Project Report

Group 041: Qijian Wu, Xinhao Zhao, Yifan Zhu

A brief introduction of our project:

In this project, we developed JobInsights, a website where you can discover and share job information. We implemented the backend using the Flask framework, the frontend with React, and deployed the SQL database on GCP. Users are divided into two roles: regular users and administrators. Regular users can search, comment on, bookmark, upload, and update job information. Data visualization and personalized recommendations are incorporated to enhance the user experience. Administrators can review job information uploaded or updated by users to ensure that the job listings visible to users are accurate and legitimate.

Question answers:

- Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).
  Our final project fully implements all the features outlined in the Stage 1 proposal.

- Discuss what you think your application achieved or failed to achieve regarding its usefulness.
  Our application is user-friendly and highly useful. On the Dashboard, users can discover a wide variety of job information, and they can leave ratings and anonymous comments on each job listing. These ratings and comments provide valuable references for other users. Users can bookmark the job listings they are interested in, and later access them in their personal favorites page. Our application also offers personalized recommendations to help users discover jobs that suit them. Additionally, by browsing visualized job data, users can quickly identify trending jobs and companies. Our platform provides publicly shared job information, all of which is uploaded by users. We have implemented an administrator review mechanism to ensure that all job information is accurate and legitimate.
  However, there are areas where our application could be improved. For instance, we could limit each user to only one rating per job to prevent multiple ratings from affecting the authenticity of the feedback. Additionally, we could offer the option to sort the job listings in search results by rating, salary, or other criteria in descending order, making it easier for users to find the information they need.

- Discuss if you changed the schema or source of the data for your application
  We did not change the schema designed in stage 2. The job data is from Kaggle. The company data is also from Kaggle. We made a unique ID for every job and removed some unrelated columns. For the location, since the job has different locations in different formats. We cleaned the location column. So that they can be joined with the company location. The user table is generated and we created some users ourselves. The other table such as upload history, favorite, review is appended when users are using the website.

- Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?
  We ensure the diagram and implementations are the same as the original design.

- Discuss what functionalities you added or removed. Why?
  We finished all the functionalities. We designed Dashboard, Job Visualization, Upload Job and Favorite Job and recommendation. We finished all those functionalities.

- Explain how you think your advanced database programs complement your application.
  - Constraints: We used various constraints, such as numerous foreign key constraints, as well as value constraints like Job.Salary >= 0.
  - Triggers: We used two triggers:
    1. update_job_clear_comment: This trigger clears the corresponding AdminComment in the UploadedHistory table when a job record in the Job table is updated, and the updated column is not Rating. This ensures that the administrator receives a reminder to re-review the job listing.
    2. update_job_rating: This trigger is activated when a new comment with a rating is submitted. It calculates and updates the Rating value for the corresponding job in the Job table.
  - Stored procedure: We used stored procedures when querying the database to retrieve data for visualization which improves efficiency and reduces cost.
  - Transaction: We also used transactions for visualization to ensure atomicity and consistency

- Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.
  - Qijian: In the pending jobs system, initially, my approach was to change the ApprovalStatus of a job from FALSE to TRUE when an admin accepts a job, and to delete the job from the Job table if the admin rejects it. However, this approach made it impossible for users to see the job's upload history. Additionally, we needed to allow users to modify and resubmit a job for review if it was rejected. Therefore, I adopted a different approach: if the admin rejects a job, I only set the AdminComment in the UploadedHistory table to "Reject" rather than deleting the job. This ensures that the job is invisible to other users but still allows the original user to update and resubmit it for review. For pending jobs visible to the administrator, they only need to meet the condition where Job.ApprovalStatus = FALSE and AdminComment is either NULL or not "Reject".
  - Yifan: If you plan to use K Means for the recommendation, an extra column is required to record the clutter number. This can reduce the computational resource usage. We used the nearest neighbor, which does not need to record clusters.
  - Xinhao: When testing data visualization, I noticed that it uses up CPU resources very fast, therefore we should be wary of which features incur high costs and test these features less frequently while ensuring their quality.

- Are there other things that changed when comparing the final application with the original proposal?
    - We added more features and fine-tuned some of the more vague features
    - Aside from that, the application we built closely followed our initial proposal.

- Describe future work that you think, other than the interface, that the application can improve on
    - The application can be improved by providing a portal where users can apply for these jobs easily.
    - add a page where employers can post jobs directly.
    - add a private message system where employers can DM users for job opportunities.

- Describe the final division of labor and how well you managed teamwork.
    - User&Admin login, pending jobs system, jobs' upload/update: Qijian
    - Job data visualization and review (comment and rating) system: Xinhao
    - Job Dashboard, Favorites feature, job recommendation: Yifan