CS411 Final Project Report

Changes from Original Proposal:

- *Discuss what you think your application achieved or failed to achieve regarding its usefulness.*
- *Discuss if you changed the schema or source of the data for your application*
- *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?*
- *Discuss what functionalities you added or removed. Why?*
- *Explain how you think your advanced database programs complement your application.*

Regarding its usefulness, our application achieved the goal as an internship finder for job hunters with filters and our gamified streak concepts. The filter for locations and work type, along with the search bar for job titles, allows users to find more relevant jobs easily. Streaks create an incentive for users to apply every day; otherwise, they lose a streak count, which draws users to use our application daily. It also fulfills the goal as an internship tracker, as users can enter the status of the applications they have applied for. However, we implemented only some of these basic functionalities, which caused our application to fail in some areas regarding its usefulness. For instance, we removed the social network features where users could see the streaks and points of their friends, which lowered the incentive for people to apply for jobs without competition. Some of the filters may not include salary or job industry due to limited data and time, which may decrease its usefulness.

The final project implemented was extremely similar to the one originally proposed. The job search and filter system was implemented based on the proposal and we added the gamified system with streaks where you could track the number of applications submitted. The functionalities that we removed are the quizzes for personalizing the job search dashboard, the social network features, and the more complex visualization tasks. We ended up not having enough time to implement the quizzes or any of the social network features and we plan on implementing that as a future feature. However, we added a more basic functionality such as a leaderboard which showcases top five schools and along with top students with highest score in order for recruiters to see which schools have the most applications.

The data sources for the application were not changed from the proposal. We still used the same data sources as proposed. The schema for the application was unchanged and there were no modifications from what was originally designed versus the final end product. We added a couple of minor fields to enhance the performance of queries in the database. This included adding in a last_updated field for the user so it was easier to query for the designed triggers. Additionally we created a separate table called the leaderboard table to hold the results of the stored procedure that was implemented.

In terms of the ER diagram and table implementation, we created a new table called leaderboard to keep track of the top five schools along with the top students with the highest scores. This final design with the new table is more suitable because it allows us to more easily write the front end to support the leaderboard feature, as the front end only requires a single API call to get the relevant data, while the backend handles most of the heavy lifting. We also added more attributes to keep track of additional information. For instance, we added last_app_date to reset the streaks if users have not been applying for jobs.

The advanced database programs in the application help complement the application by shifting a lot of the calculation burden away from the front end and the backend directly to the SQL server. This helps enhance the speed and responsiveness of the frontend and makes the application easier to use. The stored procedure is a good example of this, where instead of having to do the aggregation and calculator of the leaderboard data in the backend or frontend which would be time consuming and costly, it is instead offloaded to the SQL server which does most of the work and allows the backend to return the data to the front end using a single API call.

We created a trigger on the applications table that monitors when a user makes an application. It checks if the last application the user made is more than 3 days old, at which point it will reset the streak on the user. Otherwise, we update the score and the streak of the user using an algorithm to encourage them to consistently apply. We also created another trigger that would update the last application a user made to keep track if the streak expired. Furthermore, we created a stored procedure that recalculates the school ranking leaderboard by the total score metric of its students, and created a daily event that calls the procedure.

Technical Challenges Encountered:

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.

Daniel Zhou: The biggest challenge I faced in the design of this application was working with the GCP platform and hosting the application on GCP. The SQL server was built with the GCP SQL service which was a simple operation that only required the GUI. Building the backend application that interfaced with the SQL server was a more complicated endeavor that required us to use a Google Cloud run service. This service allows you to run lightweight containers on Google Cloud and automatically create a new deployment of the service everytime the github repo adds a new commit. This was useful for automatically deploying the code and making sure that the backend version matched the current state of our work but required us to create container file definitions for the application. If I was starting the application from scratch I would recommend reading up and understanding guides on how to containerize applications and download a local docker client to be able to test the container file image before pushing it to GCP.

Amy Bisalputra: One of the technical challenges I faced was implementing the functionality on the front-end side of the website with limited information from the dataset. Since the dataset did not initially support some of the job tags or include all the details outlined for the UI, we had to adjust some implementations and front-end displays to align with the available data. My advice to another team starting a similar project would be to thoroughly understand the API and data constraints before beginning the implementation of front-end functionality for a smoother development process.

Hieu Nguyen: We planned the queries beforehand to make sure that everything goes smoothly when we implement the endpoints. However, when we did implement them, some of the queries wouldn't work because of sqlalchemy (we didn't use ORM, only the plain text feature). Understanding the setup and the requirements as well as rewriting the queries took some time, but we eventually got the endpoints to work.

Bhumsitt Pramuanpornsatid: My biggest challenge was to make sure the frontend hosting on firebase connects properly with google cloud run and SQL database. Two particular challenges that I faced were I had to deal with cors issues from having different hostname when communicating with backend, and dealing with firebase authentication and integrating that with the SQL database.

Potential Future Improvements:

Describe future work that you think, other than the interface, that the application can improve on

In the future, we would like to take some time to implement the proposed additional features that we were unable to implement due to time constraints. These features include quizzes with curated feed, advanced visualization and a social system to track and compete with your friends.

We believe that adding streaks would greatly increase the interactivity of the application, and motivate people to apply to more jobs which would improve our functionality and increase the overall usefulness of Jobkinator. In addition, adding a social system on top of these streaks would make it a more social application and one that can be more easily shared between users and create shared experiences and narratives that would further enhance the interactivity of the system. Furthermore, adding interactive quizzes displaying a personalized dashboard or curated job feed based on the user's responses would also increase the interactivity with a more gamified experience. Lastly, by creating the detailed analyses we originally planned to create would offer a more graphical user experience and help them visualize the data in a different way which would be a good way to boost application engagement.

Division of Labor, Teamwork:

Describe the final division of labor and how well you managed teamwork.

Our team worked cohesively together. We each had well defined roles and executed those roles effectively. We relied mostly on one synchronous meeting at each checkpoint to divide up the work and ensure that everyone knew what responsibilities they had to accomplish by the end of the checkpoint, then mostly working asynchronously until we ran into a challenge. Then we would run more meetings as necessary to overcome challenges and help each other finish all the work that was assigned. We ended up assigning 2 people to work on the frontend side of the application and 2 people to manage the backend / sql side of the application. Then we would have a final wrap up meeting as a team before the end of the checkpoint to ensure that all of the elements we wanted to include in that checkpoint were present and that we all had completed the work that was assigned.

Bose: Worked primarily on the frontend, helped integrate the frontend with the backend and direct which API calls the backend needed to properly interface with the frontend. Also responsible for hosting the frontend.

Amy: Worked primarily on the functionality of the frontend. Primarily contributed in the creation of the frontend and user-facing interfaces of the frontend as well as doing the overall graphic design and layout of the website.

Hieu: Worked primarily on the backend and sql queries. Helped plan the SQL queries and translate them to the API calls that Bose requested for the frontend. Also implemented the advanced SQL functions and helped optimize the backend application.

Daniel: Worked primarily on the backend, GCP and platform engineering. Helped create the GCP instance for the project, both the SQL server and the python backend. Dockerized the backend to allow it to work with cloud run as well as auto deployment with github actions.