CMPE273: Enterprise Distributed Systems

Lab 1 Assignment: Using REST (Node.js) and React JS
Due: March 11, 2018

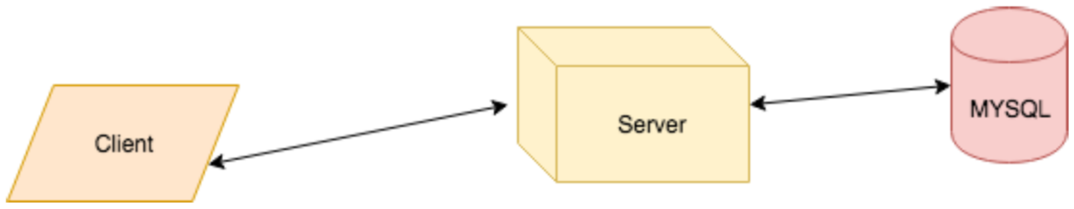
This lab assignment covers developing REST services using Node.js (Express) and ReactJS. This lab assignment is graded based on 30 points and is an individual effort (***No teamwork is allowed***)

Prerequisite

- 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

Part 1 - Calculator - 5pts
Part 2 - Freelancer Application - 18 pts
Questions - 7pts
Total - 30pts
Note: 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.



Part 1 - Calculator

Server - demonstrate RESTful Services [*2 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
Client - [*1 pts*]
Make attractive and simple User Interface to perform above operation.

Note: Client can take values from users and send to a server to perform the required operation. A Server will process the inputs and send output back to a client. A Client will display output to users. Calculator Application without User Interface will get zero marks for Part - 1.

Testing should be done using JMeter: [*2 pts*]
Automate the processes using JMeter and print the average time required for below operations:

1. Start a timer
2. Invoke 1,000 calculator calls on randomly selected tasks.
3. Stop the timer and print out the average time to perform each operation

1. Start a timer
2. Invoke 5,000 calculator calls on randomly selected tasks.
3. Stop the timer and print out the average time to perform each operation

1. Start a timer
2. Invoke 100 concurrent users with 1000 calls each to calculator on randomly selected tasks.
3. Stop the timer and print out the 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.*** Compare the performance from server and discuss results.

Part 2 - Freelancer Application

Server - demonstrate RESTful Services (8 pts)

The next node.js based server you need to develop is the "Prototype of Freelancer application". Everyone should create the account on Freelancer and see how it functions.

This server should perform the following tasks:

a) Basic **Users** functionalities:
   1. Sign up new user (Name, Email and password)
   2. Sign in existing user
   3. Sign out.
   4. Profile (Profile Image, Name, Email, Phone Number, About Me, skills)
   5. Users can update Profile anytime.

To use the system, a user must login first to the system. Password must be encrypted.

b) **Post Project** Functionality:
   1. All Users can post project as an employer (Project Title, Project Description, File Upload, Skills Required, Budget Range)

c) **Home**
   1. Users can see a list of all open projects. (Project *Name*, Description, Skills Required, *Employer*, Budget Range, Number of Bid yet, Bid Now)

d) **Details View**:
   1. Project Details. (Project Name, Description, Files, Skills, Budget Range, Average Bid)
   2. All users can bid on the project. (Bid, Period in days)
   3. List of All bids. (Profile image, Freelancer Name, Bid Price, Period in days, Hire Button only visible to employer of project)

e) **Dashboard**
   1. List of all projects you have bid on. (Project Name, Employer, Avg. Bid, your Bid, status of project)

2. List of all Projects you have published as employer. (Project Name, Average Bid, Freelancer Name, Estimate Project completion Date, status)

f) Should perform connection pooling (in-built in mySql) for database access.

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

Client - [*7 pts*]
A client must include all the functionalities implemented by the web services. Develop the Client using HTML5 and ReactJS. A Simple, attractive and Responsive client attracts good marks.

Note: Every field in an entire project must have validation. User's Name (Navigate to Profile) and Project Name (Navigate to Project Details view) must have hyperlinks.

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

1. **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.**

2. **Following tasks to be tested using Mocha**: **(1 Point)**
   Implement five randomly selected REST web service API calls using Mocha. **Display the output in the report.**

Create a private repository on the GitHub or bitbucket to manage source code for the project. Add a description for every commit. A description should consist of a one-line overview on what is committed. Include GitHub/bitbucket project link with access credentials in your report. Regular commits to your repository are mandatory. Include GitHub /bitbucket commit History to your report.
(Penalty for not including commit history would be 3 points).

Questions (7 pts)

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 in-built mysql connection pooling of database. Explain the result in detail and describe the connection pooling algorithm if you need to implement connection pooling on your own.

3. What is SQL caching? What all types of SQL caching are available and which suits your code the most. You don't need to implement the caching, write pseudocode or explain in detail.

4. Is your session strategy horizontally scalable? If **YES**, explain your session handling strategy. If **NO**, then explain how you can 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 **3 points** 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 questions.

  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.