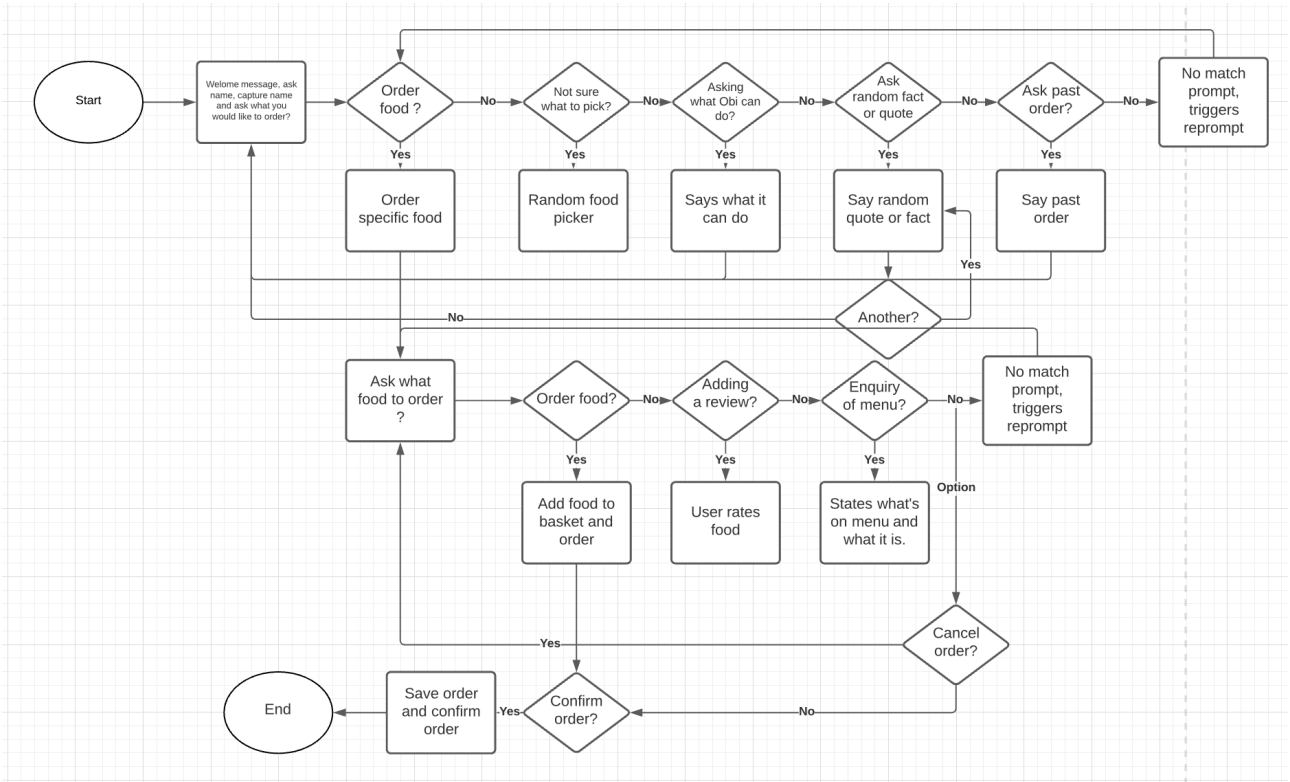
CW2: Voiceflow Prototype

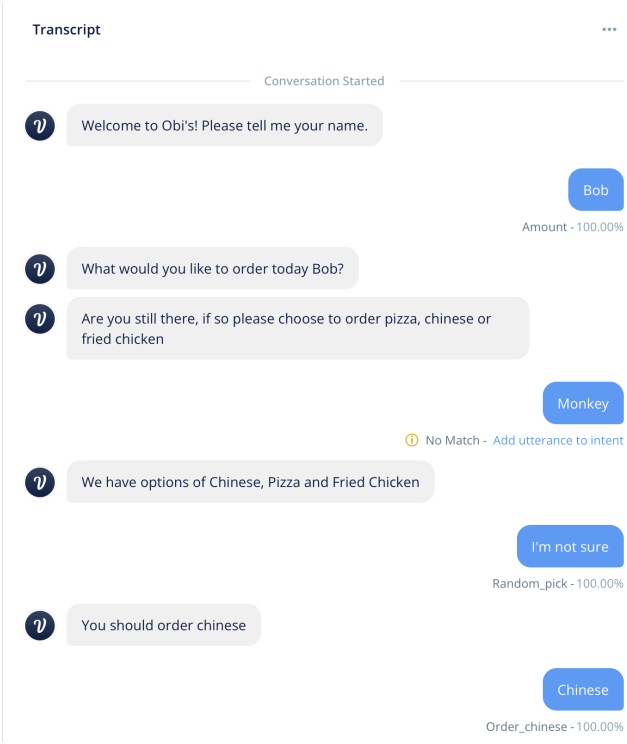
20171111

The prototype:

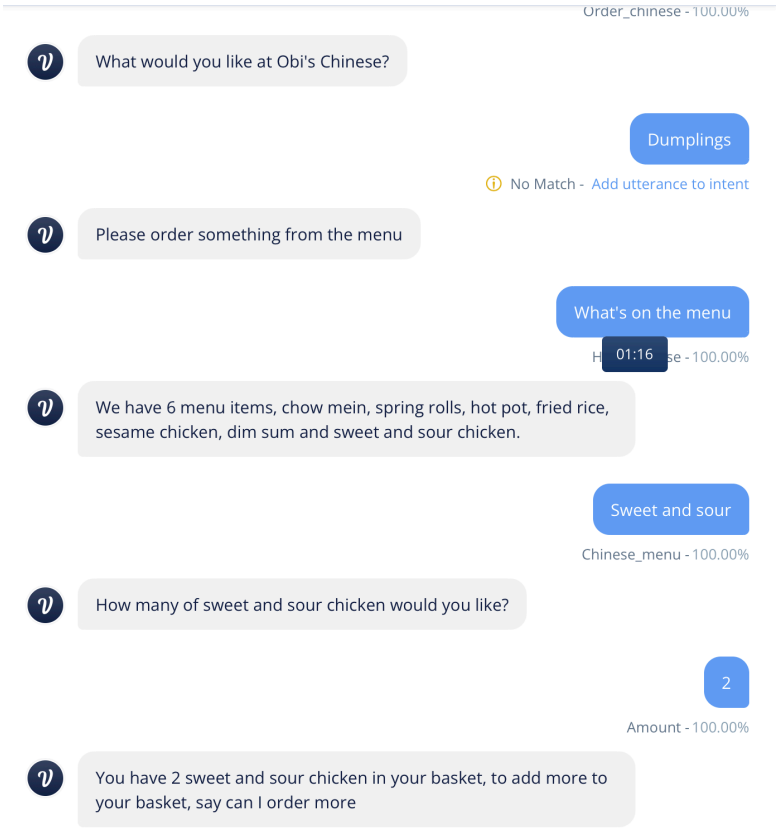
My prototype's name is Obi, it is a conversational prototype that has a main function of being able to order cooked food. My prototype is built on the domain of Amazon Alexa. My prototype gives the user options to order from 3 different types of cuisines, (Chinese, Pizza and Fried Chicken), in each type of cuisines there are at least 6 menu items. The user is generally able to order quantity and size of the food item they have picked. If the user is unsure about the food items, they are able to enquire more about each food item, Obi will tell the user the specifics of the food items. Obi is able to make basic transactions, such as confirming orders, and the ability to query what the status of the order is, generally how long for the food to come. Obi's also makes use of two API's which allows the user to ask random quotes, and also a separate API which allows the user to ask random facts. The system also uses google sheets integration to enable the user to query about their past orders, and also allows randomisation to allow the user to ask what cuisine they should order if they are not sure.

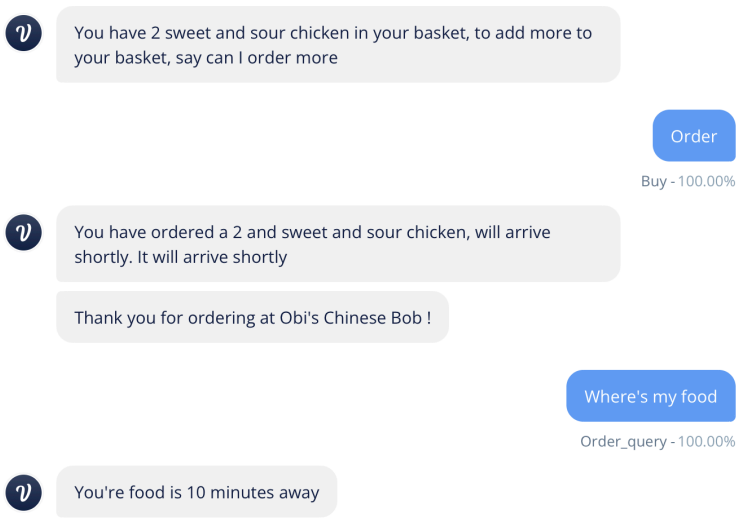
Design process:



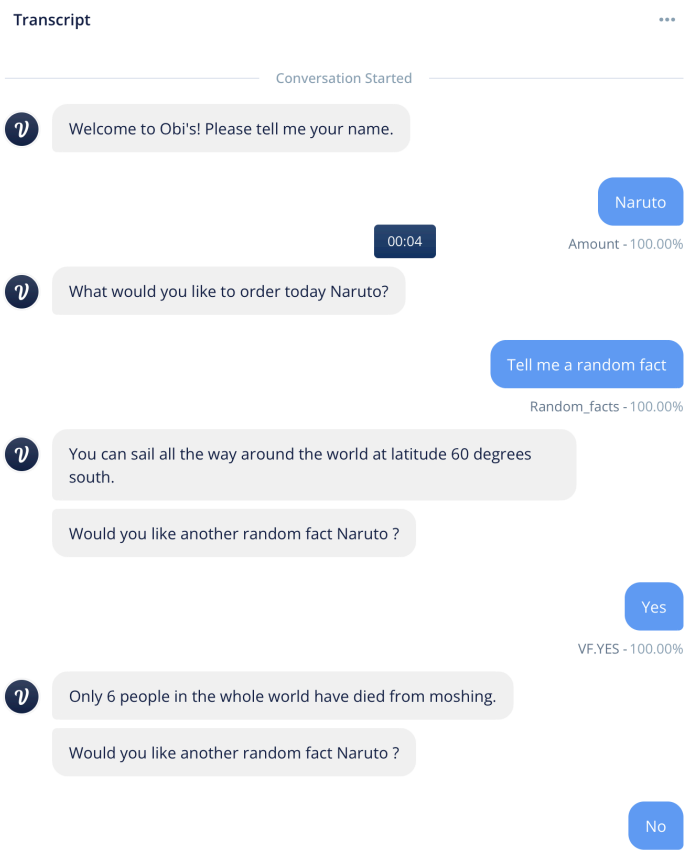


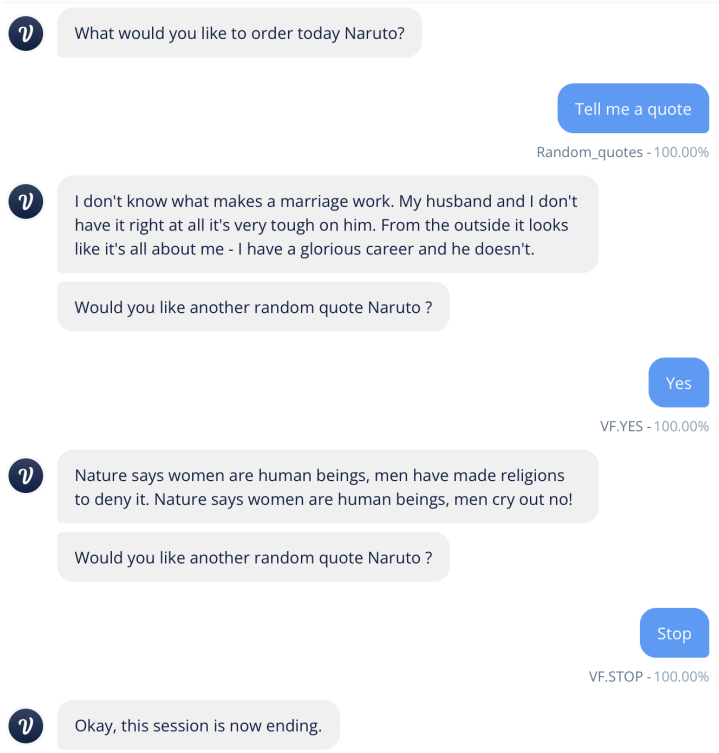
Conversation 1 involves ordering some Chinese.



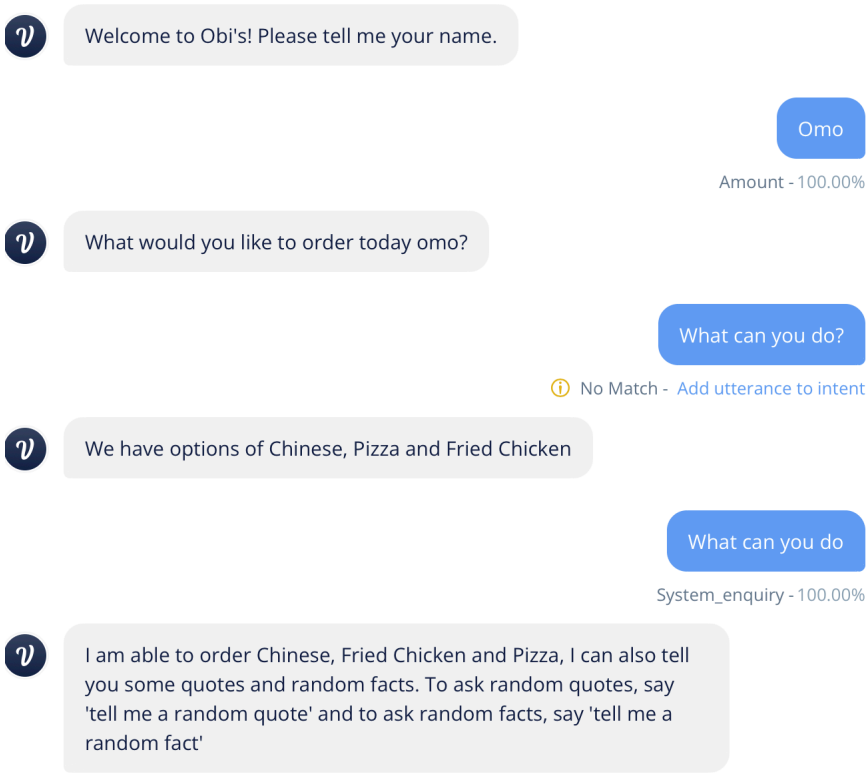


Conversation 2 involves asking random facts and random quotes





Conversation 3 asking Obi what it can do, it had a no match as it contained a question mark.



I took a combination of design processes when looking to build this VUI. For example In discoverability support I used both explicit prompts and explicit requests, at first the user can use an explicit request to know what the entire system is capable of, if they are not sure they are then guided down a path. My VUI also uses recovery, repair and progressivity, it lets the user know wrong input, and also tries to move the conversation forward by adding by giving specific options. This is how I constructed my response design. I tried to ensure that the information is delivered minimally a majority of the time, the users when testing were able to follow the VUI's response. Examples of this can be seen in the screenshots above and also in the diagram.

VUI Design Principles

Prompt Design:
Obi has clear prompts which allows user to understand what they should do next, for example in the first conversation, there are clear and concise prompts such as "what would you like to order", "please tell me your name' and so on. You can also see, that if the user says something foreign to Obi's system, it uses re-prompts which are more focused such as ordering specific options of "Chinese, pizza or fried chicken", only if the user enters something incorrect. These re-prompts tell the user what is expected and guides the user to a more desired response.

Discoverability:
The system is quite guided, it guides the user down the path of choosing a cuisine to order, choosing what food item to order, how many and confirming or choosing to order more. However, the user can ask the system at the start, after Obi has taken their name "what can you do", this will be followed by Obi telling the user all the things which it can do. Also if the user is unsure of what's on the menu it can ask Obi, and Obi will tell them what's on the menu. These features allow the user to discover what the system can do. Examples of this can be seen in conversation 3.

Error Handling:
The main strategy of error handling that Obi uses is re-prompts. There are up to three re-prompts whenever targeted input is required, the re-prompts are progressive, as they go on the re-prompts become more specific of the input, that they want, and also are more clear. These re-prompts usually guide to ask for help or guide to specific options which the user must pick.

Personalisation:
Obi stores the users name and addresses the user a few times throughout the confirmation, usually during more important speech. The VUI also saves the users previous orders into a database, this allows the user to ask Obi what their last orders were.

Confirmation:
I believe Obi's uses the right amount of confirmation, I believe over confirmation would be more harmful than good when it comes to ordering food as it would be annoying and quite tedious after every time you add something to your basket, that you would need am explicit confirmation. The system makes use of generic confirmation, this can be seen when choosing food items and the amount of food items. Obi also makes use of explicit confirmation, this can be seen when the user is asked if they want to order what is in their

basket. The VUI also makes use of non-speech confirmation quite frequently, for example when the user asks for a random fact, a random fact is given or when the user asks what's on the menu, the menu is given.

Context:
Obi makes use of basic context tracking, for example it tracks the food item after being asked the amount of the food item it uses. However, extensive context tracking is not used, Obi struggles to understand and match pronouns to food items or other things related to context tracking.

Integration:
The VUI makes uses of API integration and Google sheets integration. It uses sheets integration in order to add to personalisation, the sheets are used to save the orders into a database like system, sheets is also used to show the user their past orders. Obi uses two API's, one for random facts and another for random quotes, this gives Obi another element to it other than just food ordering.

Advanced Features

Obi makes use of more advanced features through its integration with two API's and also its integration with google sheets. The two API's help make Obi not just a food ordering VUI but also a VUI that can support some information retrieval, some entertainment factor and maybe even classify this as some small talk. The API's allow the user to query random facts or random quotes, and it gives the user a different result each time as the API's make use of large databases. The integration with google sheets allows the VUI to become more interactive and personal with the user, if the user doesn't know what to order he can ask the VUI, all orders are saved to the google sheets once they are confirmed and the user can ask for a past order.

Conclusion

Obi is a functional food ordering VUI, I am proud that it works successfully and guides you with what it can do. I believe it is a VUI that is easy to use and also tries to limit errors and has good error handling. I am also proud of the fact that it uses API and Google sheets integration to help diversify what the VUI is capable of, and also personalise the VUI. However, it struggles with context tracking, it also cannot do orders in bulk, it is limited to one order at a time and not more than 2 different items. I couldn't figure a way to do bulk orders and doing as many as required orders was limited with the entities and variables in Voiceflow, and would be exhaustive to implement. The main obstacle to this would be bulk orders, if I could figure out how to do bulk orders this would make it a more complete food ordering VUI. If I had more time with this, I would use google sheets to create a much larger data base for menu items, I would figure out a way how to integrate this with intents, so that the VUI could do a better job at intent matching. If i had more time, I might have added some photos of each individual food item, so the user can see a picture what they have ordered. I would also add times and dates to past orders, so that the user could see when they last ordered. Another thing I may add if I had more time, would be more food ordering options, so it can be a more complete food ordering option, I would also include closing and open of the food stores.