a **responsive web site** and **scalable web application** based on the **Service Oriented Architecture** (SOA) [1]. This project will require the use of **Web Services** [2] and **Microservices** [3][4] for implementing the SOA.

Mandatory Requirements

A. **HTML/CSS/JavaScript**: You are **required** to build your web site's client side Graphical User Interface (GUI) using HTML/CSS/JavaScript. You are **required** to use responsive HTML/CSS/JavaScript templated such as Bootstrap (http://getbootstrap.com) and Foundation 3 (http://foundation.zurb.com), etc.

B. **Server-side Programming**: You can use any programming language for your web site's server-side implementation and your web application's Web Services & Microservices implementation.

C. **Web Application Domain and Functionalities**: The students will implement a real-time online auction web application for selling/purchasing items:

1. A specified number of items will be auctioned by your auction house website on a first-come-first-served basis.
2. Users will specify a particular date (if available) for their items to go on sale
3. Each item will be given an amount of time to be on the auction board. During that time slot, only ONE item will be available for bidding on the auction board. For example, if a maximum of 50 items are to auctioned from 8am to 6pm (i.e. 10 hours) on a particular date. Then each item (and ONLY that item) will be available for bidding for 16 minutes at the allocated timeslot.
4. Bidders will be able to view the auction schedule at least for one month and different items available for bidding during a particular date.
5. All users should have access to real-time updates of the data on the auction board.

The web application should support the following functionalities via a web browser based Graphical User Interface (i.e. webpages):

1. New regular user registration
2. Existing user login and logout
3. Regular User
    - i. Login
    - ii. Logout
    - iii. User profile information display and editing
    - iv. Forgot password functionality
    - v. Ability to post items that you want:
        1. Specify item title
        2. Specify item description
        3. Specify initial/minimum bidding price
        4. Select an available auction date and time
        5. Upload item photo
    - vi. Ability to delete the user's own posted items
    - vii. Ability to bid for items
    - viii. View the auction schedule at least for one month and different items available for bidding during a particular date.
    - ix. Search for items that are available for bidding during a date range.
        1. Results should be should be displayed in a sortable table (i.e. allowing resulting to be sorted on any column)
        2. Search results filtering capabilities on item attributes
    - x. Page listing the summary of all the bids for an item
4. Accessible any unavailable page should retrieve a pretty and generic 404 page
5. Admin User:
    - i. Login
    - ii. Logout
    - iii. Specify the auction schedule/hours for a particular day

D. **Database:** It is mandatory that your project use a database to store all data. There is no restriction on what type of database to use. Any NoSQL database or RDBMS is fine.

   The database SQL or ORM request and response information should be available in the Web-Service web/app server logs for the TA to review the implementation of this feature. In addition, the TA might inspect the database's content getting updated via a database SQL console.

E. **Web Services:** On you web site, a Web Service call should be made for any user operation that requires database access (i.e. to retrieve information or add/update information in the database). These Web Services should be hosted as a different web application and on a different web/application server than the web/application server containing the web site. However, the two different web/application servers can reside on the same machine. For this project, Web Services are platform/programming-language independent, unassociated, loosely coupled units of functionalities that are self-contained and implemented via SOAP/WSDL or RESTful methodologies. **All Web Services should require authentication/authorization to allow only a valid user to access/modify his/her data.**

   The Web Services request and response information should be available in the both the Website and Web Services web/app server logs for the TA to review the implementation of this feature. The implementation of RESTful Web Services and its authentication/authorization feature can also be shown to the TA via browser-based REST clients such as Postman.

F. **Microservices:** The "Bid Processing" Web Service is asynchronous. Its functionality should be supported and implemented using a queue, and a separate webservice or microservice. For example, a "bid processing" request to the web service API should result in the request being added to a queue. Another independent web service or micro service retrieves this request from the queue and processes it. **The web service or microservice performing the final "bid processing" require authentication/authorization to allow only a valid user to perform update to the bidding data.**

   The asynchronous webservice or microservice request, database query, and response information should be available in the Web Service web/app server logs for the TA to review the implementation of this feature.

G. **Other Required Features:** Your web site/application implementation should also include the following three (3) features:

1. High Performance: perform distributed caching. Memcached is a good option for implementing a distributed caching mechanism.
   Cache miss and cache hit information should be available in the web/app server logs for the TA to review the implementation of this feature.
2. Client-Server Communication Encryption: encrypt the communication channel between the client (i.e. browser), web site server, Web Services, and Microservices server using TLS/SSL.
   The TA will check the implementation of this feature on the Website web/app server by checking if the URL in the browser address bar contains the HTTPS protocol.
   The TA will check the implementation of this feature on the Web-Services/Microservices web/app server by:
   - **Examining the web/app server logs for the Web Services and Microservices request calls being requested and responded to with the HTTPS protocol**

     OR
   - **Making HTTPS calls to the RESTful WebServices and Microservices using browser-based REST clients such as Postman**

     OR
   - **Examining the capture logs of packet analyzers such as Wireshark**
3. Request/Response Compression: perform compression (e.g. gzip) of:
   a. web site server's response to the client
      The TA will check the implementation of this feature by looking at the "Content-Encoding" HTTP response header field either in the browser debug console (a.k.a. inspect element console) or in the Website's web/app server log file
   b. web site server's request to the Web Service server
      Optional: The TA will check the implementation of this feature by looking for the "Content-Encoding" HTTP request header field in the Web-Service's web/app server log file
   c. Web Service server's response to the web site's server
      The TA will check the implementation of this feature by:
      - **looking for the "Content-Encoding" HTTP response header field in the Web-Service's web/app server log file**

        OR
      - **looking for the "Content-Encoding" HTTP response header field in the RESTful WebServices call made using browser-based REST clients such as Postman**

Extra Credit Features:

1. Single Sign-On: perform single sign-on using SAML or OpenID/oAuth
   Note: This is tricky given that Web Services and Microservices require authentication/authorization as well.

C. Project Report

Please write a project report (5 to 10 pages) with the following details:

- An architectural diagram showing how the various components (i.e. client browser, web/application servers, database, cache, etc.) interact with each other in your project
- For each module, a clear description of the various technologies considered and the technology that was finally used in the module development. Also provide a reason why a particular technology was selected
- A clear description of the various functionalities that were available to users on your web site
- A clear description of the Web Services supported by your web application. Including the breakdown Web Services into Microservices
- A summary of the problems encountered during the project and how these issues were resolved
- Please specify your group name and group member names on the document's cover/start page

D. Project Point Distribution

1. Maximum points available: 100 points
   a. Aesthetics (i.e. look and feel of web application): 5 points
   b. Web site functionality: 30 points
   c. Web Services implementation: 30 points
   d. Asynchronous web service or microservice implementation: 10 points
   e. Other required features implementation: 18 points total (6 points per feature)
   f. Group information: 2 points
   g. Project report: 5 points
2. Extra Credit for Single Sign-On: 5 points

E. References

[1] “New to SOA and web services” Available at : https://www.ibm.com/developerworks/webservices/newto/

[2] “Understanding Web Services” Available at : https://www.ibm.com/developerworks/websphere/library/techarticles/0307_ryman/ryman.html

[3] “What is Microservices Architecture?” Available at: https://smartbear.com/learn/api-design/what-are-microservices/

[4] “Microservices, SOA, and APIs: Friends or enemies?” Available at : http://www.ibm.com/developerworks/websphere/library/techarticles/1601_clark-trs/1601_clark.html