CMPE 273: Enterprise Distributed Systems

Lab 2 Assignment: Using REST (Node.js) and React JS

Due: November 6th, 2020, 2:00 PM

This lab covers designing and implementing distributed service-oriented application using Kafka
This lab is graded based on 35 points and is an individual effort (no teamwork allowed)

Prerequisite

● You should have prior knowledge of JavaScript, React, Redux, MongoDB, Passport, JWT token, AWS EC2

• You should be able to run Kafka sample example to be demonstrated in the class.

Grading

Yelp Application - 30 marks
Questions – 5 marks
Total – 35 marks

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.

Yelp

Lab 2: Using REST (Node.js), React JS, Redux, Passport, MongoDB and Kafka

You need to develop "Prototype of Yelp application". This prototype will be a web application using React and Node. Refer Yelp website and see how it functions. You need to use Kafka messaging service and make the system distributed.

The application should have the following persona:

1. Customer
2. Restaurant

You need to implement the following features in your application for the roles given above.

1. Customer Signup (name, email id, password)
2. Restaurant Signup (restaurant name, email id, password, location)
3. Sign in
4. Sign out

A Restaurant should be able to do the following functionalities:

Restaurant page (Dashboard – Landing page):

1. View restaurant profile – having all basic information about restaurant (name, location, description, contact information, pictures of restaurant and dishes, timings)
2. Update restaurant profile (name, location, description, contact information, pictures of restaurant and dishes, timings)
3. Update Contact information
4. Add/Edit Dishes in menu (with Dish name, Main Ingredients, Dish Images, Dish Price, description, dish category – Appetizer, Salads, Main Course , Desserts, Beverages)
5. View list of dishes added by them.
6. View Reviews given by customers
7. **Pagination should be implemented while showing dishes.**

Orders page:

1. View list of orders by customers
2. Click and view profile page of each customer
3. Update the delivery status of each order – Order Received, Preparing,
   (If Delivery option selected) On the way, Delivered
   (If Pickup option selected) Pick up Ready, Picked up
4. There should be 3 filters on orders(New Order, Delivered Order, Cancelled Order)
5. **Pagination should be implemented.**

Events Tab:

1. Post events(event name, description, time, date, location, hashtags)
2. View list of people who registered for the events.
3. Click on name and view another's person profile.
4. **Pagination should be implemented.**

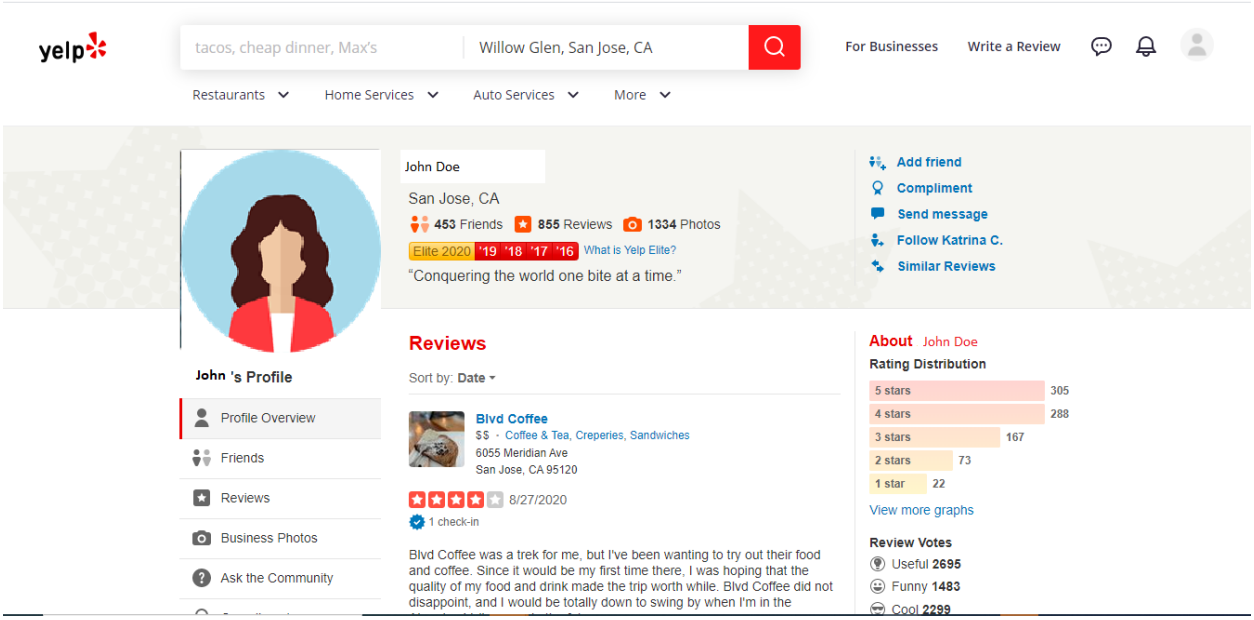
Messages Tab(New):

1. Restaurant can message customers from customer's profile.
2. Customer should not be able to initiate a conversation with a Restaurant.
3. Customer can view messages and reply to restaurant only if the conversation was started from Restaurant side.
4. View the list of conversations he/she had in Messages Tab**.**
5. Clicking on each conversation should display the message history in sorted order and Customer should be able to continue the conversation by replying**.**

A **Customer** should be able to do the following functionalities:

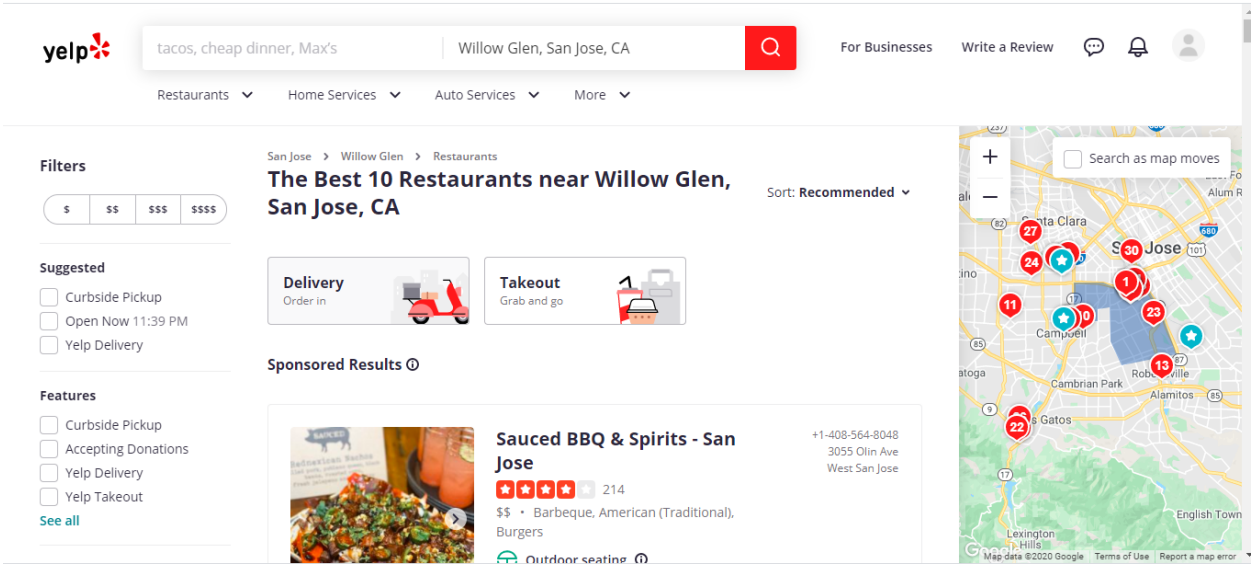
Profile Page:

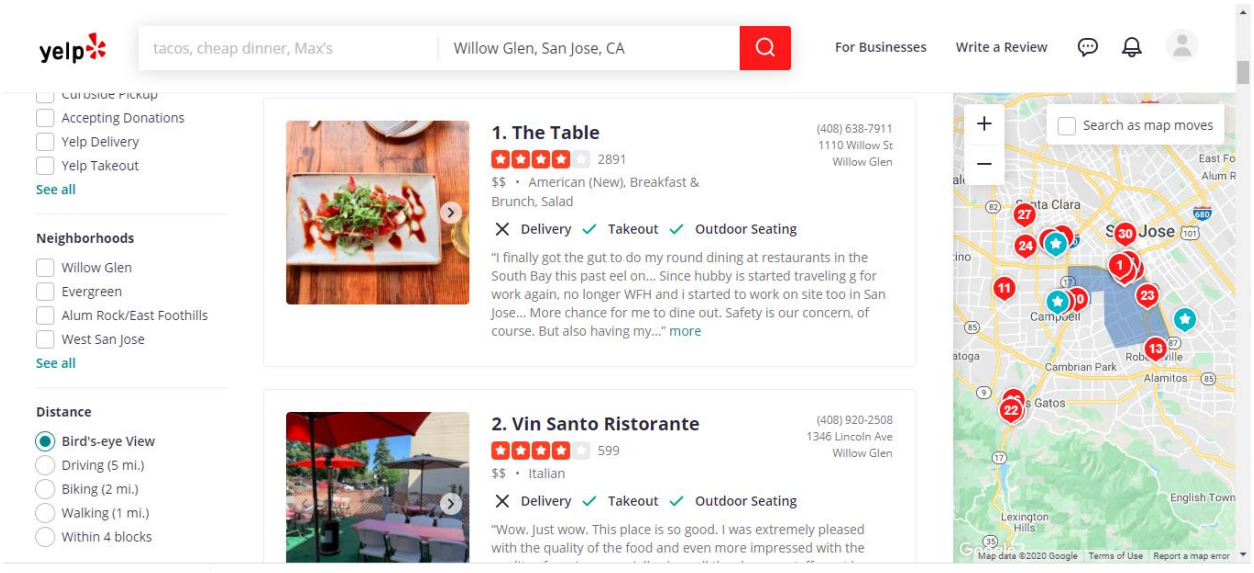
1. Display complete profile of a customer (basic details, favorites, about, profile picture)
2. Upload profile picture
3. Update about section(Yelping since, Things I love, Find Me In, My Blog or Website, etc.)
4. Update basic details (name, date of birth, city, state, country, nick name) and headline
5. Update Contact Information (email id, phone number)



Restaurant Search tab (Dashboard – Landing page)

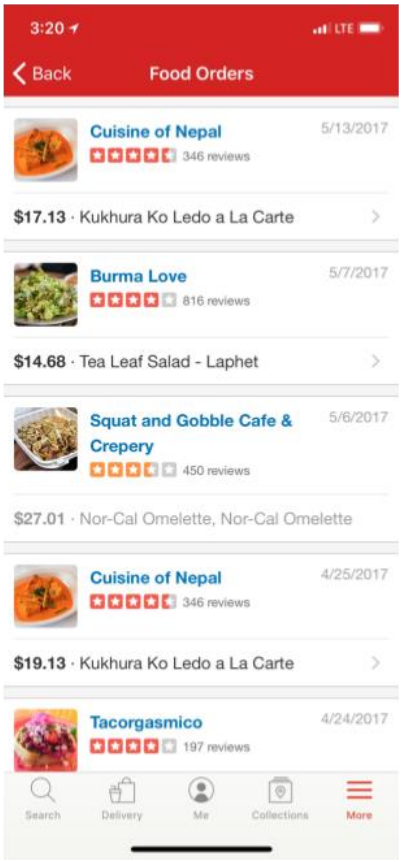
1. Search for Restaurant (using dish names, cuisines, location, mode of delivery)
2. Filter restaurant search results based on food delivery method (Curbside Pickup, Dine In, Yelp Delivery) and location (neighborhoods)
3. Click and view a restaurant page
4. Show each restaurant respective location on map via pins(You can use google map API for this).





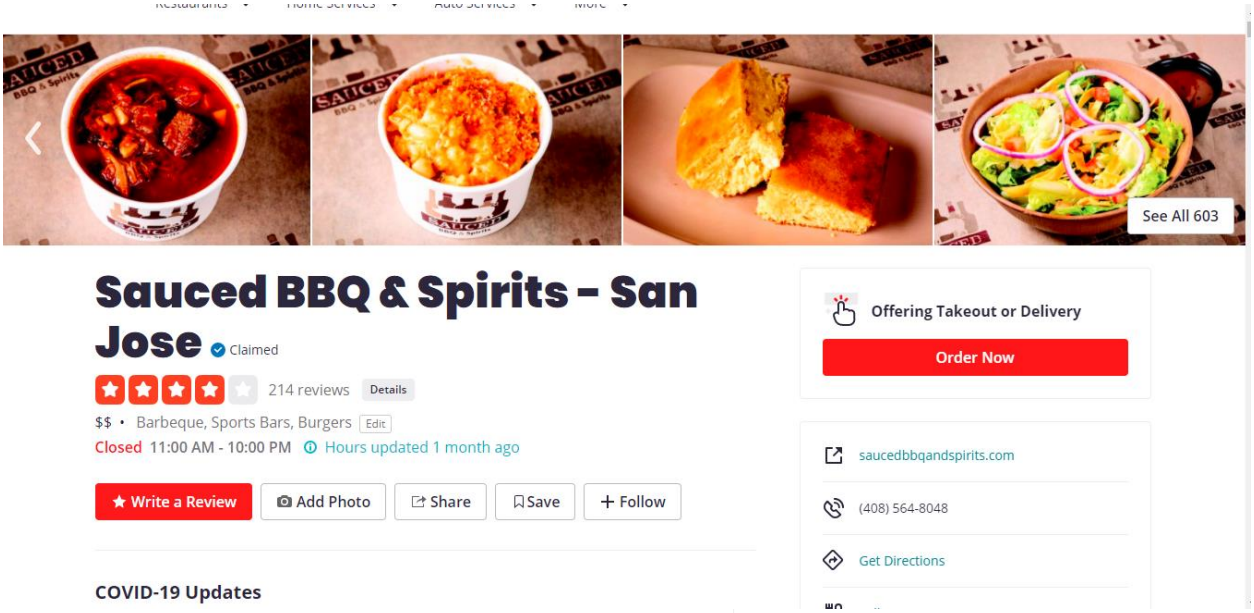
Orders Tab

1. View list of all the orders placed (along with order Date-Time, order status)
2. Filter the order based on the order status – Order Received, Preparing,
   (If Delivery option selected) On the way, Delivered
   (If Pickup option selected) Pick up Ready, Picked up
3. Sort the results on the basis of order date-time both ascending and descending.
4. **Pagination should be implemented.**



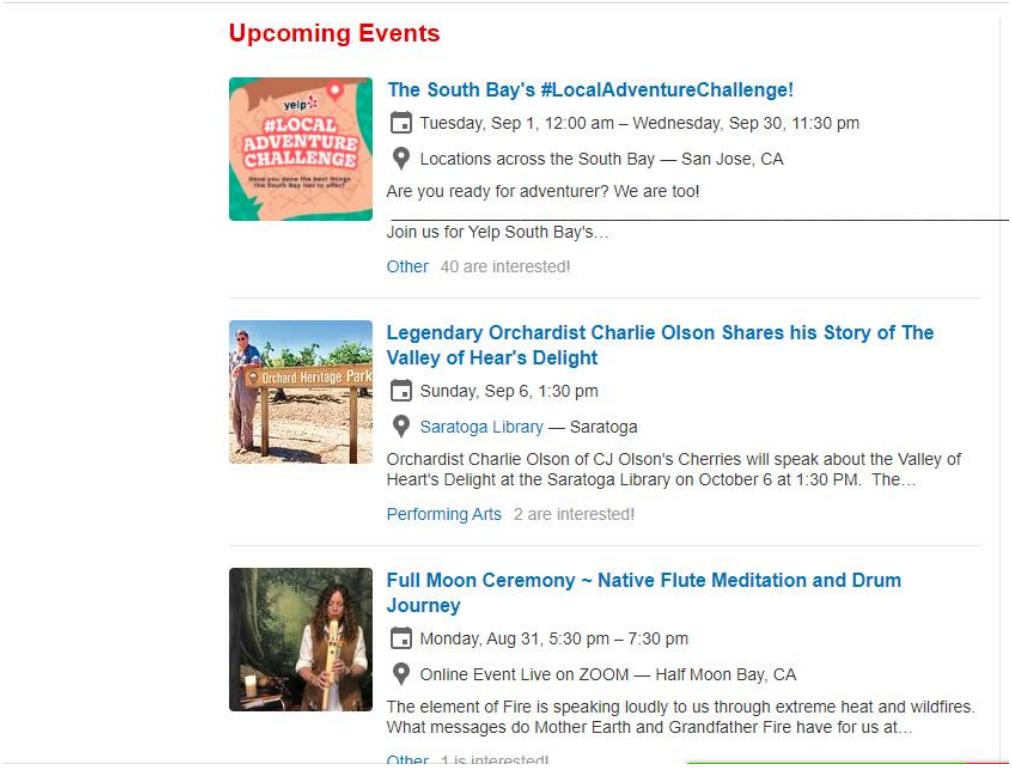
Restaurant Tab

1. View Restaurant Page
2. Add insightful reviews(date, ratings, comments)
3. Order food from the page, select delivery method.
4. **Pagination should be implemented.**



Events Tab

1. View list of upcoming events in the order of increasing date
2. Search for event (using event name)
3. Click and view event details
4. Register for an event.
5. View list of registered events
6. **Sort Events based on event date both ascending and descending.**



Users Tab (New)

1. View list of users using yelp
2. Search for user (using user's first name or nickname)
3. Filter the results based on their Location
4. Click on user's Name to view user's profile
5. Follow any user by clicking follow button in their profile.
6. Filter users based on following.
7. Pagination should be implemented.

- Every service should have proper exception handling and every input field should have proper validation.
- Password values should be encrypted.
- **Passport.js and JWT Tokens should be used for authentication.**
- **MongoDB with connection pooling should be used.**
- **Redux should be implemented in all the components.**
- **Kafka should be used.**
- ESLint should be used in your code following Airbnb style guide.
- The application should be deployed on cloud (E.g. Heroku, AWS EC2)
- A simple, attractive and responsive client attracts good marks.

Testing

1. Testing of the backend server should be done using JMeter and Mocha.
2. Enzyme should be used to test at least 3 views/pages(Optional).

- Following tasks to be tested using JMeter:
  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(1 API).

- Following tasks to be tested using Mocha:
  Implement five randomly selected REST web service API calls using Mocha. Display the output in the report.

Questions

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 part of data of the application you will store in MongoDB and MySQL respectively

Git Repository

- In your Git repository, create two sub-folders, one for Frontend and one for Backend. Place all your source code in respective Folders.
- Add a proper description for every commit describing what code changes are added.
- Regular commits to your repository are mandatory. (Penalty of 3 marks if missed).
- Do not submit dependencies or supporting libraries (e.g. node_modules) (including them causes deduction of 2 marks).
- All the dependencies should be included in package.json file.
- Readme file of your repository should contain the steps to run the application.

Project Report

- Introduction: State your goals and purpose of the system
- System Design: Describe your chosen system design
- Results: Screen captures of testing results and important screens of the application.
- Performance: What was performance? Analyze results and explain why you are getting those results.
- Git Commit history – screen capture
- Answers to the questions.

Submission

Please upload your report (John_Lab2_Report.doc) on Canvas before deadline. (Number of pages in Report should be below 30)