Laboon Capstone Fall 2017

Midterm Update: Yelp

Randyll Bearer and Zachary Barlotta and Raj Patel

October 20, 2017

**Introduction**

Since our initial project proposal, we have firmly progressed from planning to implementing. As of the time of writing this Midterm Update, we are currently coding our first feature, the ability to conduct a “vague search” on the Yelp database. As outlined in our project proposal, this “vague search” will allow us to accept generic terms and/or keywords from our users and then query the Yelp database using these terms. For example, “vague search” will allow us to handle requests like “Find pizza near me” and return an appropriate option. We have hit some snags while implementing this feature however, all of which will be discussed further in this Update.

Subsequent meetings with the Yelp/NoWait team have continued to be fruitful, helping us solidify and finalize our prioritized list of user stories. We have established a schedule for future meetings in accordance with the Agile Scrum methodology (i.e. one every two weeks), shared our GitHub repository with the team, and have created a Slack channel for further communication. Our liaison within the company has shifted from Dean Thompson to server engineer Steve Blass, to whom we presented a simple demonstration of our software last Thursday (October 12th). The demonstration’s purpose was to show our ability to connect the individual pieces of our project. It accepted a simple voice command, sent that command to the Google Assistant, fulfilled that command through a basic API call, and then verbally responded with the result.

After we complete the implementation of “vague search”, we intend to tackle the following in order: “direct search”, “random”, and “wait time”. We have prioritized “wait time” last as it requires access to the non-public Yelp API, which the team will grant us access to after we have proven ourselves capable of implementing the rest of the other features. Our main focus for testing is to rely primarily on black-box testing, and allowing various members of the Yelp/NoWait team as well as our own friends and family to conduct conversations with our software. This way, we will be better able to see the different kinds of conversations our software should be able to handle, as well as restructure some of our parsing techniques to better match the most frequently stated requests.

Lastly, In our project proposal, we discussed that we would be using Google’s API.ai software to develop our Google Home actions and intents. However, Google switched from API.ai to Dialogflow on October 10th, and while the basics of Google Home development have remained the same (It was mainly a rebranding/renaming change) we have been required to relearn a few features, but it should not pose a significant setback. We are simply listing it here to keep you up to date on the software we are using.

**Current Issues**

As stated in the Introduction, we are currently in the process of implementing the “vague search” feature. However, we have come across some unforeseen circumstances while doing so. To recap the fundamentals of our project, it is composed of an action, an intent, and a fulfillment. The Google Assistant/Home constantly polls for a user request, attempts to match that request to an action (specific program), determines which of the action’s intents can best handle that request, and then relies on the fulfilment of an intent to conduct the business logic. The fulfillment logic is required to be cloud-based, all of a fulfillment functions are hosted remotely.

To deploy our fulfillments, we have been using Google’s Firebase service. Easy access to this service was one of our major reasons for choosing to program in Node.js. And while we *can* deploy basic fulfillments to the cloud, any fulfillment which includes receiving a security authentication token from the Yelp Fusion API proceeds to crash our entire action upon simulating it. It is important to note that the same code, when ran locally, works perfectly fine. It is only when the code is ran remotely that we are receiving errors.

To work around this, we are currently investigating a couple options. The first is a switch from using Dialogflow to using the Google Actions SDK. The second is that we are looking into foregoing the Node.js client library and instead manually structuring our JSON objects and manually writing our POST/GET requests to the Yelp Fusion API endpoint. Ultimately, we are just trying to find a way to contact the API which is compatible with Firebase. If none of these work then we will be forced to switch to a different deployment service than Firebase.

A major difficulty we are facing when trying to resolve this issue is a lack of adequate debug tools. The Google Developer Console, which is currently the only way to simulate and test our action, provides very little debugging information by itself, providing only the contents of the JSON received from the fulfillment. Since the fulfillment is causing the error, the JSONs are corrupted and not providing any useful information. We have found rudimentary ways to get debugging information from the Firebase service itself, but our methods still leave a lot to be desired. We will obviously continue to look for better debugging methods, as debugging tools will, if the Yelp/NoWait team continues to pursue the project after us, help future programmers immensely.

Even with all of the errors that we have encountered throughout this project, there is some hope at the end of the tunnel. One thing that has been going well for us is that our code runs successfully in a native setting. It’s always a good thing that our code at least runs somewhere, rather than not running at all. We have also had very helpful and steady communication with the staff of Yelp/NoWait. We have had several meetings with them, and each has gone successfully in terms of communication and advice. We have also set up a Slack channel with them in order to keep precise communication with them throughout the project. Out liaison, Steve, has been especially helpful and informative through the communication through our Slack channel. Learning Javascript has also gone well for us. We picked up on the language rather quickly, and were able to get a working Javascript program rather quickly. We also found it pretty simple to make API calls using Node.js natively. This has really helped us in implementing our design as well as testing it. All things considered, all of the things that have gone right for us could help us fix the things that have gone wrong so far, and that will help us build a very strong project.

**Conclusion**

Going forward, we have a plan laid out for how we will attack the rest of our implementation. First off, we will fix the issue of the functions not running remotely. If we cannot get them running remotely, then we cannot finish the project, so this is our highest priority going forward. Once we fix this issue, we plan to implement the rest of the features we have laid out in the project proposal. The rest of the features are no more complex than what we have done so far, so we should be able to implement them rather quickly once we are able to get the functions to run remotely. Once everything is implemented, we plan to do further black-box testing to find potential bugs within the program that we may not have caught before. Once we feel like we have fixed the potential bugs, we plan to do user tests. That is, we plan to test our program to see how users communicate with the device. This will help us collect some data on how different users interact with the device, and we can use this data to refine our conversation hooks/cycles. All things considered, our road ahead may be a challenge at first, but it will get better once we fix our issue with the functions not running remotely.

In conclusion, we have firmly moved away from the planning phase, and we are currently in the implementation phase. While we have had issues with implementing the program, we have not moved away from the process itself. That is, our issues stem directly from the implementation of the program, not the process itself. We are still using the Agile/Scrum process, and we plan to keep using it for the remainder of this project. For the rest of our current sprint, we plan to fix our remote functionality issues. Once we fix those issues, we will be able to move on to adding the remainder of the features we plan to add to the program. All things considered, there are things we have done well with this project, and there are things we have not done so well with this project. As time goes on, we will have this project running smoothly, and all of the issues that we are having with the current implementation will be solved in the near future.