Table Text
Senior Project

Daniel Pino
CSC 492
Spring 2016
Cal Poly San Luis Obispo

Table of Contents

Introduction

As technology begins to break into the food industry, restaurants continue to adopt new applications aimed to ease their employees jobs and their customer's experiences. Through my senior project, I seek to explore ways to improve the existing restaurant waiting list systems.

Problem

When visiting a restaurant, customers appreciate great service. This includes being seated early. Unfortunately great restaurants can also mean long wait times. Customers often choose a different restaurant when faced with the option to wait over an hour to be seated. It's a waste of time for both the customer and the restaurant if customer chooses to bail. Lots of restaurants use vibrating pagers to notify their customers when their table is ready during busy hours. The problem with pager based systems is that the customer is expected to wait outside the restaurant for hours until their table is ready. Customers hate waiting, but they enjoy the food so they decide to stay and wait to be seated. It's an inconvenience for the customer that can easily be solved with new technology. Lots of new solutions are specific to iOS or Android, which requires restaurants to purchase new hardware to begin using their application. This is unnecessary barrier to entry.

Background

There are several seating management systems that cover a range of features. Waiting lists are handled in a variety of fashions. Low tech restaurants still continue to use pen and paper to write down their waiting list. The issue with this approach is that keeping track of all of these papers can become a hassle. They require storage space and it is unlikely that restaurant will be able to extract any meaningful information out of them.

A second option that is quite popular for busy restaurants is a pager based system. The Pager based systems can cost over $5,000 so they require a large initial investment. The way a pager based system works is the host enters the party's information into a restaurant management application and they keep track of the pager identifier that was issued to that party. The host hands the party their pager that will vibrate and light up once their table is ready. The down side for the customer is that they are required to stay sitting outside the restaurant, in order to guarantee that the pager will be within range when they get called. From a host's perspective, they don't know if the customer received the notification when the pager rings and is out of range. Therefore, if the customer doesn't return in a short amount of time, they have to

continuously page them until the host gives up and moves onto the next party. This can be frustrating. A benefit of text based notifications is that the host can rest assured that the party received the notification regardless of their location (unless for some reason, the user turns off their mobile phone or doesn't have signal). In the case that the user did not receive the notification, the host will also know this.

Restaurants are now beginning to adopt text message based waiting lists. They offer similar functionalities to table text, but the customer does not have the option of texting the restaurant back. Table text will enable this functionality to allow a conversation between the two sides. Table Text will also know how to automatically reply to common questions without the host's help, which will ease the host's role and provide a better experience for the customer.

Solution

I'd like to build a web application that allows the host to keep track of a restaurant's waiting list and communicate with the customer's via text messaging. The host can enter a party's information and once the party's table is ready, the host can easily send them a text message via Table Text, notifying the party that their table is ready. For long wait times (1+ hours), this allows the customer to be able to do whatever they want during the time they'll be waiting. They don't have to wait outside the restaurant like they would with a pager based system. The goal of my restaurant wait list app is to ease the host's job and improve the customer's experience at the restaurant. I believe a web application is the right approach because web applications do not depend on any type of hardware or operating system. The web has advanced to allow rich applications that can be used by any platform, so regardless of what hardware the restaurants already own, they will still be able to use a web based solution.

Features

Two-way Messaging

Table text's main feature is its two-way messaging. When a party is added to the waitlist, the host will ask the customer for the name of the party, the party size, and the their mobile phone number. When the host adds this party to the waitlist, a chat conversation is created so that the host and the customer can communicate with each other. The host will use Table Text's built in chat functionality to notify the party that their table is ready. When the host sends a message through the chat, the message will appear on the Table Text app and it will also send a

text message to the party's phone number. This is where the two-way messaging begins. The customer can now communicate with the host by simply replying to the text message that they received when their party was added to the waitlist. The benefit of using a text message as a notification medium is that now the customer is allowed to do whatever they please during the time they need to wait.

Inbound Messages

Once the customer has the restaurant's Table Text number, they can now send the restaurant a message at any time. As an example, let's say one day you want to revisit the restaurant, but in your previous experience there was a long line when you arrived. Having the restaurant's table text phone number will allow you to send them a text message before arriving at the restaurant. Here is a potential conversation:

Customer: (Sitting at home) "How long is the current wait time for a party of 7?"
Restaurant Host: "Our current wait time for a party of 7 is about 45 minutes to 1 hour."
Customer: (Sitting at home) "Could I be added to the waitlist?"
Restaurant Host: "Sure! What's your party name?"
Customer: (Sitting at home) "Daniel party of 7."
Restaurant Host: "Thanks Daniel, you've been added to our waitlist. I'll send you a text once your table is ready. Please plan to arrive before the above wait time."

As you can see from the conversation above, the inbound messages feature allows customers to get on the waitlist from the convenience of their home. Many restaurants already allow call ahead customers to get on the waitlist by calling ahead of time. Using messaging instead allows the host to reply at their own pace instead of having to stop what they're doing to answer a call.

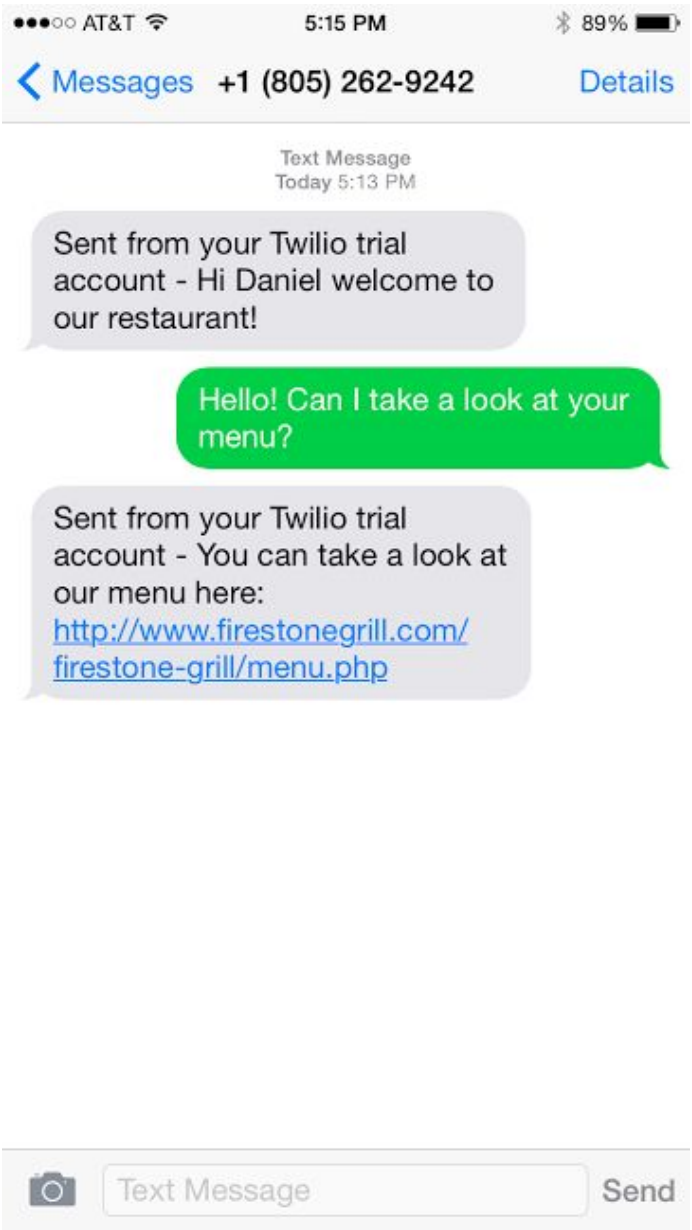
Automated Responses

Inbound messages also allows customer to ask any questions they'd like to ask the host. Table text will automatically reply to customer messages inquiring about common questions. These include questions such as:

What time are you open/close?
What are your business hours?
What's your address?
What's your website?
Can I see your menu?

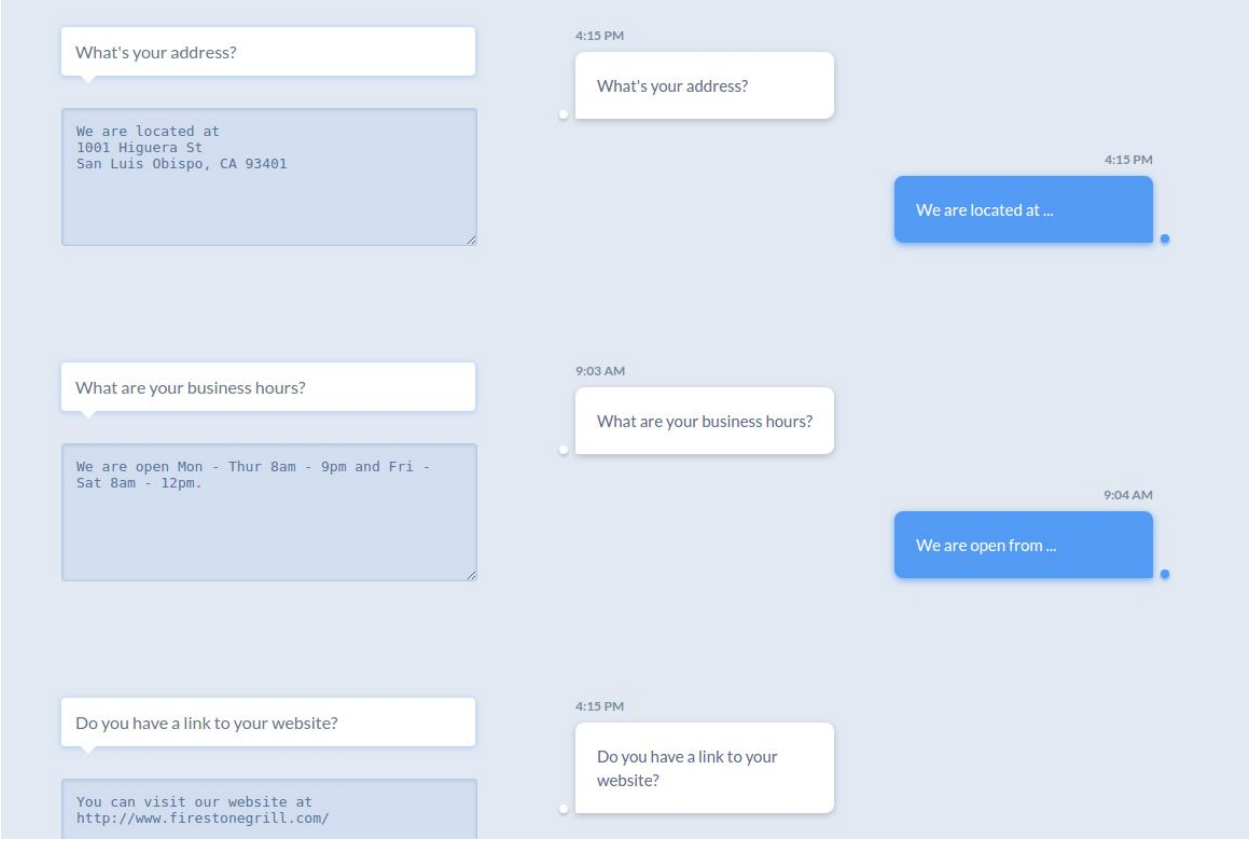
Do you offer pickup orders?

Theses are common questions that the host will often need to answer. Instead of requiring the host to tediously reply to the same questions all the time, Table Text can be set up to automatically reply to these common questions. The restaurant can customize their responses within the settings page to create a custom response to each type of question.



This is a view of the customer's perspective. They will simply be using their phone's text messaging app to communicate with the Table Text application. Regardless of what phone they're using as long as they have text messaging capabilities, their phone will work with table

text. In the image above, the iOS Messaging app is shown. The customer received an automated reply to their question



Here is the settings page where the restaurant can see which questions can be answered with automated replies. They can customize their answers at anytime and the new customized responses will take effect immediately.

Technology

I decided to build Table Text as a web application so any restaurant with an existing desktop, laptop or tablet could use it with their currently owned hardware. Therefore there will be no additional cost for the restaurant to adopt the application.

On the front-end I will be using web technologies such as HTML, CSS, and Javascript. I used jQuery for AJAX calls to the database and Socket.io to communicate from the backend to the frontend.

On the backend, I've used MongoDB as my database and Node.js as my server. I decided to use Node.js so that I could use javascript all throughout the frontend and backend. This allows me to use a single development language. This was important because it allowed me to learn and

use a single language in more depth instead of using multiple languages and touching the surface of each. MongoDB allows for schema changes on the fly, so this allows me to develop quicker as my project got started. MongoDB and Node.js work very well together and they are proven technologies that are used by large tech companies, so I was confident in its ability to perform at scale if this were to ever be used by many restaurants.

I used the Twilio SMS API for sending and receiving text messages. Twilio was a good reliable choice, but their prices are higher than other alternatives. Looking back I would have abstracted the text message sending and receiving logic to be able to easily switch to a different SMS API without affecting the rest of the backend logic. A better option would have been Plivio because their SMS sending and receiving prices are much cheaper. Like Twilio, Plivio is also used by many large tech companies and startups so I can trust that it is reliable and cheaper. Since the SMS messages are Table Text's biggest cost, I think Plivio would have been the better choice.

Topic and Domain

This project focuses on full stack web applications using technologies that are outside of the materials taught by Cal Poly courses. Since Cal Poly does not offer a full stack web development course, I strive to learn front end and back-end web technologies to build a full stack web app. On the front end, I will be using HTML, CSS, and Javascript. I will be using Facebook's React.js framework to develop the front end components. On the backend, I will be using MongoDB as my database and Node.js for my server.

Goals and Objectives:

The objective of my restaurant waitlist project is to create an restaurant wait list system that can replace existing options and improve upon their user experience from both the host's perspective and the customer's perspective.

Intended Users:

There are two sides to our application. The first side is the web client who's intended user is the restaurant host. The restaurant host is currently responsible to greeting new customers into the restaurant and using their existing systems to add the party to their waiting list. The host's role will be quite similar. They will simply be using a different application to create their waiting list.

Since the current state of Table Text does not support seating management, they might need to use two different applications. This is the biggest change in their current role that I foresee.

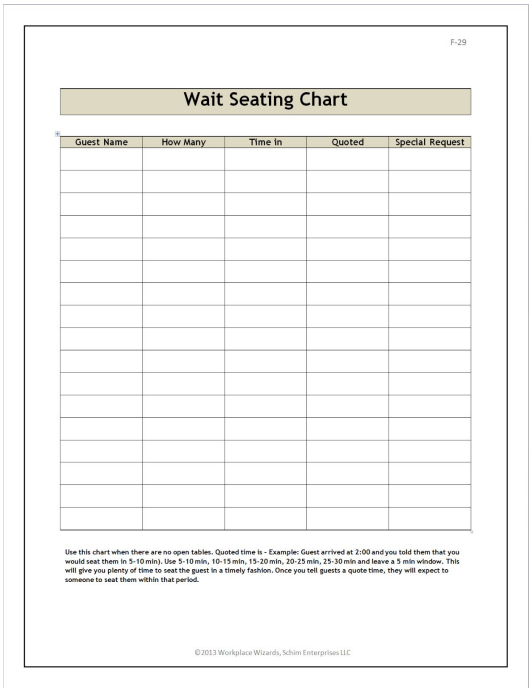
The second group of users is the restaurant customers. Although they are not direct users of the Table Text app, they will be receiving text messages from the app, some of which will come from the hosts and others will be automated.

Other potential users include restaurant managers. When a restaurant transitions to using Table Text, managers will likely learn to use the application so that they can train their hosts on how to use the app effectively.

Interviews:

Host Interview

I walked into a restaurant for breakfast and noticed about 15 people waiting outside so I showed the host my mockup design. I began by explaining to the host that I was a computer science student working on a restaurant waiting list web application. This particular restaurant still utilized paper and pen for their waiting list, so that's about as low tech as you can get. I asked the host what she thought about the look of the application. She said she liked the look of the interface. I continued by explaining to her how it worked. I explained that instead of writing the party's info on their paper, that they would simply type it in and add it in the application.



F-29

Wait Seating Chart

Guest Name	How Many	Time In	Quoted	Special Request

Use this chart when there are no open tables. Quoted time is - Example: Guest arrived at 2:00 and you told them that you would seat them in 5-10 min). Use 5-10 min, 10-15 min, 15-20 min, 20-25 min, 25-30 min and leave a 5 min window. This will give you plenty of time to seat the guest in a timely fashion. Once you tell guests a quote time, they will expect to someone to seat them within that period.

©2013 Workplace Wizards, Schim Enterprises LLC

Paper based waiting list template. My application includes the functionality of this template along with several other features and advantages of being a web application. The waiting list section of my application is essentially a digital version of this template. It provides the functionality of this template along with text message conversations with the customer. The advantages are that the digital version can easily be stored. Storing the data allows the restaurant to analyze the data and adjust their practices based on the data they've collected.
I think this restaurant was not a very good candidate for our application. They were accustomed to their old practices of using pen and paper, so using our application would require that they purchase a computer or tablet for their restaurant. Although the host liked the app, I think it would have added an unnecessary cost to their business. This restaurant was quite small, so I looked for a larger restaurant for my second interview.

Once thing I noticed was that this isn't necessarily a problem the host faces. Instead it's more of a pain point for the customer because they have to wait outside for up to 45 min (this restaurant's wait times don't get extremely high). Therefore, from this interview I learned to ask the host questions that focus on their interests and duties.

Customer Survey

I conducted a single question survey at a restaurant that had a long wait. I had been to this restaurant before and I knew that they used pagers when they have long wait times. I asked the people waiting outside whether they would prefer to receive a text message when their table is ready or use the pager that they were given. Out of 17 people I asked, here are the results:

88% would prefer a text message (15 people out of 17)

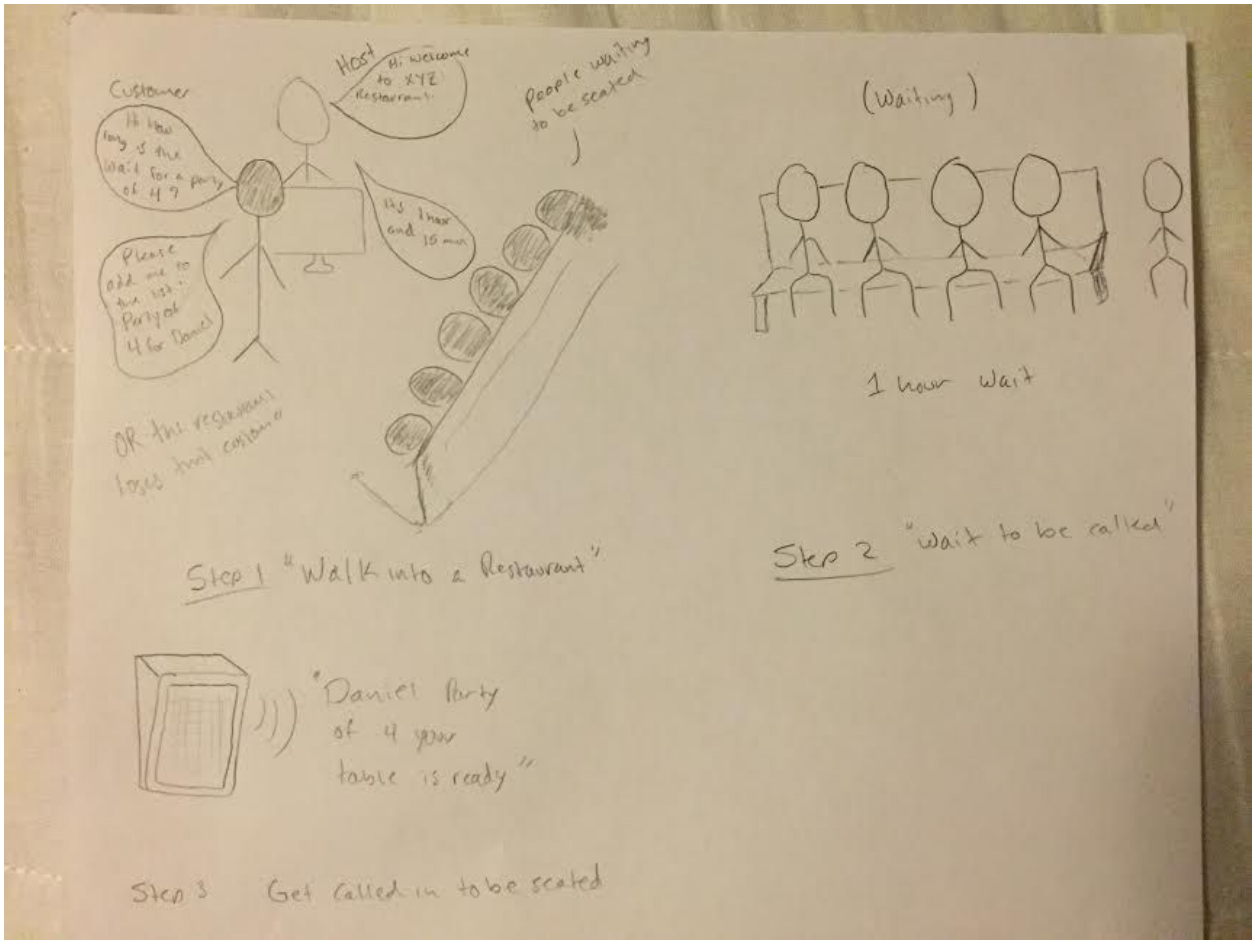
12% would prefer a pager (2 people out of 17)

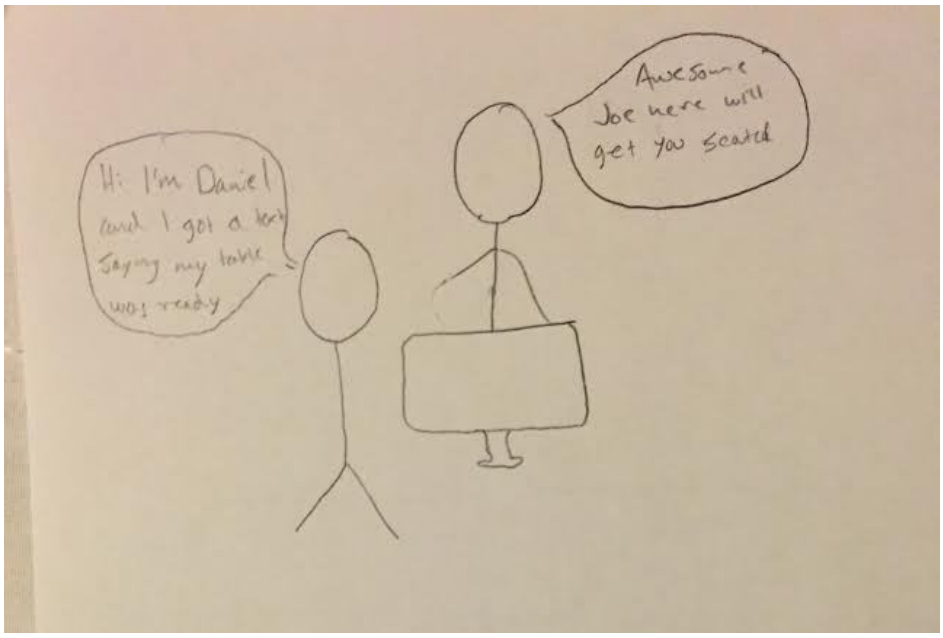
This restaurant is close to a shopping center, so the results might be biased, but I think this provides evidence that people value their time and they would prefer to have the option to spend their time doing things other than waiting. Although this was a very small sample size at a single restaurant location, the majority of the people here prefered a text message. One of the two who prefered a pager said they might miss the message on their phone, but with the pager they were sure to receive the notification. When I talked to them about the benefits of being able to do anything else in the hour that they waited, they seemed to changed their mind. The other person who prefered pagers was an older woman who suggested she wasn't too good with technology, I then told her that she could use her daughter's phone number who was sitting beside her, and she seemed to agree.

Scenarios

Scenario Without Table Text

Below is a scenario depicting how a customer and host would interact without table text.


Customer
Hi How long is the wait for a party of 4?
Please add me to the list. Party of 4 for Daniel
OR this restaurant loses that customer
Host
Hi welcome to XYZ Restaurant
It's 1 hour and 15 min
People waiting to be seated
(Waiting)
1 hour wait
Step 1 "Walk into a Restaurant"
Step 2 "wait to be called"
"Daniel Party of 4 your table is ready"
Step 3 Get called in to be seated

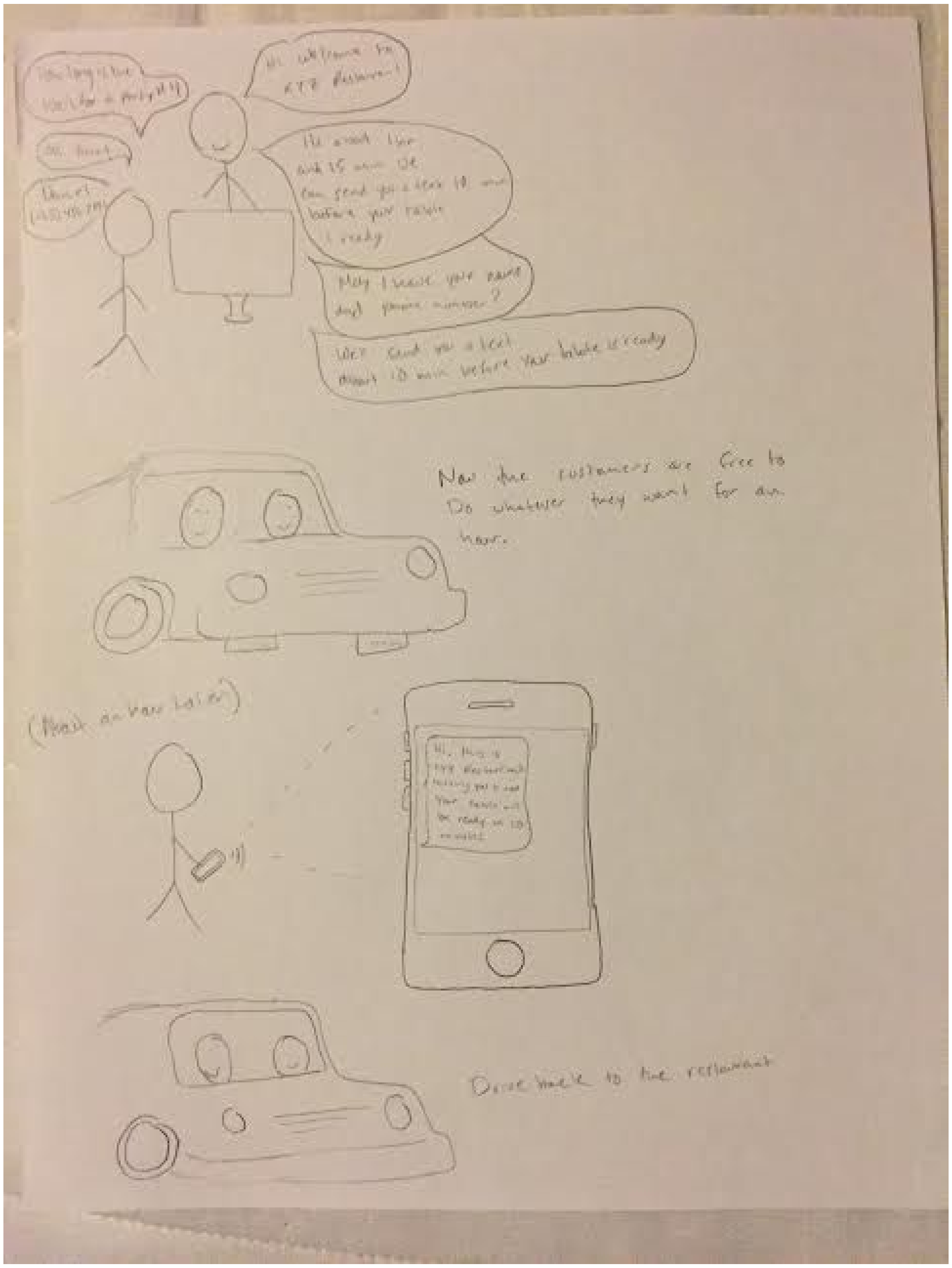

Hi I'm Daniel and I got a text saying my table was ready
Awesome Joe here will get you seated

As you can see, in the old scenario, a customer would drive up to the restaurant and greet the host. In a situation where there is a visibly long line waiting outside the restaurant, the customer might ask the host what the current wait time is for a group of 5 people. Then the host will take a look at their current table capacity and give an estimate to the customer. The customer with have to choose whether to wait outside or choose a different restaurant. If they choose to wait outside, they might be given a vibrating pager to get notified when their table is ready. Typically these pagers have a range limit, so customers will stay outside of the restaurant to make sure their pager is in range. Once their table is ready, their page will vibrate and they will come to the host to let them know their pager was alerting them. Then they will be seated.

In the new scenario using table text, there are two scenarios.

Table Text Scenario 1

In the first scenario, the customer drives to the restaurant and greets the host. Similar to the original scenario the customer might ask what the wait time is. Lets assume there is a long wait time of 1 hour and the customer agrees to be added to the list. The host will know ask the customer for their name, party size and phone number and add them to the table text waiting list. The advantage is that now the customer has the liberty to do whatever they please in that hour. Lots of times restaurants are close to shopping centers where they can spend some time while they wait without worrying about walking outside the range of their pagers. When the party's table is ready, the host will simply tab a button on the table text web app which will send the customer a text message notifying them that their table is ready. The host can avoid calling them or trying to find them in some cases. In the case of a no show the host and customer can send each other text messages through the table text app like any regular text message conversation.



May I have your name and phone number?
Now the customers are free to Do whatever they want for an hour.
Drive back to the restaurant

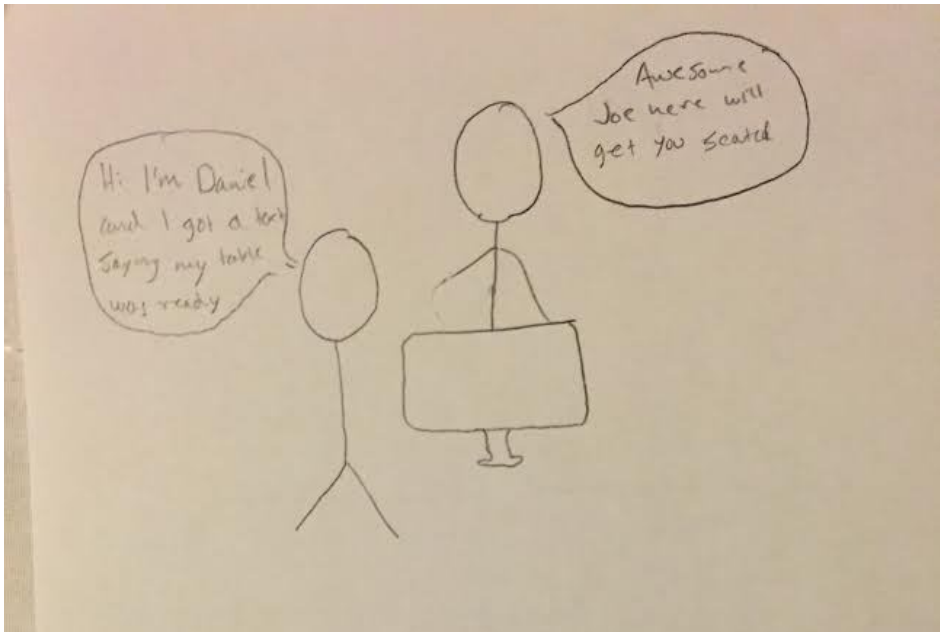
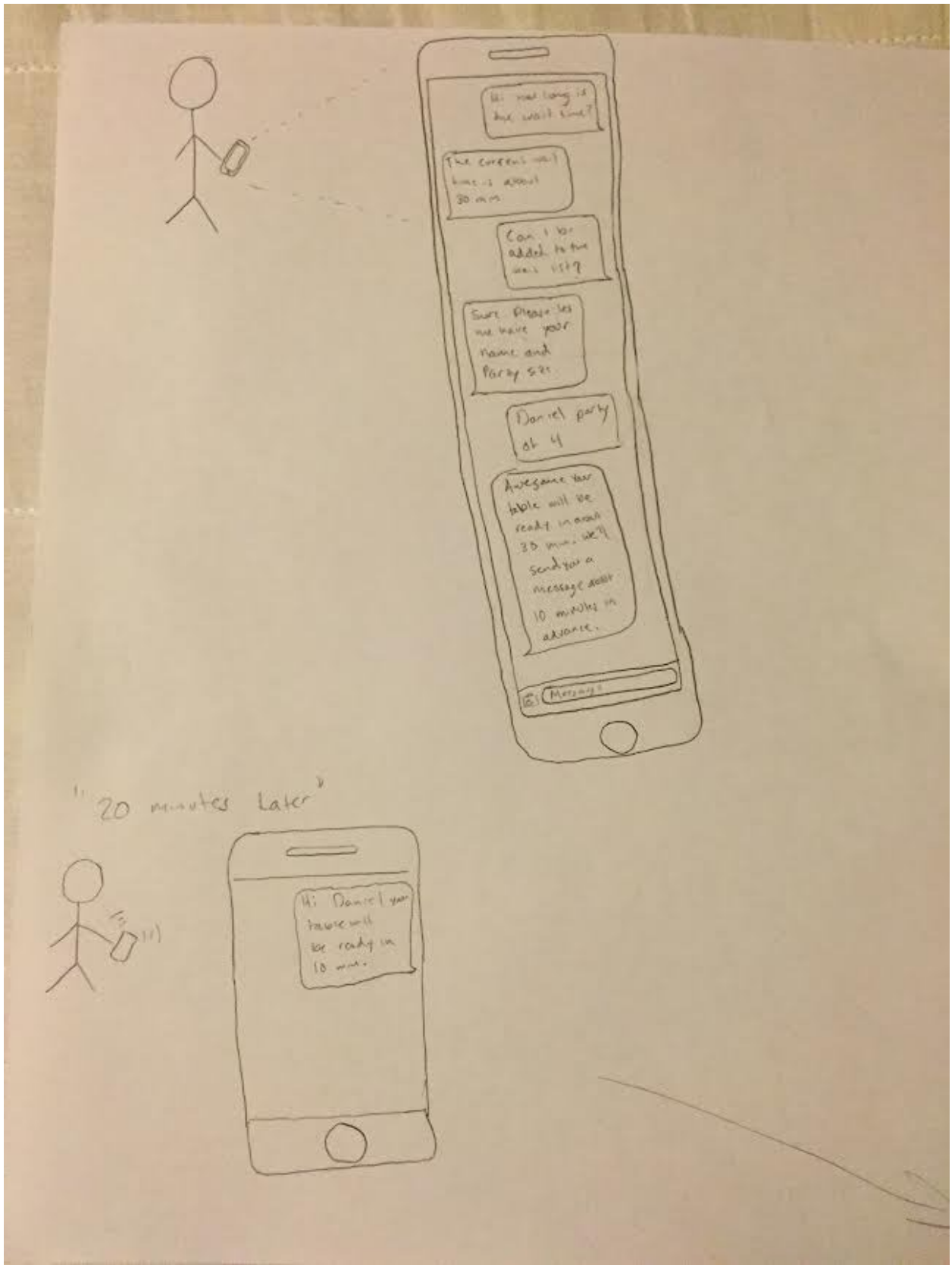
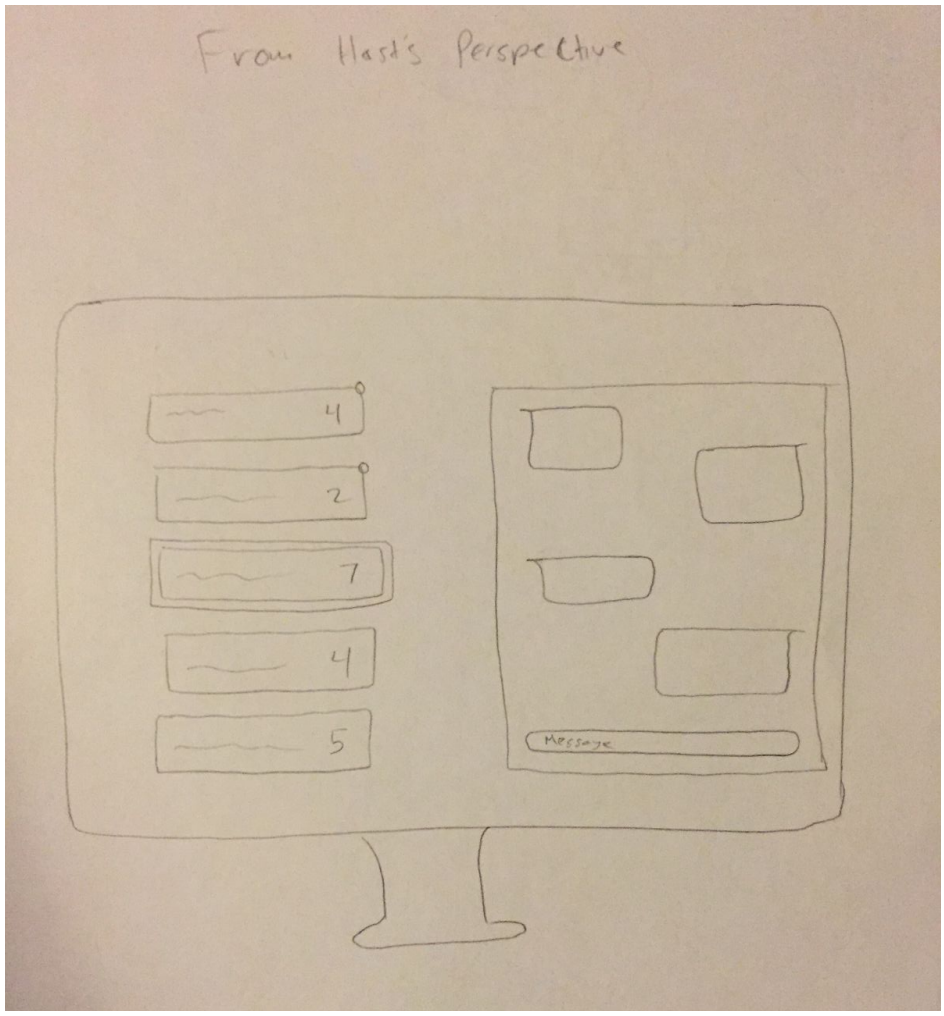
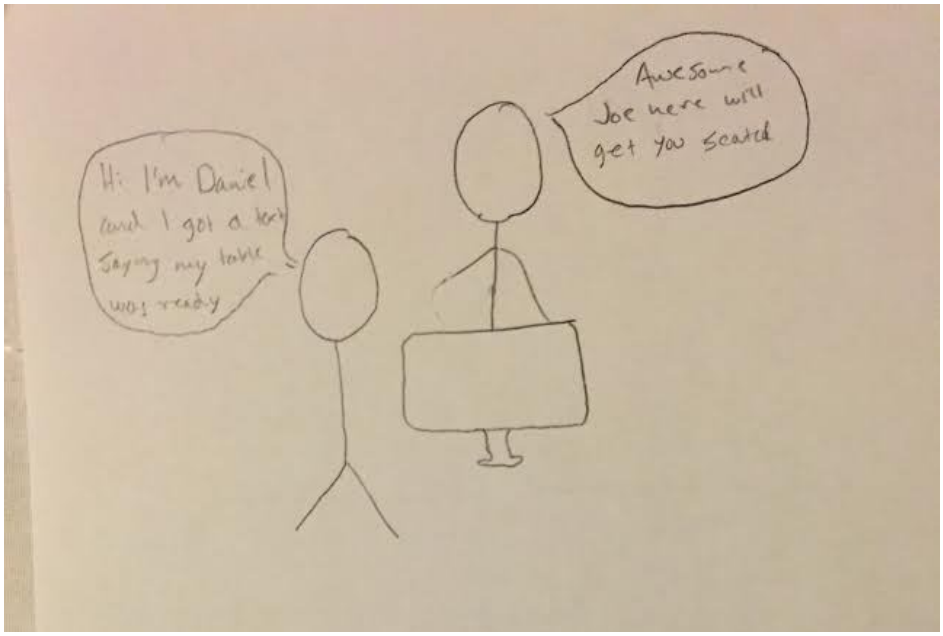


Table Text Scenario 2

The second scenario shows one of the biggest benefits that other waitlist management applications don't offer. The customer has already been to the restaurant once and used table text, so they now have the restaurant's phone number on their mobile phone. Before going to the restaurant, they can simply text the restaurant to ask them how long the wait is or any other questions before actually driving to the restaurant. They can also request to be added to the list via text message, and they will be notified about 10 minutes before their table is ready so that they can drive to the restaurant and tell the host their table is ready.

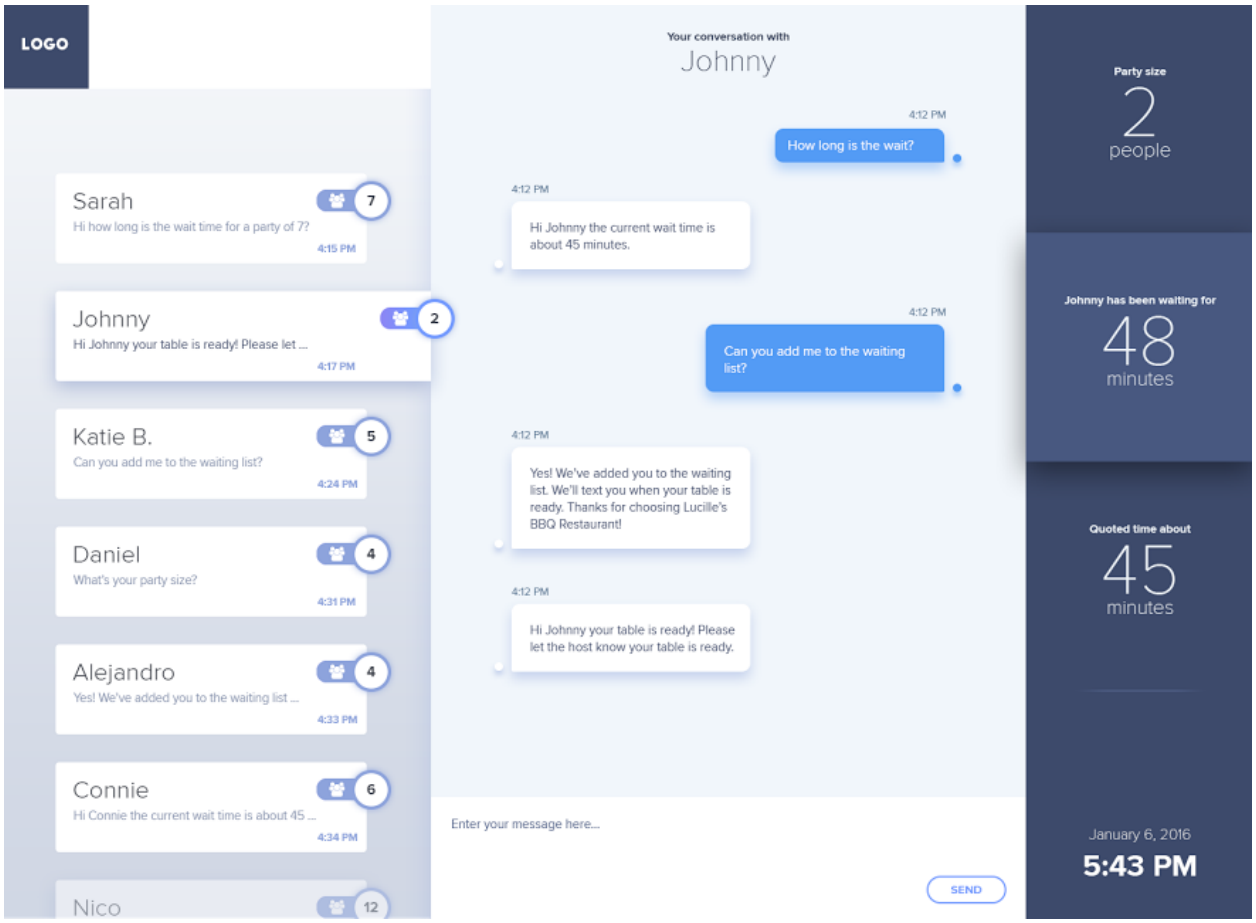

Hi how long is the wait time?
The current wait time is about 30 min
Can I be added to the wait list?
Sure. Please let me have your name and Party size
Daniel party of 4
Awesome your table will be ready in about 30 min. We'll send you a message about 10 minutes in advance.
Message
"20 minutes Later"
Hi Daniel your table will be ready in 10 min.


From Host's Perspective
4
2
7
4
5
Message


Hi I'm Daniel and I got a text saying my table was ready
Awesome Joe here will get you seated

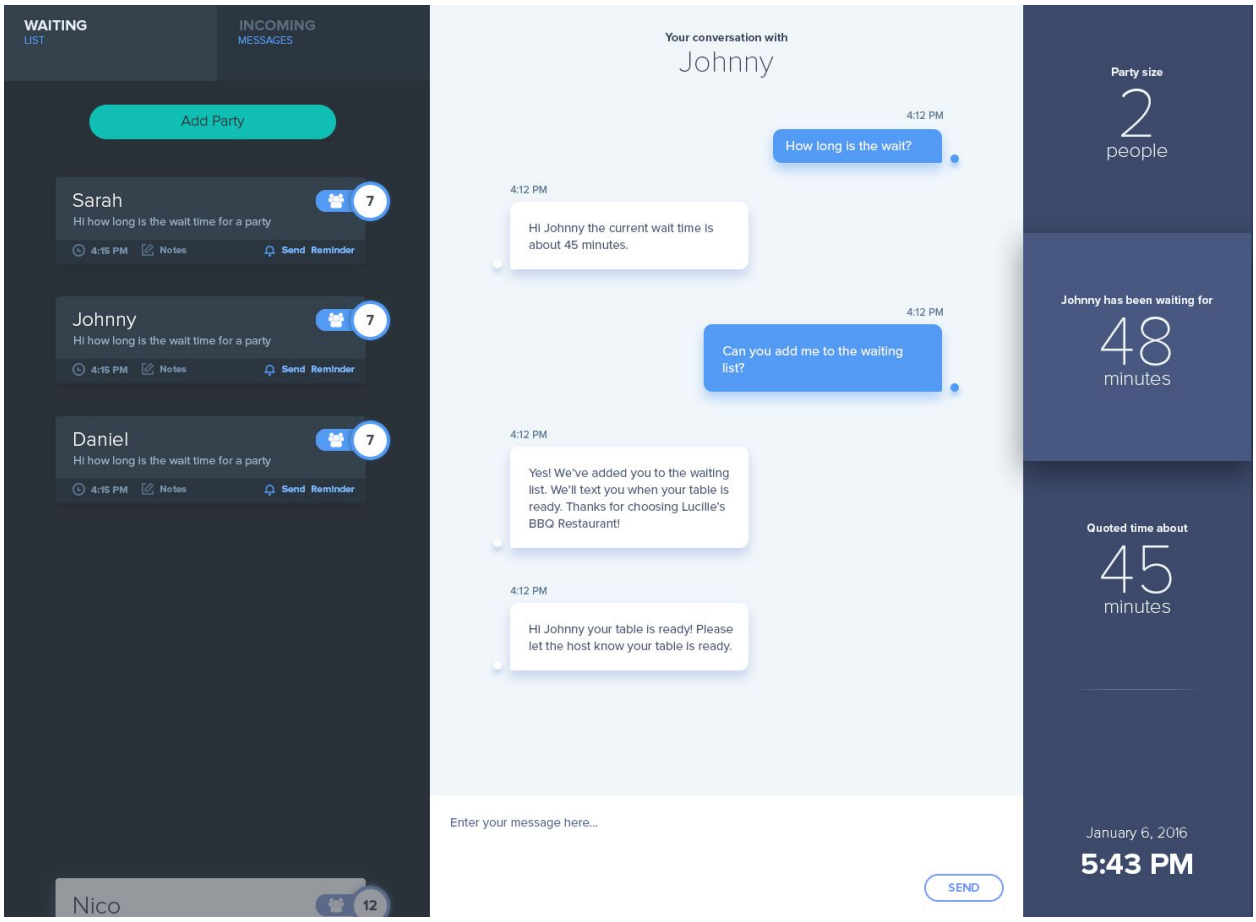
Initial Mockup Designs

Shown is a design for the web application's user interface. The intended user for this application is a host at a restaurant.



Refined Mockup Designs

After conducting the initial interviews with hosts, I collected valuable feedback that helped me adjust my initial design to better suit the hosts needs. I was also completely missing a crucial “Add Party” button in the initial design which allows the host to add a party to the waiting list. The new design shown incorporates these changes and will be developed as the final design.



In the new design there are tabs for different sections of the waiting list such as “Waiting List” and “Incoming Messages.” Clicking on the “Add Party” button brings up a form to input the customer’s name, party size, and phone number. The incoming messages section is for potential customers who are messaging the host, but are not yet on the waiting list. These customers might be asking about the current wait time, business hours, or any other general information.

System Architecture

MongoDB is a NoSQL database that offers document based storage. MongoDB uses collections which are similar to tables in a relational database, and documents which are similar to the rows in a traditional relational database. The collections in my database are described below:

Party

- Id - *ObjectId*
- name - *String*
- Size - *int*
- notes - *String*
- arrival_time - *Date*
- mobile_number - *String*
- conversation_id - *ObjectId*

Conversation

- Id - *ObjectId*
- Created_at - *Date*
- updated_at - *Date*
- is_incoming - *Boolean*
- mobile_number - *String*

Message

- id - *ObjectId*
- message - *String*
- is_incoming - Boolean
- state - *int*
- created_at - *Date*
- conversation_id - *Date*

Setting

- user_id - *ObjectId*
- hours - *String*
- address - *String*
- menu - *String*
- website - *String*
- takeout - *String*

When a customer wants to be added to the waitlist a party and a conversation are created. Every party has a conversation associated with it. Then when a message is sent via the app, a message document and a twilio sms message are created. When a customer sends a text message to the restaurant's phone number, twilio's api will send a request to my server for an incoming message. So I then save the message after setting "is_incoming" to true and link it to it's respective conversation. There is a second type of conversation that is created without a party. This is an incoming conversation which can be initiated before a customer is at the restaurant. They can ask the restaurant a question and it will create a two-way conversation between the host and the customer. When a request for an incoming conversation reaches my server and there currently does not exist a party on the waitlist with the same phone number, then a new conversation document is created and "is_incoming" is set to true. Now their conversation will appear in the incoming conversation section of the app instead of the waitlist.

Competitors

There are several established competitors in this space. A few companies who have dominated the vibrating pager based waiting system, have developed an option for sms based waiting system.

The first product is HME SMS SmartCall. HME's solution offers the waiting list functionality, but it does not allow the host and customer to interact beyond the initial notification text message. The functionality here is limited and the interface offers a lot of room for improvement. SMS

SmartCall is a cloud based solution so it can be accessed on any device just like Table Text. The main feature it is lacking is two-way messaging. This means that the feature set is essentially the same as having a pager. The host can blindly send messages, but the customer will not be able to reply back to the restaurant if they are not able to make it or they are late. This is an inconvenience for both the customer and the restaurant.

The next product is JTECH's Restaurant Wait List app. JTECH's solution has waitlist and table management features, but it is an iOS only solution. This means that restaurant's will need to purchase new iPads in order to be able to use their product creating an unnecessary barrier to entry that Table Text doesn't have. JTECH's advantage is that they already have market share in the pager based systems, so their brand is well known in the space so restaurants can transition from their old pager based systems to the iPad solution with the same company they already trust. It does not offer two-way messaging (only messages from the host to the customer) and Table Text's user interface is much easier to use. They also do not allow incoming conversations which is one of Table Text's major advantages.

Future Improvements

Table Management

After interviewing several hosts and restaurant managers, table management functionality came up in several conversations. Although some restaurants liked table text's functionality, they also required table management to fully manage their restaurant. This seemed to be a deal breaking feature because restaurants didn't want to have two separate applications for a wait list and table management. This means that they would need to train their employees for two separate applications and pay for 2 separate applications. Integrating this feature would really provide lots of value. The waiting list feature will be in sync with the table management functionality. For example, when a table of 5 opens up, the next party on the waiting list that has a party size of 5 or fewer will show an option to add them to this table, and automatically send them a text message notifying them that their table is ready. Tapping this button will also reflect on the table management side, the name and size of the party that was added to the new table and set the new table as being occupied. The idea is to sync the table management and waiting list features as much as possible to create a seamless user experience for the host.

Text Message Marketing

Once a restaurant begins to use Table Text, it would be beneficial for the restaurant to be able to create marketing campaigns. This is also beneficial to customers because they would receive coupons to their favorite restaurants. An example of a text message marketing campaign is for every party that has been to the restaurant 4 times in 1 month will receive a 50% off their next meal. These would be regulars who would really appreciate the 50% off and might even invite others now that they can afford an extra meal. If their party size is always 4 and you send them a coupon for buy 4 meals and get the 5th free, it may encourage them to invite a few friends to join them for dinner, thus increasing the restaurant's customer base. Marketing campaigns would be targeted toward customers with multiple visits because we would like to target the customers who will get the most value out of them. Targeting all customers may feel like spam to non-frequent customers, and thus create complaints for the restaurant. Some customers

really dislike the idea of receiving advertisements via text messages, so they could easily opt out of the text message marketing.

Extracting Information From the Data

Once Table Text has been used by a restaurant for a long period of time, it will have lots of data that could be useful to a restaurant. The data will already be stored in our database, so all that needs to be done is present this data in a manner that is useful to the business. As an enterprise application, restaurants can use this data to make their restaurant more efficient. A few examples of useful information would be:
How often customers return to their restaurant?
How many returning and non-returning customers they have?
How often they have call ahead parties?
Do the call ahead parties come early/late on average?

All of these questions can be answered by collecting the data that Table Text already collects and presenting it in a readable and practical manner.
In collaboration with SMS marketing, the data can help businesses make more accurate data driven decisions about marketing campaigns. A few questions that could be answered include:
Did the users that received coupons use their coupons?
Did these customers bring larger parties after receiving coupons?
Did a marketing campaign present an increase in first time customers?

Full Source Code available here:

https://github.com/dannyp32/tabletext