CMPE 273: Enterprise Distributed Systems

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

Due: November 7th, 2021, 11:59 PM

This lab covers designing and implementing distributed service-oriented applications using Kafka. This lab will grade for 35 points and is an individual effort (no teamwork allowed)

Prerequisite

- You need to have prior knowledge of JavaScript, React, Redux, MongoDB, Passport, JWT token, AWS EC2 and Kafka

Grading

UberEats 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 a maximum.

Uber Eats

You need to develop a “Prototype of **UberEats** application”. This prototype will be a web application using React and Node. Refer to the **UberEats** website and see how it functions.

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 the 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 the menu (with Dish name, Main Ingredients, Dish Images, Dish Price, description, dish category – Appetizer, Salads, Main Course, Desserts, Beverages)
5. View the list of dishes added by them.

Orders page:

1. View list of orders by customers
2. Click and view the 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, Cancel the order
4. There should be 3 filters on orders (New Order, Delivered Order, Cancelled Order)

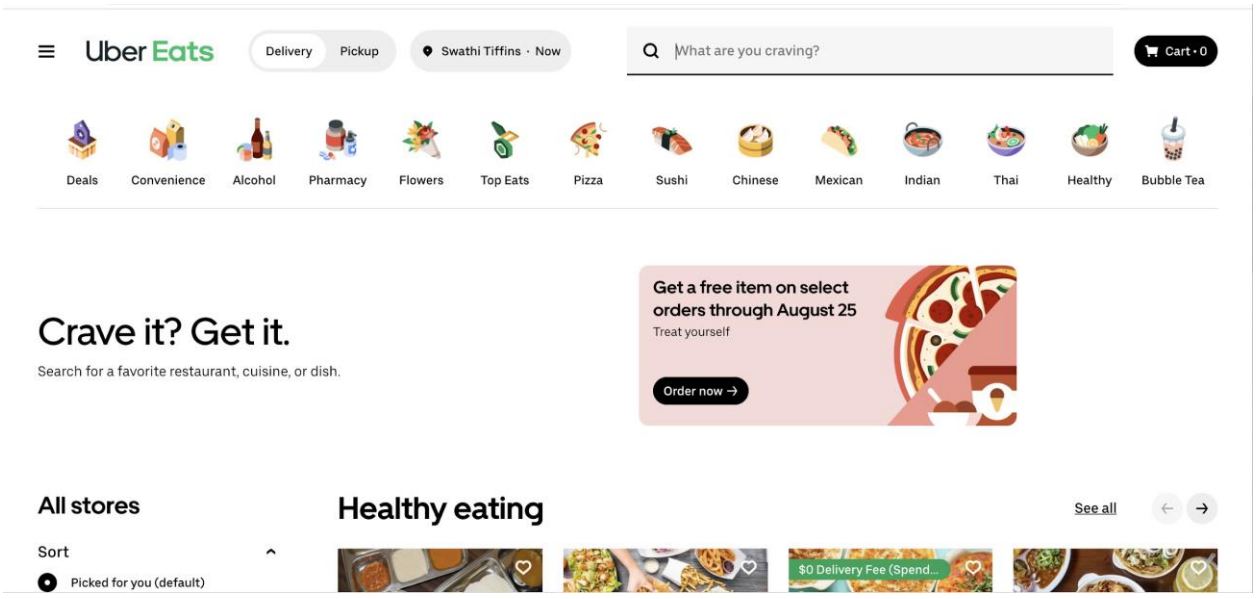
A **Customer** should be able to do the following functionalities:

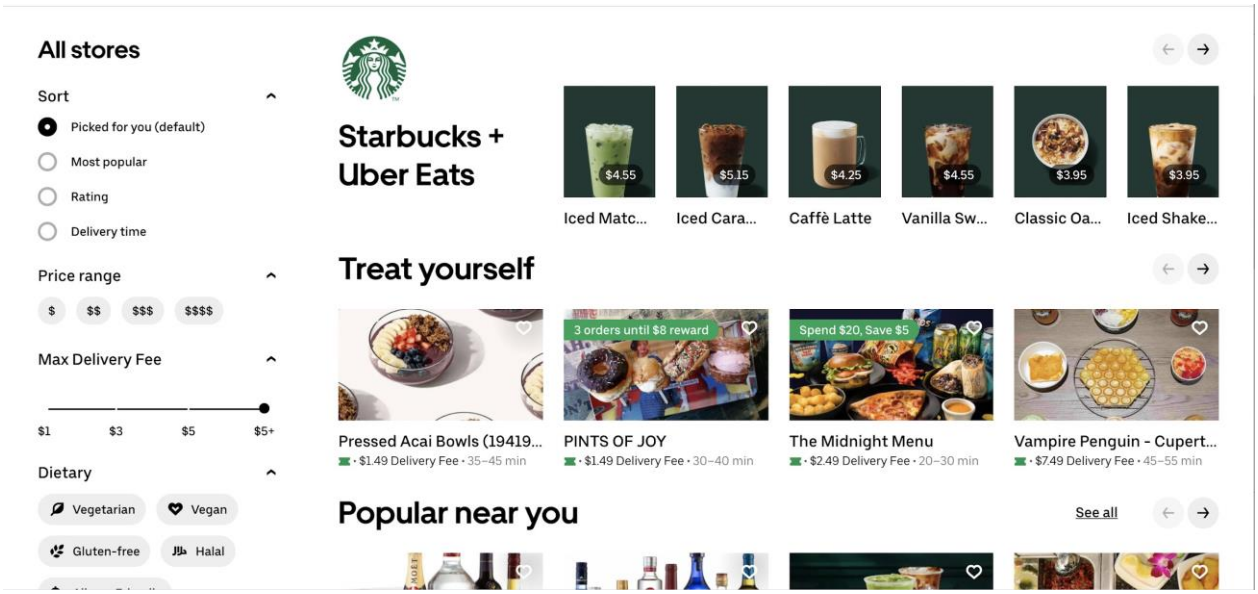
Profile Page:

1. Display complete profile of a customer (basic details, favourites, about, profile picture)
2. Upload profile picture
3. Update basic details (name, date of birth, city, state, country, nickname)
4. Update Contact Information (email id, phone number)
5. **The country** should be a dropdown with the list of countries to select from and should not be entered manually.
   Note: Your profile page need not be the same as Uber Eats profile as they have little information available.

Restaurant Search tab (Dashboard – Landing page)

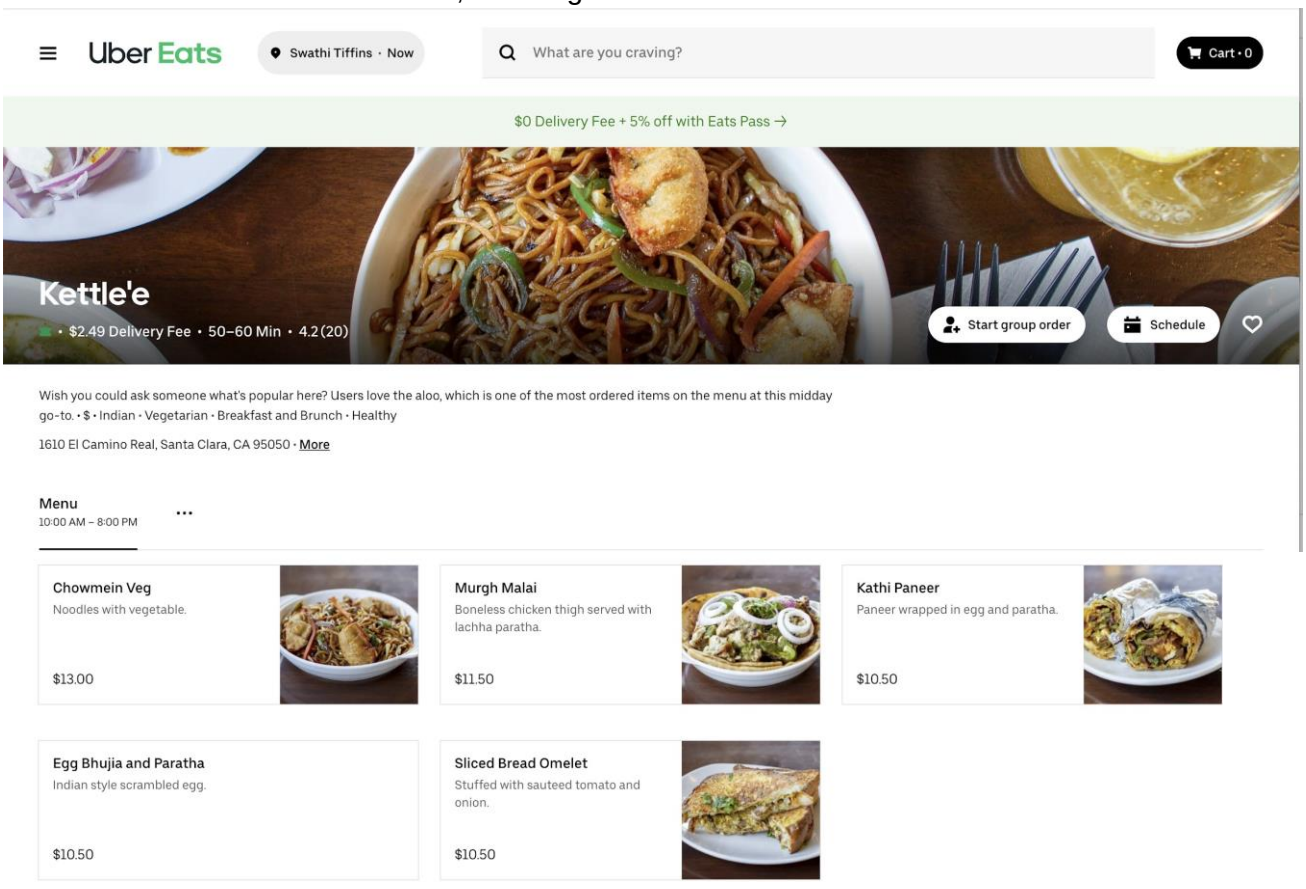
1. Display the list of restaurants to the customer with the same location entered on the profile page. (i.e, Suppose I entered my location as Sunnyvale, my dashboard should show me all the restaurants available in Sunnyvale on the top and then the other restaurants available in your database. If no location is updated, show all the restaurants)
2. Customers should be able to search for Restaurant (using dish names, cuisines, location, mode of delivery)
3. Add a filter for the type of delivery (Delivery, Pickup) a customer can select. Also, add filters for the type of food a customer can order. The filter should include Veg, Non-Veg, and Vegan.
4. When a filter is selected, the restaurant's page should show the filtered result of the restaurant.
5. Customers should be able to click on the Restaurant and should be redirected to the restaurant landing page.





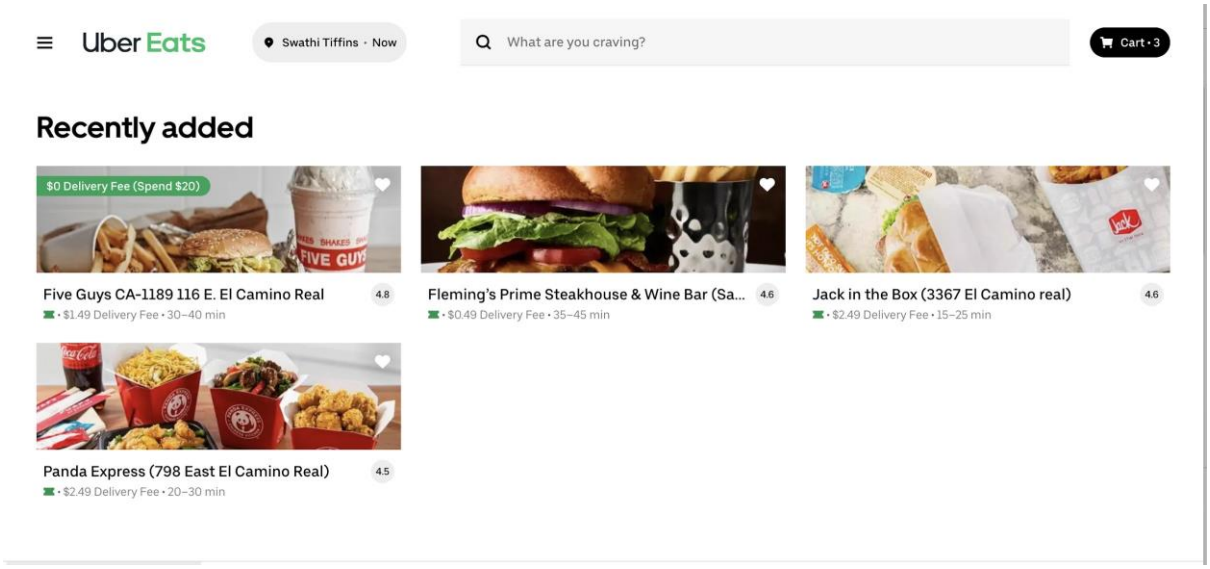
Restaurant Tab

1. View Restaurant Page
2. Add a small description of the restaurant.
3. Show the menu of the restaurant along with the amount.
4. Customers should be able to select a dish and it should be added to the cart.
5. Once the customer is done, he can go to the view cart to finalize the order.



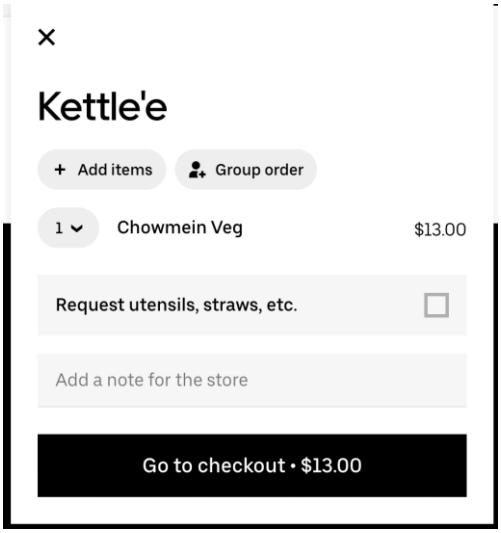
Favourite:

1. Customers should be able to select a restaurant as their favourite.
2. A customer should be able to select the Favorite tab and it should display a list of restaurants selected by the customer.

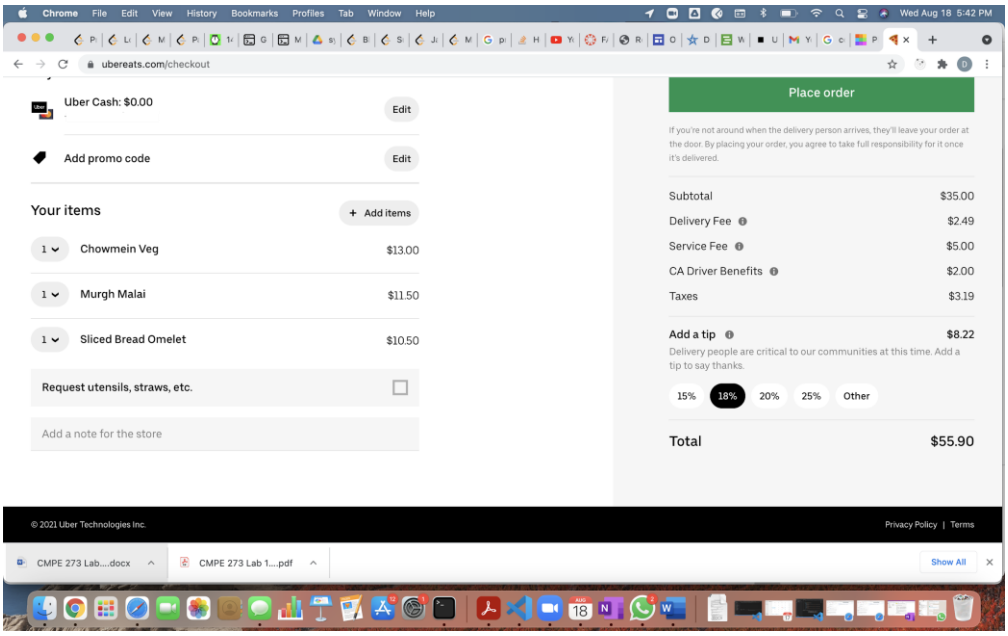


Place Your Order

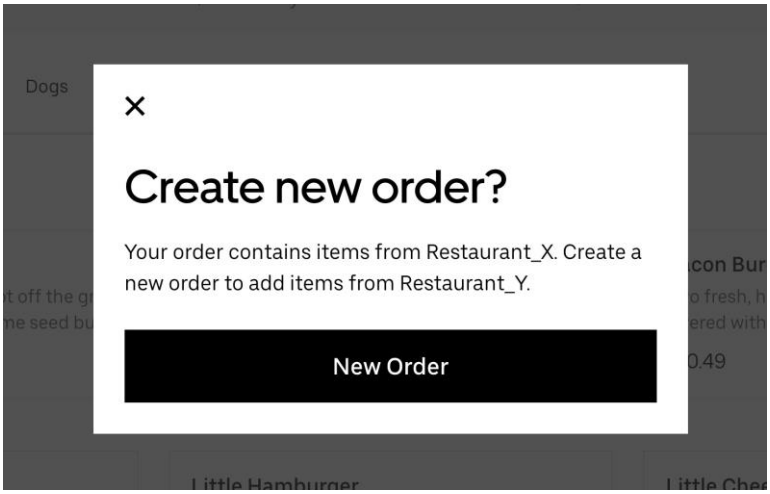
1. The customer should click on the cart and a pop-up window should be displaying the list of items added by the customer



2. Users should be able to edit the items in the above pop up i.e., remove items, modify the quantity.
3. Click on checkout and the customer should be redirected to the next page to order the food.

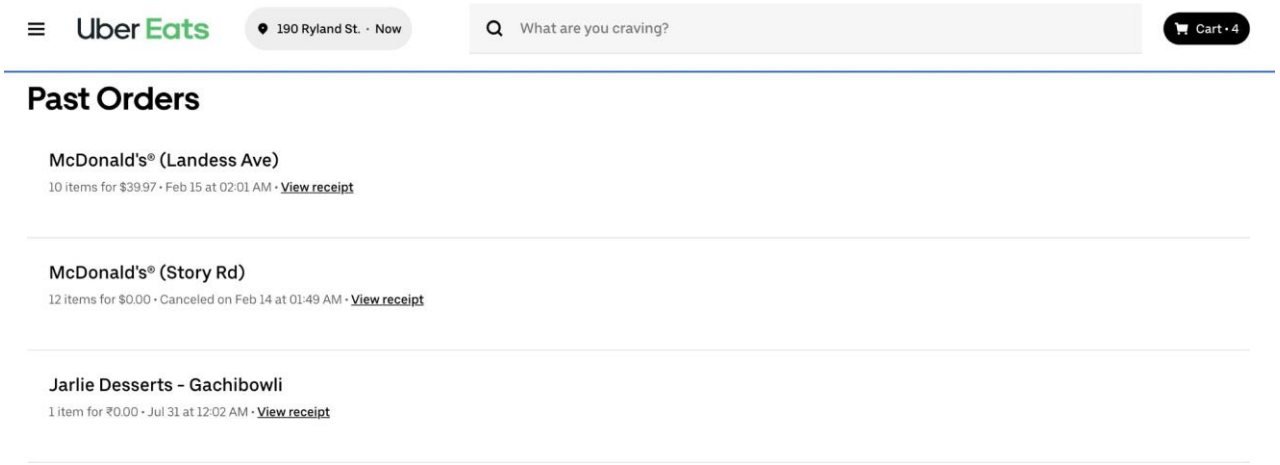


4. Customers should be able to see all the selected items and the total amount which needs to be paid.
5. Users should be able to add a delivery address. If any addresses are previously given, the User should be able to select one from them or add a new one.
6. Users should be able to add special instructions to orders before placing the order and these should reflect in the order receipt/summary page.
7. The user should click on Place order and the page should notify the customer with “Order placed successfully”.
   Note: If a customer is trying to add items from another restaurant, a confirmation dialog should be displayed as shown below.

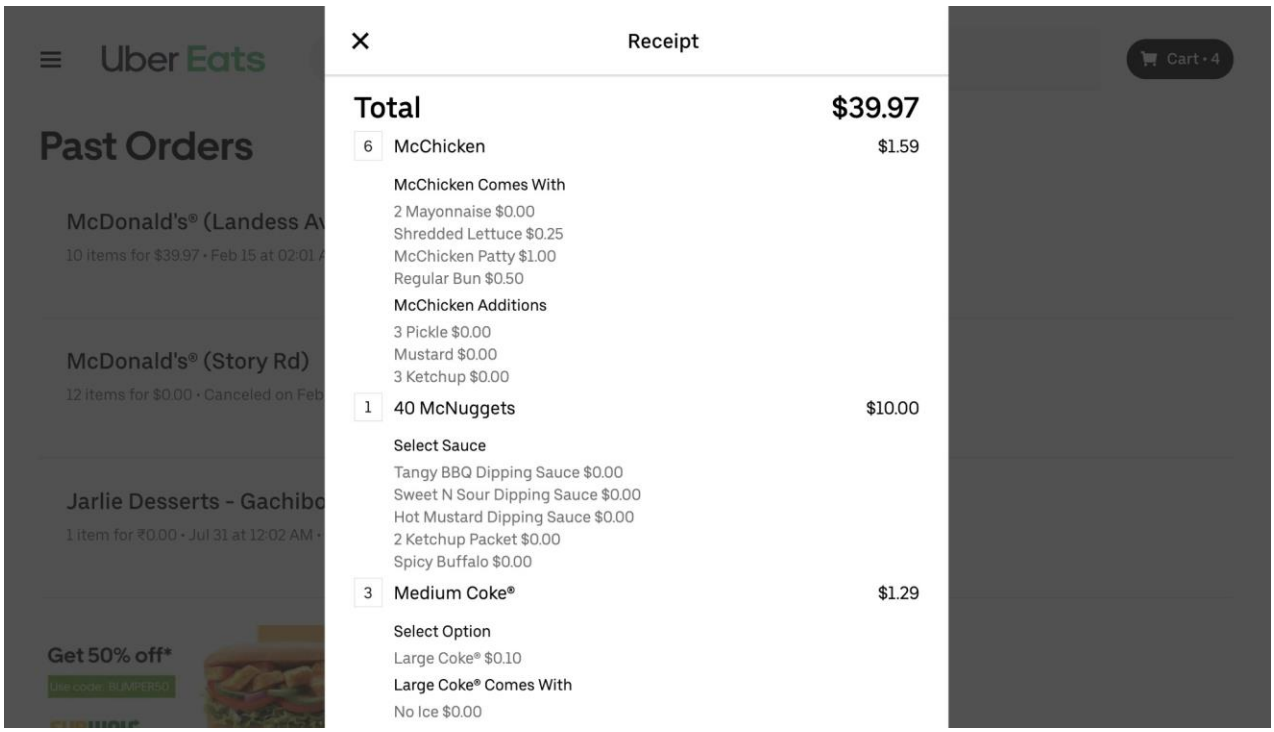


Orders Tab

1. View list of all the orders placed & cancelled (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. Clicking on any order should open a pop-up that displays the order summary, status of the order, delivery address, and other order-related information.



4. Paginate the above orders page and allow users to select page sizes of 2, 5 or 10 from a dropdown. The default page size selected in the dropdown should be 5.
5. User should be able to cancel the order before the restaurant takes any action.

Notes:

1. Every service should have proper exception handling and every input field should have proper validation.
2. Password values should be encrypted.
3. Your code needs to be formatted using ESLint and follow the Airbnb style guides
4. The application needs to be deployed on the cloud (ex Heroku, AWS)
5. Passport.js and JWT Tokens should be used for authentication.
6. MongoDB with connection pooling should be used.
7. Redux should be implemented in all the components.
8. Use any cloud platform to store uploaded images.
9. All images are just for reference. Please follow the instructions written.

ESLint should be used in your code following the Airbnb style guide.

The application should be deployed on the 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. The enzyme should be used to test at least 3 views/pages.

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.

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

Please find the questions for Lab2 which you have to answer in your lab report:
1. Compare the passport authentication process with the authentication process used in Lab1.
2. Compare performance with and without Kafka. Explain in detail the reason for the 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 the package.json file.
- The 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 the 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 the deadline.
(Number of pages in Report should be below 30)