Project Report

Team 026 - Benhao Lu, Zijin Zhou, Yuyang Wang, Xin Xu

Changes in Project Directions

- **Synthetic Dataset:** Our initial goal was to build the database with real-world datasets. However, we later discovered that publicly available datasets required extensive data-cleaning efforts, which exceeded the scope of the class. So we shifted to a synthetic dataset instead, focusing more on the frontend and backend development.
- **Data Integration:** Our original plan assumed a static dataset. As the project progressed, we decided to enhance functionality by integrating real-time job feeds via APIs. This change enabled the dynamic updating of job postings on the frontend interface.

Achievements and Failures

- **Achievements:** For the frontend, we successfully designed and implemented a clean and intuitive interface, making it easy for users to interact with the system and access the information they need. On the backend, we developed and integrated a robust keyword search feature that allows users to filter data with precision, enabling users to quickly find relevant information based on specific criteria.
- **Failures:** Some advanced features, such as implementing a machine-learning-based recommendation system, could not be achieved. The primary limitation was the poor quality of the synthetic data, which made it impossible to train an effective model. With access to a higher-quality dataset, we would be able to develop a more robust recommendation system.

Changes in Schema and Data Source

- **Schema Changes:** The schema is similar as originally planned.
- **Data Source Changes:** As mentioned above, since publicly available datasets are too noisy, we use synthetic datasets instead.

Changes in ER Diagram and Table Implementation

- **ER Diagram:** We slightly modified the ER Diagram to better reflect real-world situations. Compared to the tables in Stage 2, we added the *Job Type* attribute to the *Desired* table, since people may have different requirements (e.g., looking for in-person, remote, or hybrid jobs). We also introduced *ExperienceId* as the primary key for the *Experience* table for easy management.

- **Table Implementation:** The table implementation is the same as Stage 3.

Functionalities Added or Removed

- **Added:** We introduced enhanced filtering options, such as job locations, types, and skills. This enables more fine-grained job searches, allowing users to search for jobs more effectively and improving the overall user experience.
- **Removed:** As discussed above, we did not integrate the machine-learning-based recommendation into our system due to poor data quality.

Advanced Database Programs

- **Store Procedure:** The *SearchJobsByKeyword* improves the search functionality on our platform. It allows users to search for jobs based on keywords and preferences. By leveraging this logic, we enhance performance through precompiled queries and reduce the burden on the backend server. This design ensures faster response times and maintains consistency in search logic. Additionally, the use of store procedures abstracts the search logic from the application layer, making it easier to update or optimize in the future without altering the codebase.
- **Transaction:** The *PerformJobTransaction* showcases the importance of transactions in managing complex database operations. This procedure encapsulates multiple actions—such as inserting a new job, updating job details, and logging the transaction—into a single atomic unit. By leveraging transactions, we ensure that either all operations within the procedure are successfully executed or none at all, preventing data inconsistencies. For example, if an error occurs during any step, the database will roll back to its previous state, safeguarding against partial updates that could lead to corrupted data.
- **Triggers:** Triggers like *after_job_insert* automate post-insertion logic, enhancing the robustness of our data layer. This trigger ensures that newly added remote jobs with durations shorter than 24 months are automatically adjusted to meet the platform's minimum requirement. This automation reduces the risk of errors and ensures compliance with business rules without requiring manual intervention. Triggers are particularly valuable for maintaining consistency and enforcing policies across the database, complementing the application's validation mechanisms on the backend.

Technical Challenges

- **Benhao Lu:** Source of Data - The existing datasets we found related to the job market were not well-organized and could not be directly integrated into our database. To address this, we developed custom data generation scripts to create datasets to inject into our MySQL database.

- **Zijin Zhou:** Testing stored procedures and transactions was a challenge for me. My focus was on developing comprehensive test cases to validate the integrity and reliability of these database components. I considered lots of corner cases to ensure data consistency. This process requires detail and iterative debugging to ensure the database operates as expected under various conditions.
- **Yuyang Wang:** As this was my first full-stack project, I found the integration between the frontend and backend to be particularly challenging. Ensuring a smooth flow of data and functionality required me to learn new tools and frameworks quickly. Debugging those issues was time-consuming but highly educational. Despite the initial struggles, this process greatly improved my understanding of full-stack development.
- **Xin Xu:** Since I was unfamiliar with the database or GCP setup previously, connecting to the database and operating it proved to be a challenging task that required extensive configuration and troubleshooting. I experimented with various approaches and ultimately succeeded in establishing the database both on GCP and locally.

Future Works

- **Advanced Recommendation Features:** With more time, we could collect a clean and well-structured dataset to train a robust recommendation model. Leveraging advanced techniques, such as graph-based models, could significantly enhance performance and deliver more personalized job suggestions to users.
- **ATS Integration:** Integrating Applicant Tracking System (ATS) features would allow users to track their application statuses directly within our platform. This would require extensive development and collaboration with hiring platforms and companies. If implemented, it would add significant value and impact.
- **Mobile Application:** To enhance accessibility, we can also develop a mobile-friendly version of the platform. This would allow users to interact with the system conveniently on their smartphones, expanding the platform's reach and usability.

Division of Labor

- **Benhao Lu**: Team leader, CRUD Operations, Advanced Features
- **Zijin Zhou**: Frontend and Backend Development, Advanced Features
- **Yuyang Wang**: Frontend and Backend Development, Keyword Search
- **Xin Xu**: Environment Setup, Dataset, Recommendation