Lab 4 Report
Group: 8
Justin Yearwood (1007854608)
Alicia Barlow (1008061918)

1.0 Design of Enhanced Search Engine

On the front-end side, an autocomplete feature was implemented to enhance the user experience of the application. When the user starts typing, a dropdown appears under the search bar, showing a list of suggestions that match the user's query. These suggestions in the dropdown change according to the user's input in the search bar as they continue to type. This feature was implemented by adding a new endpoint called '/autocomplete', which retrieves the user's current query in the search bar as well as a list of all existing words in the lexicon table in the SQLite database. The user's query is compared with every word from the lexicon table, and it is added to a list called "suggestions" if the lexicon word starts with the user's query. A JavaScript script was also created, which includes an event listener that triggers every time the search bar input changes. This script makes a request to the '/autocomplete' endpoint to retrieve the "suggestions" list. The words from this list are displayed to the user as a dropdown.

We enhanced the backend by integrating fuzzy logic to identify the closest search matches when user queries do not exactly match stored terms. Previously, searching for "directory" would fail to return pages labeled "dir," but with fuzzy search we can now resolve near matches and return significantly more relevant results. To maintain efficiency, the fuzzy step is only executed when the base search does not yield strong matches, ensuring minimal unnecessary processing.

We also implemented persistent caching, allowing every query to be stored automatically. This optimization delivered more than a one hundred times speedup for repeated searches, reducing them to the cost of a single dictionary lookup. This improvement is essential, as it enables us to support more sophisticated search heuristics without making the system slow or resource-intensive.

These features came together to enhance the overhaul search process by complementing each other to function like the typical search engines we are accustomed to.

2.0 Difference Between Proposed Design and Completed Design

There wasn't a significant difference between the proposed design and the completed design, as we always strived to create a Google-like search engine, with a similar styling and user interface. We feel that we were able to achieve this goal, as the final product resembles Google in terms of visuals and usability.

3.0 Testing Strategy During Development

During development, we would first test the application manually by interacting with it ourselves, trying different inputs on both modes (anonymous and logged-in). The majority of bugs were found this way, and it also helped with adjusting the UI of the application to enhance the user experience. We also created unit tests to ensure that specific components of the code work as intended. For corner cases, we tested the search engine with "extreme" values such as empty inputs, long inputs, inputs with multiple results, inputs with no results, etc, to confirm the functionality of the application for even unusual cases.

4.0 Lessons Learned

One of the biggest lessons I learned from this project was gaining hands-on experience across several real-world technologies: user authentication (especially Google OAuth), routing, unit testing, working with a SQLite3 database, and deploying our system on AWS EC2 instances. Before this project, many of these tools felt abstract, but implementing them strengthened our understanding of full-stack development and cloud infrastructure. If we had the opportunity to spend more time on this project, we would focus on improving concurrency and scalability such that the system could handle many simultaneous connections. During the benchmarking portion of the project, we were consistently stuck at <30 concurrent connections. There is significant potential for optimization in areas such as request handling, asynchronous workflows, etc., and exploring these components further would be a valuable learning experience. Some parts did take longer than expected. For example, getting OAuth set up correctly, configuring EC2 instances, and dealing with environment differences between local development and remote servers. These tasks required more troubleshooting and experimentation than anticipated, but they ultimately taught us valuable skills in debugging and general problem-solving.

5.0 How the Material From the Course Helped with the Project

Although much of our implementation relied on external resources such as official documentation, StackOverflow, YouTube tutorials, and online articles, the material from the course still provided a helpful foundation. The early Python review gave us the baseline familiarity we needed to write clean, modular code, and concepts like lambda expressions and list comprehensions came up frequently while building our system. The introduction to relational databases and SQL also helped us understand how SQLite worked and how to structure and query our data effectively. Overall, even though the project required many tools and technologies that extended beyond the course content, the fundamental principles we learned in class made it much easier to approach new documentation and quickly understand unfamiliar concepts.

6.0 Required Time for Each Lab

We typically started the labs on Saturday, and we would work into the next day to complete and submit the lab on Sunday night. In other words, it usually takes us two full days to complete each lab, during which the lab is our sole focus.

7.0 Highlights of the Project

Setting up EC2 instances and interacting with AWS was an interesting learning experience since AWS is a relevant tool for real-world applications. Learning how search engines work and creating one throughout the semester in general was a very practical learning experience as it meshed together various components of full-stack development (building UI, managing a database, designing routes, writing unit tests, structuring for deployment, etc.) One recommendation we have for future labs is to use a more up-to-date crawler.py file, as the one provided for this lab was from 2011 and included deprecated modules and outdated comments, making it difficult to follow and understand during Lab 1.

8.0 Useless Parts of the Project

Overall, we feel that the project was very useful as it covered various aspects of full-stack development, and there were no parts that felt “useless”. Having the opportunity to use real-life frameworks and tools such as AWS, SQLite, etc., was a valuable experience.

9.0 Feedback/Recommendations for Course

A recommendation that could be made for the course is to provide more practice material in preparation for assessments. Although the assignments were useful practice, we felt unprepared, especially for the midterm as we felt we didn’t have enough variety in practice problems to enhance our skills and deepen our understanding of course concepts. This was particularly true for einsums, as we feel we weren’t properly taught how to solve problems relating to einsums in class, and the lecture examples that were provided were much simpler than the questions on the midterm. Furthermore, this topic was not covered in the assignments, and since there were very limited online resources to refer to for practice due to the subject being relatively niche, we felt very unprepared for the midterm despite doing the best we could with the resources we had at hand.

10. Responsibilities for Each Member

Alicia worked primarily on frontend, and Justin on the backend. Both members contributed to debugging and writing the scripts for deployment and termination.