CMPE273: Enterprise Distributed Systems

Lab 1 Assignment: Using REST (Node.js) and React JS

Due: October 8, 2017

This lab assignment covers developing REST services using Node.js (Express) and ReactJS. This lab assignment is graded based on 30 points and is an individual effort (e.g., no teamwork is allowed).

Prerequisites

• You must have carefully read the Environment Setup document. You should be able to run the "Hello World" node.js application described in Environment Setup document.
• You must know programing language basics, JavaScript.

Grading

Server1/Client1 (7pts) and Server2/client 2 (16pts) and Questions (7pts): total 30 pts.
Late assignments will be accepted, but will be subject to a penalty of -5 points per day late:

• Submissions received at or before the class on the due date can receive maximum

Server 1 - "Calculator" to demonstrate Stateless web services (3 Pts)
The first Node.js based server you need to develop is the "Calculator". This server should perform the following tasks:

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

Take care of exceptions.

Client 1 – "Calculator Client" (2 pts)
Use features of HTML5 and preferably React. Client should all the functionalities provided by the web service.
The client for Server 1 should behave in the following way:
Perform each operation of Server once. Print out the result in a systematic format, then automate the following process and print the average times as mentioned below.

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

1. Invoke 1,000 calculator calls on randomly selected tasks and print average time to perform each operation.
2. Invoke 5,000 calculator calls on randomly selected tasks and print average time to perform each operation.
3. Invoke 100 concurrent users with 1000 calls each to calculator on randomly selected tasks and print average time to perform each operation.

Your output should be one single client run displaying all the requirements. You need not display results of calls to server; only the average time for each operation is enough**. Draw a graph with average time and include it in the report (2 pts).** Compare the performance from server1 and discuss results.

Server 2 - "Dropbox application" to demonstrate RESTful Services (8 pts)
The next node.js based server you need to develop is the "Prototype of Dropbox application". Everyone should create the account on Dropbox and see how it functions. This server should perform the following tasks:

- Basic **Users** functionalities:
    - Sign up new user (at least first name, last name, Email and password)
    - Sign in existing user (Encrypt Passwords)
    - Sign out. Sign Up should have first name, last name, Email and password.
    - Upload a file (small files are good enough)
    - List a file
    - Create a directory
    - Star a folder/directory.
    - Share a folder/directory by email/name/link.

  In order to use the system, a user must sign in first to the system.
- **Users account** should provide basic details such as:
    - **About**: User overview, Work and education, contact info and life events.
    - **Interests** like music, shows and sports.
- Provide **file list and activity report** functionality.
- Provide **Groups (share)** functionalities:
    - Create group
    - Add member in a group
    - Show members in group
    - Assign access permission to a directory
    - Delete member from a group
    - Delete group
- Should perform **connection pooling** for database access.

The Service should take care of exception that means validation is extremely important for this server. **Good exception handling and prototype similar to actual Dropbox application would attract good marks.**

Client 2 – "Dropbox Client" (5 Points)
Client must include all the functionalities implemented by the web services. Develop the Client using HTML5 and ReactJS. Client similar to Dropbox 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 10 front-end and 10 back-end test cases for mocha. **Display the output in the report.**

Create private repository on the GitHub or bitbucket to manage source code for the project. Add description for every commit. Description should consist of one-line overview on what is committed. Include GitHub/bitbucket project link with access credentials in your report.

Questions (7 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 graphs with and without connection pooling of database. Explain the result in detail and describe the connection pooling algorithm used in your code.
3. What is SQL caching? What all types of SQL caching is available and which suits your code the most. You don't need to implement the caching, write pseudo code or explain in detail.
4. Is your session strategy horizontally scalable? If **YES**, explain your session handling strategy. If **NO**, then explain how can you achieve it.

Deliverables Required:

- Submissions shall include **source code only** for each client/server pair
- Project directory must include the group ID/Name (e.g., Lab1-caffeine)
- Archive the project, and report into one archive file (e.g., zip)
- Do not submit binaries, .class files, or supporting libraries (e.g., junit.jar, javaee.jar) (including them would be **3points** deduction).
- Include the Readme file to document the steps to run the application.
- All the dependencies should be added into package.json file.
- Project report
    - Introduction: state your goals, purpose of system,
    - System Design: Describe your chosen system design
    - Results: Screen image captures of each client/server pair during and after run.
    - Performance: What was performance? Analyze results and explain why are you getting those results.
    - The answers to the questionsposed

  For example:

  Smith is submitting a project. You have provided the following files and source directory:

    - smith-lab1-report.doc
    - lab1/ *(do not send class or jar files)*
- zip –r Lab1-smith.zip Lab1

Submission

- **On-line submission**: shall include all source zipped with your last names (ex. Lab1-Smith.zip) and your report (smith_lab1_report.doc). Submissions shall be made via Canvas.