Milestone 2 Report

Vincent Chen, Eunseo Lee, BK Kang, Andrew Lu

Our application is a database-driven web application tailored towards students (Waterloo co-op students are the users of the application) at the University of Waterloo to allow them to explore co-op salaries across a variety of companies, locations, roles/positions, terms, etc, which helps students make informed decisions on which companies and positions they would apply and compare salaries across programs and over time. One of the datasets from the internet we will use is a self-reported co-op salaries datasheet retrieved from a Reddit thread. Additionally we will use similar internship datasets from Kaggle and additional websites to obtain a large enough production dataset to demonstrate the improvement in performance for certain queries. The dataset from Reddit contains companies that offer at least one job posting on WaterlooWorks along with their hourly salary and work term number, and additional information such as benefits, year of job posting, and any companies put on a separate blacklist. However since the hourly salary for a company is likely to be an aggregation of individual job positions/roles within the company and many records within the dataset are empty, we will transform the Reddit dataset into a compatible relational format and populate the empty entries with synthetically generated numbers that reflects the average values observed in the current dataset. This populated and transformed dataset will become our production dataset. We (the project group) will be the administrators of the database system and will maintain the performance and availability of the database, ensure backup and recovery, handle blacklisted companies (via constraints and triggers), etc. The core functionalities we would like to support are: listing salaries by program, faculty, or location, list top-paying companies by role/position, display salary across terms, flag companies with very low salary, provide table and chart-based outputs, etc. In addition to these functionalities, the system also incorporates several advanced features that make use of more complex SQL capabilities. These include full-text relevance search, transactional blacklist entry management, percentile-based salary analysis, blacklist-aware safe employer recommendations, and enhanced views of historical blacklist activity, etc. Together, these features provide students with a more comprehensive and insightful way to explore co-op salary data.

The system support for our application primarily uses Python with the Flask framework for the backend because it is a lightweight framework used for building web applications and is easily integrated with databases. Our application can be run on cross-platform OS such as MacOS, Windows, Linux, or Ubuntu, because these OSes support Flask natively and will facilitate each group member's development on parts of the application. We will use MySQL as our DBMS because MySQL supports the relational data schema we will use, is fast in query processing, allows for advanced features such as indexes, triggers, procedures, etc, and is compatible with the Flask framework through the *mysql-connector* library in Python. Additionally, for data processing we will use the *pandas* library for handling CSV files and formatting the data into relational structure and use *matplotlib* for visualizing the data into a graphical output. For deployment and testing, we will do so on the school server since it facilitates the hosting of our project without too much overhead. Finally, we will use Javascript, HTML, CSS, and React as the lightweight interface for the frontend of our application; it will have interactive elements such as dropdown menus, filters, buttons, etc.

We will get the sample data used to populate our database by sourcing them from publicly available datasets that have information related to internship and co-op opportunities. Our primary sources for our datasets are the self-reported co-op salaries datasheet from WaterlooWorks co-op postings and Kaggle datasets on internships. We will select a small subset of records from these datasets (around 100-200) for initial testing and format them as appropriate for testing our database in the early stages. Later on, our production dataset will incorporate more public datasets from the web and add in synthetically generated data to fill in certain entries. To populate our database with the sample data, we used our own synthetic data because it is easier for testing our database in the early stages and only requires inserting records via INSERT INTO statements in the .sql file; we may also use a Python script to insert more synthetically generated records into our database for further testing downstream.

Our production dataset will be sourced from publicly available, real-world co-op salary datasets such as the Waterloo Co-op Salaries + Blacklist, Kaggle's Internship Opportunities, levels.fyi Internship Data, and various other sources which we will include in the final report later on. These datasets will be processed through a Python data transformation script that standardizes the column names, converts any monthly salaries into equivalent hourly rates, cleans and removes duplicate entries, maps the reasons for blacklists, and fills in missing fields/attributes with realistic values. The script will use Pandas and SQLAlchemy to load the data into MySQL tables and help in resolving the foreign key dependencies between tables.

The four members of our team and the work they did are:

- Vincent Chen: Worked on designing the database schema, E/R diagram, all of R5, and wrote SQL files for creating tables, constraints, and triggers, etc (C2), and R6.

Added more features in application code, updated SQL files to reflect any schema changes (C2), and worked on R10, R11, R12.

- Eunseo Lee: Worked on test-sample.sql file for Task 5 and the test-sample.out for testing our sample database (C3) and R8.
  Updated test-sample.sql and test.sample.out files to include all 5 basic features (C3), added more features in application code, refined all the sections from M1 to reflect project's progress, and worked on R13.
- BK Kang: Worked on setting up the MySQL and Flask environment for our project on GitHub, built the beginning parts of both the frontend and backend of our application, and implemented simple functionalities regarding the database (C1, C5) and R9.
  Worked on Production Data Plan (R4), added more features in application code, updated README file, and worked on R15.
- Andrew Lu: Worked on the Milestone 1 Report (R1, R2, R3) and R7.
  Updated README file, worked on Production Data Plan (R4), test-production.sql & test-production.out files (C4), and R14.

GitHub repository: https://github.com/bkctrl/cs348-project