**CMPE273: Enterprise Distributed Systems**

**Lab 2 Kafka And MongoDB**

# Due: Nov 4, 2018 23:59 PM

This lab covers designing and implementing distributed service oriented application using Kafka. This lab is graded based on 30 points and is an individual effort (no teamwork allowed)

# Prerequisites:

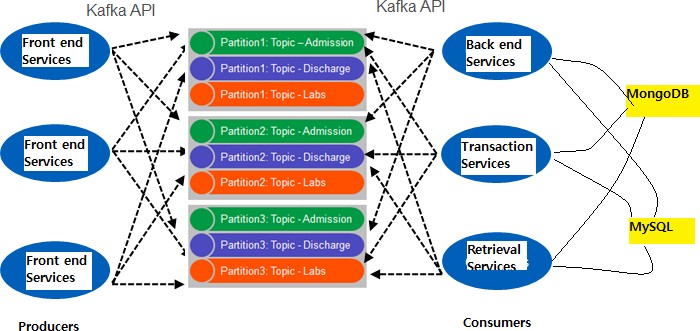
* You should be able to run Kafka sample example.
* You should have prior knowledge of JavaScript, default Sessions

# Grading

* + Submissions received at or before the class on the due date can receive maximum
  + Late assignments will be accepted, but will be subject to a penalty of -5 points per day late

# The Assignment

* + You will be developing a client and server
  + Separate Node into two parts connected via message queues.

Design “backend services” as consumer and “frontend services” as producer as shown below diagram.

* + Implement your own **connection pooling** as discussed in the lecture.
  + Use MongoDB as the database.
  + Passwords need to be encrypted (use **passportJS + JWT**).
  + Use of Redux is mandatory for state management.
  + Client and Server should communicate via **Kafka** Streams.
  + On, or before the due date, you have to turn in the following:
    - Code listing of client and server
    - Document with architecture of the Kafka interaction in your client/server application, system design description and screenshots

**HomeAway Application**

**Server** - demonstrate RESTful Services (12 pts)

The next node.js based server you need to develop is the “Prototype of HomeAway application”. Everyone should create the account on HomeAway 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, City, Country, Company, School, Hometown, Languages, Gender)

5. Users can update Profile anytime.

To use the system, a user must login first to the system. Password must be encrypted. Use PassportJS with JWT for session management. Use of Redux is mandatory.

b) **Post Property** Functionality: (Owner)

1. Only the Owner can post property for rent (Property Location, Details, Booking Options, Photos, Pricing, Amenities, Availability)

c) **Home**

1. Users can search for a property for rent based on search criteria (Place to Visit, Start Date, End Date, No of Guests).

2. Include search bar where you can search property based on location, dates and number of guests.

d) **Details View**:

1. Property Details. (Name, Type, Files, Bedrooms, Bathrooms, Sleeps, Price)
2. Allow users to book a listing.
3. Add pagination max 10 properties per page. Add a filter on price, bedrooms, location and dates.

e) **Dashboard**

1. List all previous booking history. (Owner Dashboard)
2. List all previous trip details. (Traveler Dashboard)

3. Include search bar for searching property by Name. Add pagination max 5 property per page. Add filter on dates.

f) **Ask Owner a Question**

* Traveler can ask a Question to the owner.
* Owner can reply to a message after he logs in back.
* Show the messages under Inbox Tab.

g) Should perform **DB provided** for database access.

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

**Client -** [ *4 pts*]

A client must include all the functionalities implemented by the web services. Develop the Client using HTML5 and ReactJS-Redux. 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) etc. must have hyperlinks.*

**Hosting -** [ *8 pts*]

* You must host your HomeAway lab to cloud.
* You can user either Heroku or Amazon Web Service to host your client and server.
* You can use MLab for your MongoDB database.

**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 (a)** without connection pooling **(b)** DB provided connection pooling and. **Draw the graph with the average time, your analysis of the graph on why, why not and how in the report.**

1. **Following tasks to be tested using Mocha**: **(2 Point)**

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

**Questions (6 pts)**

1. Compare passport authentication process with the authentication process used in Lab1.
2. Compare performance with and without Kafka. Explain in detail the reason for difference in performance.
3. If given an option to implement MySQL and MongoDB both in your application, specify which data of the applications will you store in MongoDB and MySQL respectively

# Deliverables Required (Git Deliverables):

* Inside the git repository assigned to you, create a folder, Lab 2.
* Inside this folder create three sub-folders, one for Frontend-Code and one for Backend-Code and one for Kafka. Place all your source code in respective Folders.
* 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.**

# Submission (Report Submission)

* **On-line submission**: shall include your report (smith\_lab1\_report.doc). Submissions shall be made via Canvas.
* **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 running.
  + Performance: What was performance? Analyze results and explain why you are getting those results.
  + The answers to the questions.