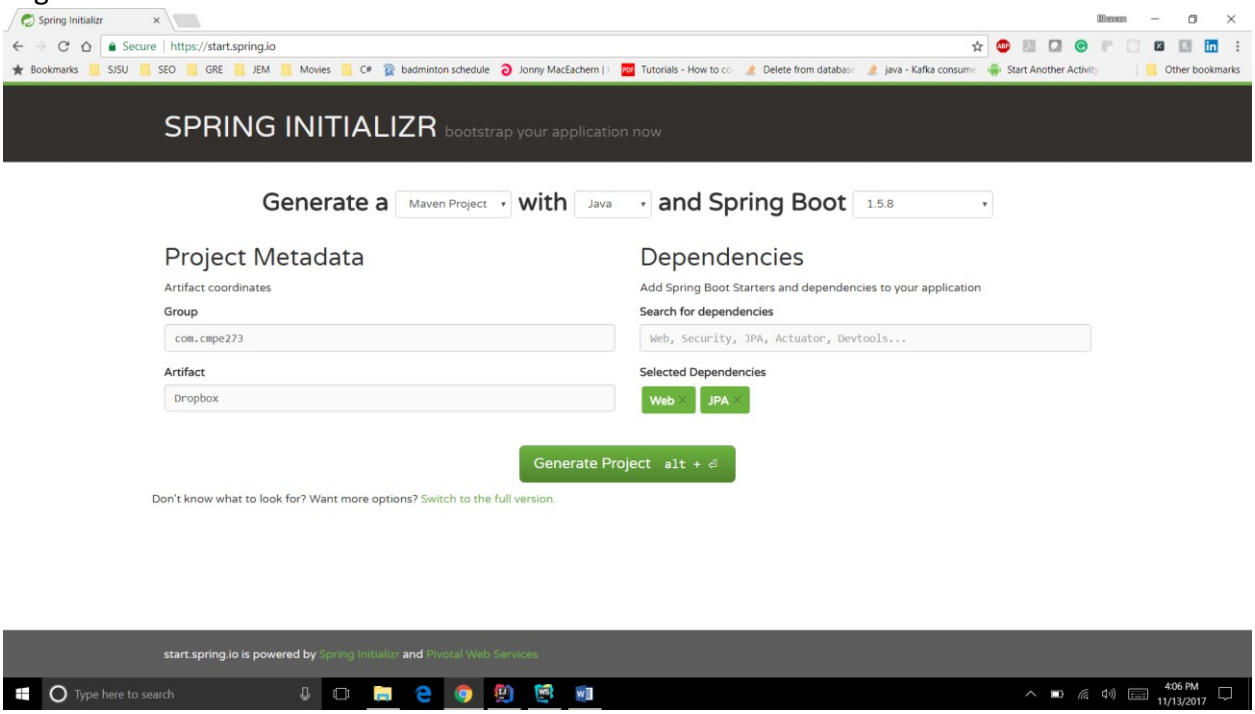
CMPE 273 – Enterprise Distributed Systems
Lab3 Assignment: Using Spring Boot with JPA

Due: May 8, 2018

This lab assignment covers developing REST API using Spring boot and JPA (Java Persistence API). This lab assignment is graded based on 20 points and is an **individual effort** (e.g.: no teamwork allowed)

Prerequisites

- You must have carefully read the Environment Setup document. You should be able to run the Login Demo from Canvas.



- You must know Java, Maven, MVC, collection API.

The Assignment

You will be developing two clients and servers during this lab.
On the due date, turn in the following (via canvas):

- A code listing of each of your clients/servers
- Screen captures of each client/server during execution

Grading

- 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

Freelancer Application

Server - demonstrate RESTful Services (***15 pts***)
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)

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 - [*2 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**: **(1 Points)**
   Test the server for **100, 200, 300, 400 and 500 concurrent users**. **Draw the graph with the average time and include it in the report.**

2. **Following tasks to be tested using Junit**: **(2 Point)**
   Implement five randomly selected REST web service API calls using Junit. **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).

Deliverables Required:

- Submit **architecture diagram.**
- Submissions shall include **source code only** for each client/server pair
- Project directory must include the group ID/Name (e.g., Lab3-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 pom.xml 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.

For example:

Smith is submitting a project. You have provided the following files and source directory:
- smith-lab3-report.doc
- lab3/ *(do not send class or jar files)*
- zip –r Lab3-smith.zip Lab3

Submission

- **On-line submission**: shall include all source zipped with your last names (ex. Lab3-Smith.zip) and your report (smith_lab3_report.doc). Submissions shall be made via Canvas.