CMPE 273: Enterprise Distributed Systems

Lab 3: Using GraphQL

Due: May 10th, 2020, 11:59 PM

This lab covers designing and implementing a GraphQL application. This lab assignment is graded based on 25 points and should be an individual effort (No Teamwork allowed)

Prerequisite
- You should have prior knowledge of JavaScript, React, Node.js and MongoDB.
- You must have carefully read concepts of Query, Mutations and React Apollo client
- You should be able to run GraphQL sample example demonstrated in the class.

Grading
Application and Code – 20 Marks
Questions – 5 Marks
Total Marks – 25 Marks
Note:
Late assignments will be accepted but will be subject to a penalty of -5 points per day late. Submissions received before the deadline can receive maximum

Handshake

You need to develop "Prototype of Handshake application". This prototype will be a web application using React, Node and GraphQL.

Client must include all the functionalities listed below. Develop the Client using ReactJS and React Apollo Client. You should use MongoDB to store the data.

The application should have the following persona:
1. Student
2. Company

You need to implement the following features in your application for the roles given above.
1. Student Signup (name, email id, password, college name)
2. Company Signup (company name, email id, password, location)
3. Sign in
4. Sign out

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

Job Postings page (Dashboard – Landing page):

1. Post Job openings (with job title, posting date, application deadline, location, salary, job description, job category – full time, part time, intern, on campus)
2. View list of job postings posted by them
3. View list of students applied for each Job posting posted by them.
4. Click and view profile page of each Student

Company Profile Page:

1. View its profile – having all its basic information (name, location, description)
2. Update Company profile (name, location, description)

Students Tab:

1. Search for students (using name, college name)
2. Click and view profile of each Student

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

Profile Page:

1. Display complete profile of a student (basic details, career objective, education, experience
2. Update basic details (name, date of birth, city, state, country) and Career Objective
3. Update Education Details (College name, Location, Degree, Major, Year of passing, current CGPA)
4. Update Experience details (Company name, Title, Location, Start & End dates, Work description)

Job Search tab (Dashboard – Landing page)

1. Search for job postings (using company names or job titles)
2. Click and view a Job posting description
3. Click on the company name to view company profile (name, location, description).
4. Apply for a Job opening by clicking Apply button

Applications Tab

1. View list of all the job postings applied

Students Tab

1. View list of students enrolled in Handshake
2. Search for students (using student name or college name)
3. Click on Student details to view Student Profile.

- Every service should have proper exception handling and every input field should have proper validation.
- Password values should be encrypted.
- A simple, attractive and responsive client attracts good marks.
- Use of Passport.Js, Kakfa and Redux is optional.
- If you are using Passport.js for authentication, then use REST route for login which will be a non-protected route. Obtain token from the login and use this token to access in your GraphQL route for mutation and query calls.

Questions

1. How will you enable multi-part data in GraphQL?
2. Discuss the architecture for using multi-part data in GraphQL without using any open source library from Git.
3. State any open source library for enabling multi-part data transfer using GraphQL with sample code. Argue why do you think that this particular library is a good fit?

Code Repository

- Use Bit bucket for saving your code and keep it private.
- Add a proper description for every commit describing what code changes are added.
- Regular commits to your repository are mandatory. (Penalty of 2 marks if missed).
- Do not submit dependencies or supporting libraries (e.g. node_modules)
- All the dependencies should be included in package.json file.

Project Report

- Link to your Bit bucket repository
- Answers to the questions.
- System Design: Architecture diagram
- Results: Screen captures of important screens of the application and graphql terminal.

Submission

Please upload your report (John_Lab3_Report.doc) on Canvas before deadline. (Number of pages in Report should be below 20)