

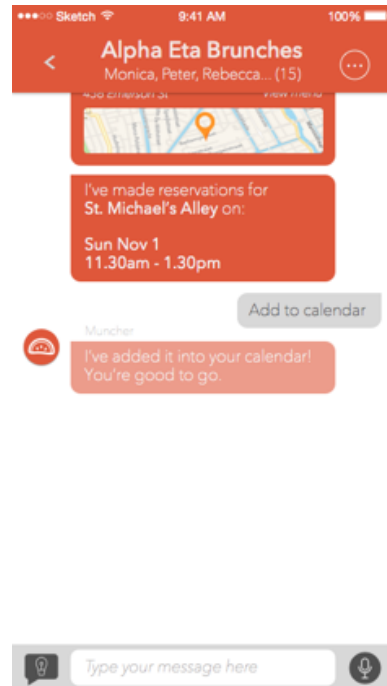
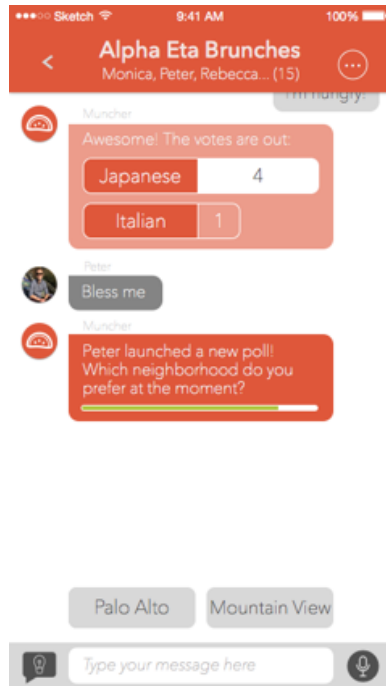


CS 147 Final Report

Chua Kai Jian
Gloria Chua
Monica Yupa
Peter Farejowicz

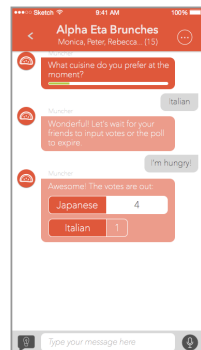
Problem and Solution Overview

When groups of friends go out to eat, finding a place everyone agrees on can be difficult. Muncher's mission is to help you decide where to eat in groups, removing the frustration in the process. With the help of *human-based artificial intelligence and a messaging user interface*, Muncher understands your preferences and makes the hard decisions for you. It is like having a genie in your pocket, so you can fuss less, dine more.

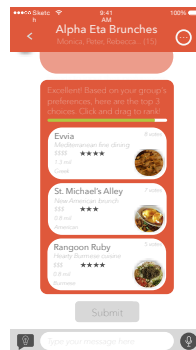


B. Deal with user discontent (complex)

Since it is likely that there will not be full consensus on a particular decision, groups of friends need to be able to iron out any disagreements and discontent in the decisions Muncher has made.



Showing the poll results increases transparency as a means to deal with discontent



Having a drag & drop ranking system allows users who did not get their top choices to still express their opinions

C. Coordinate the actual plans (simple)

In this task, the users figure out the logistics relating to any auxiliary action related to eating, such as making reservations and adding the appointment to one's calendar. We decided to include this task to complete the user flow from planning to action. (Storyboard on the next page)



If the majority of the group rejects the choice, the next best choice appears.

Make restaurant reservations directly

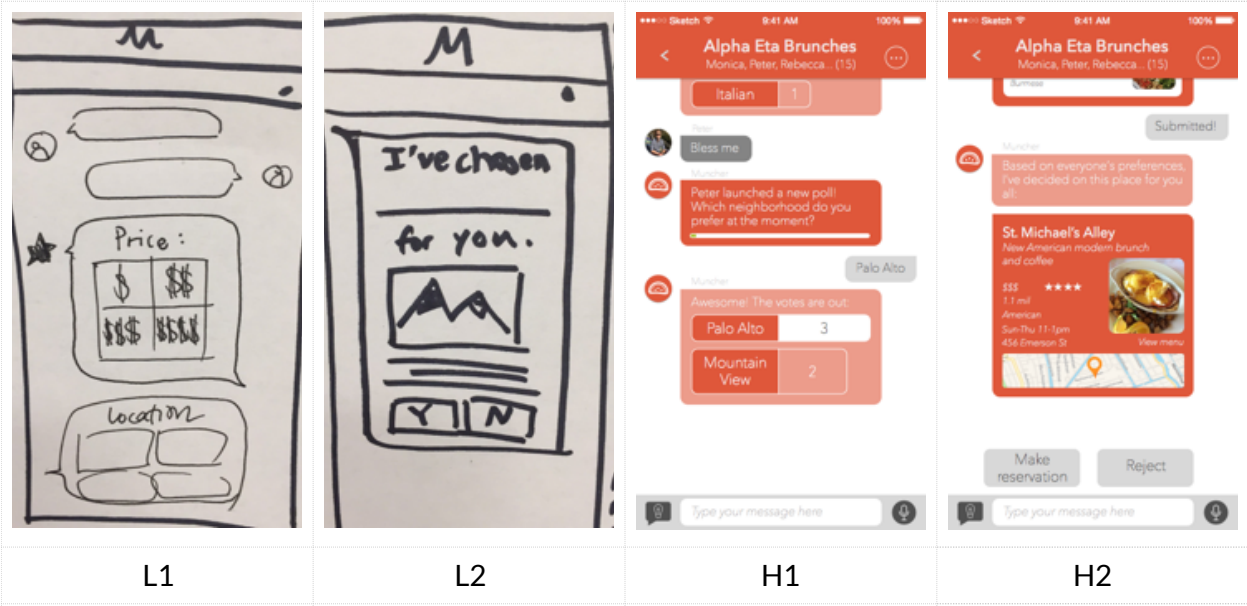
Add appointment to one's personal calendar directly

Design Evolution

Through user-testing our low-fidelity prototype, we gained many deep insights about our initial sketches. These were super helpful in guiding our subsequent design evolution through the medium and high fidelity prototypes, in terms of AI presence, process flow and optionality. Overall, the interviews helped us realize that Muncher needs to stick to a simple interface flow to avoid user frustration and decrease time spent deciding where to eat.

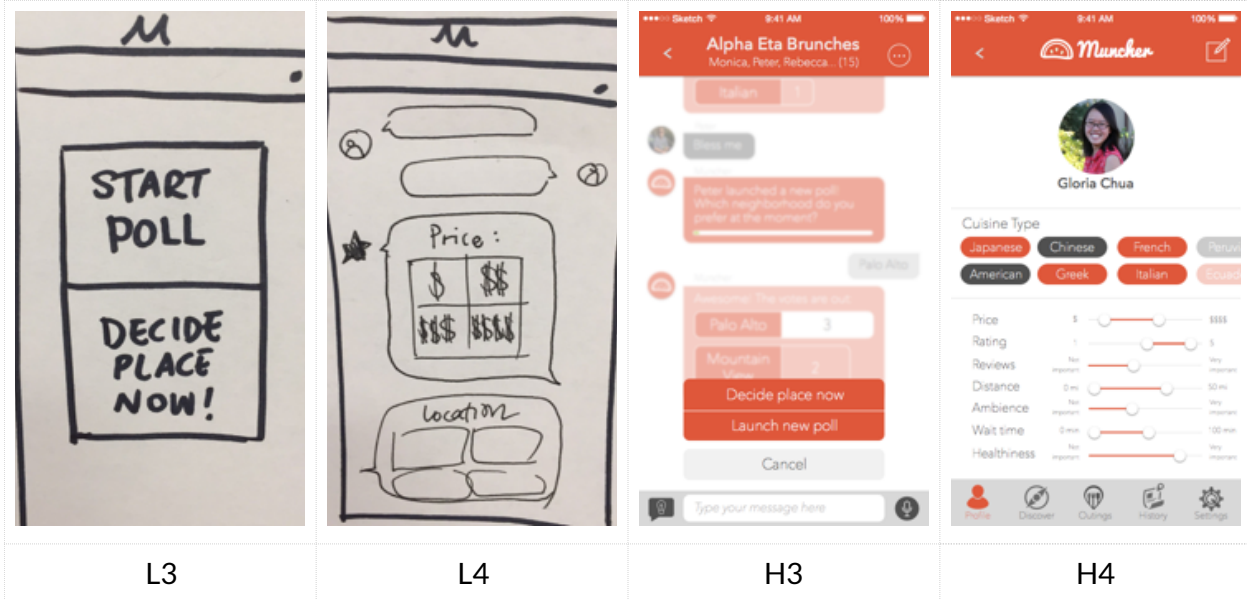
A. The idea of a ‘Muncher genie’ was well-received, but not immediately apparent to users

Users noticed the genie presence in the group chat (L1), but not when they were presented with the restaurant suggestion (L2). Hence, in our med-fi and hi-fi iterations, we kept and augmented the AI genie concept, and made the genie speak like a human being, so as to ensure that the genie’s presence is felt whenever the user is faced with a decision (H1, H2).



B. Users found redundancy in expressing preferences

The app requires users to input their preferences when the group is first started, then the genie asks for preferences again during the poll (L3, L4). This repetitiveness makes the process laborious and frustrated users. Hence, we decided to de-prioritize the polling and encourage users to go straight to deciding a place by moving the “Decide place now” above “Launch new poll” (L3, H3). We also made an assumption that at the backend, the preferences entered in the “Profile” (H4) would be automatically imported into the group created, so there is no need to gather information from users about price, rating, reviews etc. through polling.



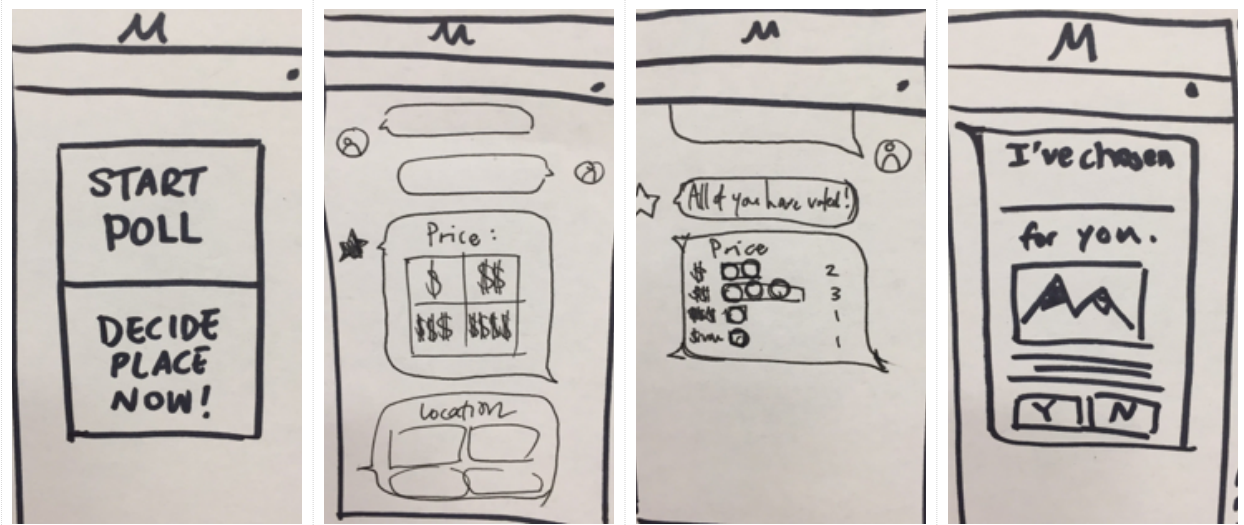
C. When the genie decides on a place for the users, users wanted multiple options instead of one

We noticed that users had the most trouble with figuring out the prototype flow when they did not agree with the first restaurant suggestion; they found it a hassle to keep going back to poll or decide on a new place to eat.

D. Users only had the patience for up to 3 restaurant suggestions.

After 3 different suggestions, most just wanted to finalize a location, as the app started to feel repetitive. They suggested more restaurant options up-front and less back-and-forth, especially when they prefer spending more time looking at food pictures and restaurants than inputting preferences.

Taking C and D into consideration, we introduced “restaurant cards” (H7) that will be curated based on the automatic collation of all users’ profile preferences (H4) when “Decide place now” is pressed. Next, instead of just giving one option, the Muncher genie shortlists the top 3 restaurants, and give each user the chance to rank them. This provides an extra layer for making all voices heard, so that users whose choices did not make the top 3 would still be able to express their preferences, knowing that the 3 restaurants represent the group’s collective decision.

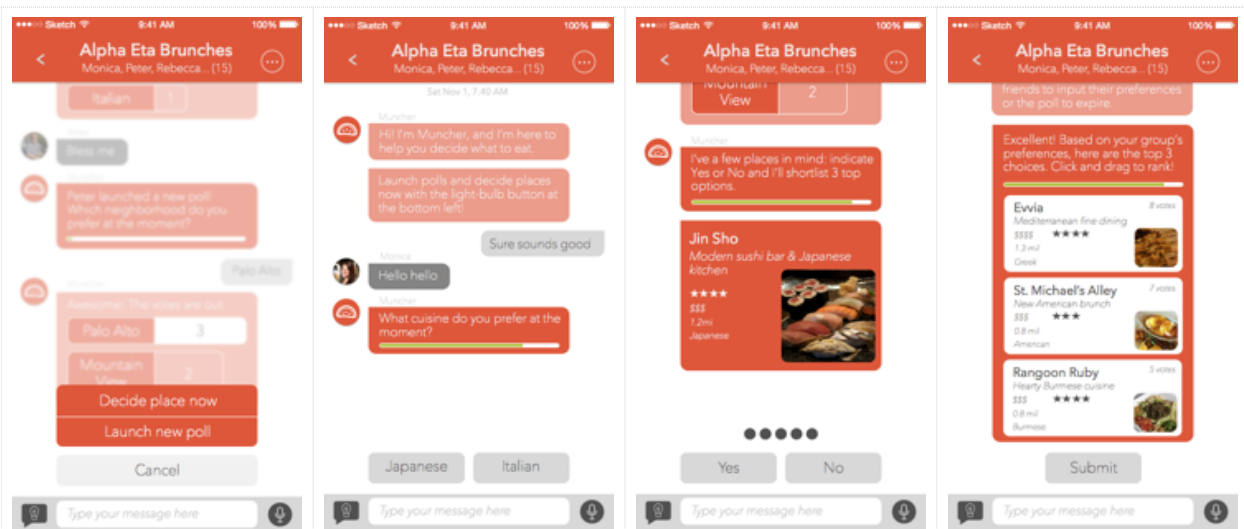


L5

L6

L7

L8



H5

H6

H7

H8

Addressing Usability Issues

1) H2-2: In the profile, it is unclear what the ranges are for the values, so the user could not properly gauge their interests

- Suggested fix: Two indicators on the bar, one for min and one for max
- Status: Fixed. Having two sliding indicators allow users to express a range for each preference, instead of just a point



2) H2-3: Binary options on polls doesn't allow for user to express opinion if they don't like either options

- Suggested fix: provide another option to allow for user input
- Status: Not fixed. The premise of Muncher is limited choice, so we do not want users to input too much. The concern of opinion expression is addressed by having the option to reject the final decision, which will then pull out the next best restaurant option.



3) H2-3: No way to return after modifying settings with the (...) button.
Clicking back on the conversation is not intuitive

- Suggested fix: Consider a < button
- Status: Fixed



4) H2-5: Booking a reservation, which is a big step, is only done with one tap

- Suggested fix: Presenting a confirmation button before committing reservation
- Status: Fixed, by having an extra layer of confirmation



5) H2-5: It would not be convenient to repeat the entire process if the restaurant is not open or unavailable for reservation

- Suggested fix: N.A.
- Status: Not fixed. We are using Wizard of Oz techniques here and assume the backend will deal with this, not the UI.

6) H2-6: On outing info page, there is no clear distinction between actionable vs. non-actionable items. Users have to memorize settings that are clickable

- Suggested fix: Make clickable items prominent and different
- Status: Fixed, by making clickable items have huge icon and > button



7) H2-7: In the drag and drop part, it doesn't seem like the up and down arrows have a function

- Suggested fix: Remove redundant arrows
- Status: Fixed



8) H2-9: Chat-based systems aren't irreversible, so there's no way to undo an accidental vote

- Suggested fix: N.A.
- Status: Not fixed. Multiple users input multiple votes, so accidental votes do not have much impact. The undo feature is not implemented for simplicity, and users can still reject restaurants at the final step. This might be something we can look into in the future as a premium feature

Other changes

1) Included cut-off time for each action item and decision so that Muncher can move the decision-making process along, if there are users who do not check the app or respond

- This time is shown by the moving green bar. Once the bar hits full, time is out



Prototype Implementation

Tools

We started out making the hi-fi prototype using Sketch for design and Xcode for the implementation. However, we quickly found that the learning curve for the language Swift was too steep for us, without having anyone who has prior iOS development experience. As a result, we switched to the hybrid tool, Ionic, as our framework because it gave us more flexibility and slightly more familiarity.

We switched from Xcode to Ionic as our framework because it gave us more flexibility as to the hybrid approach. Ionic made the implementation easier. It was faster to learn the angular.js language used in ionic than the Swift language, and we could straight up code the Ionic app whereas Xcode had us doing storyboard stuff in a graphical way.

For our languages, we used HTML, CSS, and JavaScript. We also imported the angular.js and ionic.js libraries and used Sublime as our editor. Lastly, Ionic's iOS simulator helped us visualize things.

Wizard of Oz

Our solution concept depended heavily on an AI backend that we did not have time to build out. As a result, we smartly employed Wizard of Oz techniques throughout to make the Muncher genie respond to user input in a simple and believable manner. For example, Muncher is supposed to provide suggestions based on the users' profile preferences, but we just made the assumption that our app did, and restaurant suggestions are based on the group's overall preferences, not just the individual user's. Furthermore, Muncher gives very generic text replies in response to whatever the user types in, so as to mimic an actual conversation as a genie.

Hard-coded Data

Similar to where we applied Wizard of Oz techniques, we hard-coded Muncher's responses as well as the restaurant cards that appeared.

Future features

Our ambitions for Muncher have grown beyond the class. The most basic of features to implement next is having integration with phone contacts and/or Facebook so people can actually use the app with their groups. Most importantly, we need to build out the backend AI to have a functioning Muncher genie that is at the core of our solution. Beyond these, we also hope to, at some point, implement the Discover and History features, if we do choose to continue pursuing this project. Discover would have a Tinder-like UI/UX, where users get to swipe on restaurants individually to discover new dining places. History would be where users can look at places they have been to, and we will focus that feature on user-driven content creation, such as reviews, social etc.

Conclusion

It has been a wonderful journey with a fantastic team. Our solution concept for tackling the problem of deciding a place to eat in groups has evolved throughout the last 10 weeks in this class as we went through the rapid prototyping and user testing phases. This allowed us to hone in on a pain point of many people, and decide on a Muncher genie as the backbone of our value-added solution. As we moved from the med-fi to hi-fi prototype, the heuristic evaluation feedback allowed us to discover usability issues and design errors. We acted on these fixes to refine our UI and tighten the process flows for users of Muncher. Although we have come a long way, there is still much to work on for the future, but we are thankful for all the help and feedback throughout this project development process that have allowed us to come so far.