# CMPE 273 – Enterprise Distributed Systems

## Lab 1 – Using REST (Node.js), HTML 5 and Angular JS

**Due Date: 20 March 2016**

This lab assignment covers developing REST services using Node.js (Express JS), HTML5 and Angular JS. This lab is graded for 30 points and is an individual assignment (no teamwork is allowed)

**Prerequisites:**

- You must carefully read the environment setup document. You should be able to run "Hello World" Node.js application described in environment setup document
- You must know programming language basics, JavaScript

**Grading:**

**Late assignments will be accepted with a penalty of -5 per day late**

## Part 1 – Calculator (8 points):

**Server:** Calculator to demonstrate stateless web services (4 Points)
The server should include the following functionalities:

1. Addition
2. Subtraction
3. Multiplication
4. Division

Take care of exceptions

**Client:** Calculator Client (2 Points)
Use features of HTML5. Client should all the functionalities provided by the web service

**Testing should be done using JMeter:** Automate the processes using JMeter and print the average time required for below operations:

1. Start a timer
2. Invoke 1000 calls on randomly selected tasks
3. Stop the timer and print out the average time for each operation returned by JMeter

4. Start a timer
5. Invoke 5000 calls on randomly selected tasks
6. Stop the timer and print out the average time for each operation returned by JMeter

7. Start a timer
8. Invoke 100 concurrent users with 1000 calls on randomly selected tasks
9. Stop the timer and print out the average time for each operation returned by JMeter

You need not display result of calls to server, only average time for each operation is enough. **Draw the graph with the average times and include it in the report (2 Points).** Compare the server results and discuss in brief.

## Part 2: Twitter Application (16 Points)

In this part you have to build a prototype "Twitter Application" using Node.js, HTML5 and AngularJS.

**Server:** Twitter to demonstrate REST web services **(7 Points)**
Server should perform following tasks:

1. **Basic User functionalities:**
   a. Sign up the new users (First name, Last name, Email, Password). Passwords have to be encrypted
   b. Sign in with existing users
   c. Sign out
2. **Profile:**
   a. About: Birthday, Twitter handle, contact information and location
   b. Followers and Following list. You should be able to follow people.
   c. Show your tweets and re-tweets
3. **Twitter feed functionality** showing tweets of people you are following and option to re-tweet.
4. **Implementing Hashtag** (#) functionality (in Search and Tweets).
5. Implement **Connection pooling** for database access

Validation is extremely important; exception handling should be implemented. **Implementations with proper validations and exception handling will attract good marks.**

Database - MySQL: REST web services are stateless in nature. To express statefull behavior some persistence providing technologies need to be used. MySQL is the optimum choice.

**Client:** Twitter Client **(6 Points)**
Client must include all the functionalities implemented by the web services. Develop the Client using HTML5 and AngularJS. **Client similar to Twitter will attract good marks.**

**Testing of the server should be done using JMeter and Mocha.** Mocha is a node.js testing framework.

**Following tasks to be tested using JMeter: (2 Points)**

Test the server for **100, 200, 300, 400 and 500 concurrent users** with and without connection pooling. **Draw the graph with the average time and include it in the report.**

**Following tasks to be tested using Mocha: (1 Point)**

Implement 5 randomly selected REST web service API calls using Mocha. **Display the output in the report.**

## Part 3 – Questions: (6 Points)

1. Explain the encryption algorithm used in your application. Mention different encryption algorithms available and the reason for your selection of the algorithm used.
2. Compare the Results of the graphs with and without connection pooling. Explain the results in detail. Describe the algorithm of connection pooling used in your application
3. How would you implement Request Caching? Explain in detail. No need to implement a function – use pseudo code or detailed explanation

## Deliverables Required:

- Submission should include only source code (no node modules or lib should be included)
- Project directory should include the group ID/Name (e.g.: Lab1-caffiene)
- Archive the report and source in one archive file (e.g.: ZIP file)
- Do not submit binaries, node modules, .class files or supporting libraries **(3 Points will be deducted if included)**
- Project Report:
    - Introduction: Goals and purpose of your system
    - System Design: Describe your chosen design
    - Results: Screen captures of Part1 and Part2 Client/Servers
    - Performance graphs. Analyze the graphs and explain the results
    - Answers to Questions in Part3
- Sample directory structure for submission (for person with name Smith) Archive the below files in ZIP file with name: **Lab1-Smith.zip**
    - **Smith-lab1-report.doc**
    - **Lab1/ (directory, do not include libs)**
- **Online Submission only –** Submissions should be made on Canvas before the due date.