**Intelligent Search**

Sidhant Chitkara, Alexander Meyer, Apoorva Parmar,

Yash Pujara, James Shao, Michael Vieck

**What went well?**

Our team had to spend some time learning Angular as well has how to call the backend functionality from the application. While this learning curve took some time away, there were still several aspects of the project we were able to complete that we believe turned out well.

Much of the layout of the Web application was completed and is designed in an intuitive, simple, and appealing manner. We feel that despite having the different activities and views split among us we were able to effectively work together to create a cohesive theme for the application.

The server set-up for the front end web-service went well. The maven auto-build of the .war file makes deploying servlets very simple and saved a lot of time. The API calls are very straightforward to implement and made the front-end calls very simple.

We underestimated the power of the lexer, and the ease with which we could program it. This allowed us to support more user stories and more searches than we thought would be possible. The 2 people working on the lexer also had previous experience with lexical analyzing, and this allowed us to develop efficient methods of extracting tokens.

It was easy to integrate the queries to the back-end. The tokens provided allowed us to simply pass things like a location to the API. Displaying the results dynamically on the front-end was difficult, as the amount of information was a lot for one page. Styling the cards to flip over and reveal more information was an elegant solution we found to combat this problem.

The back-end and front-end implementation for logging in was completed without too much difficulty, but changes had to be made when they were integrated together. The integration was difficult, but manageable, and the issue was resolved.

**What did not go well?**

The biggest issue that we ran into was the integration between the front end, back end, and lexer teams. We initially had thought that it would be a very simple few lines of code, but eventually realized that there were other issues, from server configuration to java file package incompatibility that would take time and resources to fix. While integration ended up working well, the time it took was very wasteful and something that has to be improved upon.

Another major issue was the user preference user stories. All of these user stories did not get completed because we underestimated the amount of time the base user stories would take for the front end. While the back end had all of these features complete, the front-end was swamped for time, mainly because of inexperience and integration issues. These user stories were things like users would like to be able to set favorited items by “pinning” the item or dining court card and then it would display under favorites. While the front end functionality exists and so does the back end, the integration had not yet been completed. On the other hand, user food preferences and user settings pages had not even been made at the end of the sprint and remain to be completed for sprint 2. These user stories have been pushed back to Sprint 2, however, there were definite issues here that should have been addressed earlier.

A minor issue would be the login functionality of the website. While users are able to login with their accounts, a user session (cookies) is never set. Additionally, there is no option for a user to logout or access their user specific preferences. Furthermore, user account passwords are not hashed / salted and makes our database relatively unsafe for the time being.

**What can we improve upon?**

Communication is something that we will have to improve upon for next sprint. There were a few communication issues between teams and even within teams. The major issue was communicating between teams during integration. Knowing what other teams were doing would have helped integration go more smoothly and probably saved a lot of time. This is something that can be fixed through team coding sessions. We ended up having a few team coding sessions near the end of the sprint, but some sessions at the beginning could have helped as well. Communication among feature team members is also something that needs to be worked on because the sub-teams need to be in constant communication with each other in case one person needs help and their team member has already encountered something similar.

As mentioned earlier, group coding sessions will be used next sprint because it helps us finish our user stories faster and integration becomes less of an issue. When everyone is present and working together, issues get fixed faster and teams have a better idea of what to expect from the front/backend.

Another improvement is researching all of our options thoroughly first. We ran into language issues for the backend where we initially planned on doing it in C++ before realizing that Java was far easier for both the servlet technology and the lexing. Prior research on both teams parts would have made this decision faster and therefore saved coding resources and time.

The lexer was relatively easy to implement, but only because we used logic that isn’t too complex, efficient, or robust. Changing the way we find tokens in a future sprint would allow us to support more robust user searches, without any other changes being needed on the front or back-end.