CMPE 273: Enterprise Distributed Systems

Class Project (Fall 2020) - Simulation of Glassdoor.com

Team size: **6 people per team**
Due date: **December 4th, 2020**

1. **API Documentation:** **October 30th, 2020**
   - Submit your API report document on Canvas before 11:59 PM.
   - You must turn in a document describing the interface of your backend server consisting of all the API request- response descriptions.

2. **Presentation:** **November 20th, 2020**
   - Submit your presentation document on Canvas before 2:00 PM
   - Group wise presentation should be given in class meeting.
   - Describe the system architecture used in your project.

3. **Project Demo:** **December 4th, 2020**
   - Submit the project report before 2:00 PM.
   - Demo of your application in class meeting.

4. **Peer Review:** **December 4th, 2020**
   - Submit the peer review document on Canvas before 11.59 PM.
   - Grade your team members except yourself on the scale of 10 and describe their contributions to the project.
   - This will be an individual submission and should keep the grades and comments confidential.

Glassdoor.com

Glassdoor is a website where current and former employees anonymously review companies. Glassdoor also allows users to anonymously submit and view salaries as well as search and apply for jobs on its platform. It also allows user to create job alerts and save their searches for future use. The site also allows the posting of office photographs and other company-relevant media.

Project Overview:

In this project, you will design a 3-tier distributed application that will implement the functions of Glassdoor website. You will use the techniques used in your lab assignments to create the system. In Glassdoor prototype, you will manage and implement different types of users.

Though Glassdoor.com has wide range of features, this project is limited to only few important functionalities which involves writing reviews anonymously, post jobs, apply for jobs, reveal interview experience and interview questions and implement the functionalities which involve different roles.

For each type of object, you will also need to implement an associated database schema that will be responsible for representing how a given object should be stored in a database.
Your system should be built upon some type of distributed architecture. You have to use message queues as the middleware technology. You will be given a great deal of "artistic liberty" in how you choose to implement the system.

Your system must support the following types of entities:

a. Students(Employees included)
b. Company
c. Admin users

Your project will consist of three tiers:
- The client tier, where the user will interact with your system
- The middle tier/middleware/messaging system, where the majority of the processing takes place
- The third tier, comprising a database to store the state of your entity objects

Tier 1 – Client Requirements

The client should have a pleasing look and be simple to understand. Error conditions and feedback to the user should be displayed in a status area on the user interface. Try to create a UI similar to Glassdoor.

Your application should not allow users to look at more than 5 reviews without logging and writing their own review.

Login & Sign up

- User should be able to create a new account by providing his Name, email id and password.
- A user can sign up either as a Student or a Company.
- In case of Company, the name should be treated unique.
- User should be able to login using his email id and password.
- There should be single login page for Student, Company and Admin.
- You can create credentials for admin from backend.
- Based on their account type, they should be redirected to their respective pages.

Company/Job Search Results page(Landing Page for Students)

- Student should be able to search company based on Company name.
- Student should be able to search jobs based on Job title.
- Student should be able to search Interview experience and salary of particular company based on company's name.

Company Search page

- Student should redirect to the company search page where results of your search should display.
- Companies detail should have the following details:
    - Company name
    - Average Rating
    - Location(City, State)
    - Company website
    - Number of Reviews
    - Number of Salary Reviews
    - Number of Interview Reviews
- Pagination should be used to display search results. (Server side recommended for better performance).
- Click on company name to open "Company Home Page"

Company Home Page

- There will be 6 sub-tabs in company's page:
    - Company Overview(Default tab)
    - Reviews
    - Jobs
    - Salaries
    - Interviews
    - Photos
- Pagination should be implemented in every tab

Company Overview Tab

- Company overview tab should have following details about the company:
    - Company name
    - Website
    - Size
    - Type
    - Revenue
    - Headquarters
    - Founded
    - Company Description
    - Company Mission
    - Average Rating(Overall, Recommended to a Friend, CEO approval)
    - Featured review selected by company should be displayed on the profile.
    - 2 reviews to be displayed(1 negative review and 1 positive review)

Reviews Tab

- Reviews Tab should have following features:
    - Average Rating(Overall, Recommended to a Friend, CEO approval)
    - Review Card should have Review headline, average rating(Overall, Recommended to a Friend, CEO approval), Description, pros, cons, helpful button.
    - 2 reviews to be displayed(1 negative review and 1 positive review) with most helpful votes.
    - Add Review Button

Jobs Tab

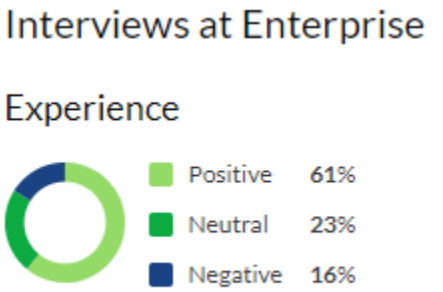
- Jobs Tab should have following features
    - Search all the open jobs in the company
    - Filter the jobs based on job title and city
    - Job card should have Job title, Company name, City, State, posted date, favorite icon
    - Clicking on a Job tab should open respective job page which will have Job description, Responsibilities, qualification, resume upload button, cover letter upload button, Apply Job button.

Salaries Tab

- Salaries tab should have following features
  - Add salary button which will open Modal where you can add salary details like (Base Salary, Bonuses, Job Title, Year of Experience, Location, Employer name)
  - Salary range of all the job title

Interviews Tab

- Add an interview button which will open modal where you can add interview details like (Employer name, Rate Overall experience(Positive, Negative, Neutral), Job title, Description, Difficulty(Easy, Average, Difficult), Offer Status(Rejected, Accepted), Interview Questions, Answers)
- Display experience, Offer status, Experience rating in percentage.(Example shown in below picture)



- In interview review card, show all the details related to the interview.

Photos Tab

- Office photos should be shared in this tab.
- Add photos button that will open a modal where user can select multiple images to upload.

Jobs Home Page

- List of all the jobs should appear sorted by most recent posted date.
- User can sort search results by most recent posted date and most rated.
- User can filter jobs by salary range, location, job type
- Job card should have following details(Company name, average rating, Job title, State, city , expected salary(If present), date posted.
- User can click on job title and can see further details regarding job like (Job description, Responsibilities, qualification, resume upload button, cover letter upload button, Apply Job button)

Job Application Page

- User can track the application status of the application.
- If the application status is submitted, user can withdraw the application.
- Employer will not be able to retrieve the user's job application if it is withdrawn by user.

Student Profile & Activity

- User should be able to view his profile, update his name and profile picture.
- User should be able to view the counts of ratings he has added, reviews he has provided.
- User should be able to view the list of reviews he has added for companies, jobs, interviews, and salaries.
- Users should also be able to see the photos they added for a company.
- Clicking on each review should take the user to the review page.

Resumes Tab

- Users can add/delete their resumes.
- Users can mark one resume as a primary resume.
- This resume will be automatically selected when user apply for a job. User has a option to change the resume for that particular job role if he wants.

Job Preferences

- User can update their job preference.
- Job preference should have following fields
    - Job Search Status(Not Looking, Casually Looking Actively Looking)
    - Job title he is looking for
    - Target Salary
    - Open to Relocation(Yes/No)
    - Type of Industry they prefer

Demographics

- User can update his race/ethnicity, gender, Disability, Veteran Status

Employer

Type text here

An employer will be having a different set of pages. He will not be able to view Student's pages.

Company Profile page

- An employer should be able to view, edit and update his profile page containing his name, address and profile image.
- An employer should add/edit following details for the company.
    - Website
    - Company Size
    - Company Type
    - Revenue
    - Headquarters
    - Industry
    - Founded
    - Mission
    - CEO Name

Review Page

- Employer can see every rating and review made by users on his company.
- An employer can reply to the review made on his company.
- Employer can also mark some reviews as favorite.
- Employer can mark featured review if he wants to show that review on company's profile.

Job Posting Page

- An employer can post the jobs from job posting page.
- Employer should give following details about the new job
    - Company name
    - Job title
    - Industry
    - Country
    - Remote/ In-person
    - Street address
    - City
    - State
    - Zip
- Pagination should be used to display products.

Applicant Page

- Employer can go through each job posted by them.
- Employer can see number of applicants for each job.
- Employer can click on job title and can see the list of users applied for the job.
- Employer can see the resume and cover letter attached by applicant in front of their name.
- Employer can then click on applicant's name and can see the profile page of applicant with the job preference of that employee.
- Employer can change the applicant status.
- Status of application (Submitted, reviewed, initial screening, Interviewing, Hired)

Report

- Employer should be able to see the statistics of all the jobs posted by him in last year, applicants applied to it, applicant selected, applicant rejected.
- Employer should be able to see the statistics of demographics of applicant applied for his job.

Admin User

They are the service staff working for Glassdoor who will have to keep a track of all the reviews and photos added by users in the website. They will be able to view different set of pages.

Reviews and pictures

- Admin should be able to filter out inappropriate reviews added by users.
- Admin should be able to filter out inappropriate photos added by users.
- Only after the admin approval, a review or image will be made public, before admin approval only the user can see the review and image.

Company Profile Page

- Admin users should be able to see the list of companies in the system.
- They should be able to search for companies using Company name.
- On clicking a Company name from results, they should be able to view all the reviews of the company(both approved reviews and rejected reviews).
- Admin user should also be able to see the statistics of job related data of that particular company. For example (Number of hired applicant, demography of the applicant).

Analytics Dashboard

- Admin user should be able to view different types of analytics graphs showing following data:
    - No of reviews per day.
    - Top 5 most reviewed company.
    - Top 5 company based on average rating.
    - Top 5 students based on total accepted reviews made.
    - Top 10 CEO's based on rating.
    - Top 10 company based on viewed per day.

Tier 2 — Middleware

You will need to provide/design/enhance a middleware that can accomplish the above requirements. You have to implement it using REST based Web Services as Middleware technology. Next, decide on the Interface your service would be exposing. You should ensure that you appropriately handle error conditions/exceptions/failure states and return meaningful information to your caller. Your project should also include a model class that uses standard modules to access your database.

Use Kafka as a messaging platform for communication between front-end channels with backend systems.

Tier 3 — Database Schema

- You will need to design your database to store your data in both relational and NoSQL databases. Choose column names and types that are appropriate for the type of data you are using.
- You will need to decide which data will be stored in MongoDB and MySQL. (Hint: Choose the database for every information stored based on pros and cons of MongoDB vs MySQL)
- You need to implement SQL caching using **Redis**

Scalability, Performance and Reliability:

The hallmark of any good object container is scalability. A highly scalable system will not experience degraded performance or excessive resource consumption when managing large numbers of objects, nor when processing large numbers of simultaneous requests. You need to make sure that your system can handle many users and incoming requests. Pay careful attention to how you manage "expensive" resources like database connections.

Your system should easily be able to manage 10,000 users and 10,000 reviews+photos. Consider these numbers as **minimum** requirements. Further, for all operations defined above, you need to ensure that if a particular operation fails, your system is left in a consistent state. Consider this requirement carefully as you design your project as some operations may involve multiple database operations. You may need to make use of transactions to model these operations and roll back the database in case of failure.

Testing:

To test the robustness of your system, you need to design test cases to test all the functionalities that a regular client would use. You can use your test cases to evaluate the scalability of the system as well by running the tests with thousands of products and users. Use these tests to debug problems in the server implementation.

- **Mocha Testing:** Implement testing on **Ten** randomly selected REST web service API calls using Mocha and include the results in the project report.

Presentation:

1. Group number and team details
2. Database Schema
3. System Architecture Design Diagram
4. Performance comparison bar graphs that show difference in performance by adding different distributed features sat a time such as B (Base), S (SQL Caching), D (DB connection pooling), K (Kafka) for 100, 200, 300, 400 and 500 simultaneous user threads. (5 Bar graphs in total).
   Combinations are:
   a. B
   b. B+D
   c. B+D+S
   d. B+D+S+K
   e. B+D+S+K+ other techniques you used

   Note: Populate DB with at least 10,000 random reviews and measure the performance on an API fetching the reviews.
5. Performance Comparison of services with below deployment configurations with Load balancer
   a. 1 Services server
   b. 3 Services server
   c. 4 Services server

Project Report:

Your project report must consist of the following:

- A title page listing the members of your group
- A page describing how each member of the group contributed for the project. (One short paragraph per person)
- A screen capture of your database schema
- System Architecture Design diagram
- A short note describing:
  - Your object management policy
  - How you handle heavy weight resources
  - The policy you used to decide when to write data into database
- Screen captures of your application (few important pages)
- A code listing your server implementation for entity objects
- A code listing of your server implementation of the security and session objects
- A code listing of your main server code
- A code listing of your database access or connection
- A code listing of your Mocha test and output result screenshots
- Observations on performance of the application based on the comparative analysis at different deployment configurations with graph

Scoring Points:

- **Functionality** (40 marks) – Every functionality mentioned in the requirement above carry some marks and they will be randomly tested along with the code and ESLint.
- **Scalability** (15 marks) – Your project will be tested with thousands of objects. There should not be any degradation in performance of application and it should not crash for any reason.
- **Distributed Services** (15 marks) – Divide the client requirements into different services and deploy them on the cloud. Each service should be running on the backend server connected using Kafka. Divide the data into MongoDB and MySQL and provide reasons for the choice of database for every object.
- **UI** (15 marks) **–** It is important that the user interface of your application is similar to Glassdoor.com.
- **Report** (15 marks) **–** Performance testing results, Mocha testing results and your explanations in Project Report carry marks.

Notes:

1. Maintain a pool of database connections. Opening a database connection is a resource-intensive task. It is advisable to pre-connect several database connections and make use of them throughout your database access class so you don't continually connect/disconnect when accessing your database.
2. To prevent a costly database read operation, you can cache the state of the entity objects in a local in-memory cache. If a request is made to retrieve the state of an object whose state is in cache, you can reconstitute the object without reading the database. Be careful to update the cache, when the state of the object changes. You are required to implement **SQL caching using Redis**.
3. Don't write data to database that didn't change since it was last read.
4. Focus on the implementation of all the features first. Complete implementation that passes all feature tests is worth as much as performance and scalability part of the project.
5. Do not over optimize. Groups in past in attempt to optimize the project, implemented complex optimizations but failed to complete the feature set.
6. Try to implement all the features with proper exception handling and user-friendly errors. Implement proper input validations in all pages. Do not give scope for the user to make the application crash.
7. Try to implement all the features as similar as it is in Glassdoor.com.

Type text here