Demo 1 Presentation Line Up:

Tim Gilligan: Introduction of SpeedByte

When we made the initial decision to create an all-inclusive restaurant automation app the first thing we did was set up a meeting with a local restaurant owner, Rob Mozzarella owner of Panatieri's Italian Cuisine. Rob described all of the endeavors that his business faces every day, from inventory shortages, to under or over staffing at certain hours of the day. In addition to Rob we also chatted with some of his employees who talked endlessly about the lack of communication creating big problems, especially in the hectic hours of prime time. We then sought input from a broader scale, interviewing countless customers who all seemed to complain about waiting for tables, tipping expenses, and the uncertainty of the quality of what they were ordering.

We then tasked ourselves with the mission of somehow solving all of these problems simply by introducing a mobile app to the restaurant. After endless hours of brainstorming we came up with the solution that will benefit everyone involved while maintaining a simplicity that will allow anyone to experience a great dining experience. That solution was SpeedByte.

Jimmy Jorge: The SpeedByte Mission

After extensive hours of research, we've come to the conclusion that restaurants care about one thing, and that's profitability. Here at SpeedByte, we allow the business owner to efficiently run his or her business without having to worry about all of the headaches that come with being a business owner.
To start off, our mission is to increase restaurant efficiency and profitability while helping to maintain a productive restaurant atmosphere and experience. With SpeedByte's all in one, automated, easy to use system, we will guarantee any restaurants that implement our app into their systems will help create a reputation that customers want to be a part of.
As we all know, restaurants need business to be successful. So working off of that idea, one of our main focuses is creating something easy and straightforward that the customer will love to use, and will make their experience better. From a customer's perspective, when going out to eat there are always a ton of variables. Waiting to get seated, waiting for food, waiting to pay... we want to eliminate all of the waiting. From an employee's perspective, there's problems with inventory, problems with workers clocking in/out, and there's always issues on the floor throughout the working day. We want to simplify all of that through our multiple features and infrastructure which you will all see in a moment.

I will know let Tim, the lead designer of SpeedByte's User Interface, talk about how our app will help everybody in the restaurant force, from Customers and employees, all the way up to the owner.

Tim Gilligan: User Interface

When designing the interface for SpeedByte I kept one thing in mind at all times, ease of use. This means finding a balance between usability and functionality, while limiting the number of clicks by the user. Each interface is unique for that type of user and has features tailored to address their needs.

- **Customer Interface Notable Features**
  - Table and Time selection
  - Dish Rating system
  - Dish wait time
  - Interactive menu
  - Out of Stock Display
  - Help Page

- **Manager Interface Notable Features**
  - Message Center
  - Statistics Tracking
  - Add/Remove Item
  - Add/Remove Employee

- **Employee Interface Notable Features**
  - Message Center
  - Real time updates
  - Easy to use interface
  - Helps organize work load

Vamshrikrishnan B: Implementation

This mobile application consists of two major parts: the front-end and the back-end. The front-end (also known as the user interface) is developed used a development environment called Android Studio. The programming languages used in android studio are Java and XML. The back-end consists of two parts: the server and a non-relational database. The server is programmed in Node/Express JS and the database is mongo DB. Node JS contains its own library for mongo DB, so the database was also programmed in JavaScript.

The following is the algorithm for the customer side of the application:

Customer: The customer has the option of choosing if s/he wants to dine in, order takeout, or order delivery. Next, the customer will choose the items from the menu. To maximize efficiency, items will be added into an expandable data structure as they are selected by the customer. If the customer unchecks an item, then that item will be deleted from that data structure. The expected big O notation for this algorithm is O(n) time and O(n) space. If the customer chooses to dine in, he/she will select a table and choose the time that s/he wants to eat. A simple algorithm is needed in order to accomplish this. When it gets closer to the scheduled time, the customer will get alerted. The database will contain information regarding the tables (number of people for each table), menu, and a temporary storage of customer's information. The menu items order will be stored permanently, and a graph representation will be made for the manager's use. The customer will also be given an option to take a survey of the restaurant. When the customer is done with the survey, the data will be stored into the database in the order that it is

presented. The big O notation for this algorithm is O(1). No data structure will be required in order to implement this. This algorithm is yet to be implemented.

Pawel and Ram: Quality Assurance and Security

We predict that plenty of people from Managers to Customers will be using our application. Hence, we would like to present them an application that ensures reliability as well as smooth and power-efficient performance. Progress and stability of our application is tested at each major development phase using Manual Testing. In order to minimize the amount of defects or "bugs" as developers call it from our application, we would like to stress test application at all levels. Manual Testing itself can be broken into several categories, including, but not limited to, Graphical User Interface (GUI) Testing, Functionality Testing, Code Review, and Securities Testing. Hopefully, in the future, we can make the testing automated as opposed to manual that is there currently. Our ultimate goal is to present the users with the most functional, simple, yet secure application that will meet the high level of expectation.

Concerning testing, we will go through the clickable objects manually, attempting to access each object individually to see where it leads and compare the given destination to the expected destination. Method tests will follow a similar suite, with their input being varied in order to cover the entire spectrum without calling anything else, with the results being recorded and compared against what was expected. Following those tests, the methods that usually call other methods will be encouraged to do so, with their input also varied in order to cover all possible types of input. Moving onto security, our app will track the customer information, in the case that the customer decides to not pay for the food ordered. Such information will include the phone's hardware identification number, phone number, and possibly credit card number (if it was supplied at any point in time). This information will then be hashed using SHA-2 and then stored into the company database, in order to have something to pass onto the authorities. Any other information, customer or otherwise, that is stored on the database will also be hashed in order to deter any corporate espionage. Such information would include employee details, passwords, etc. Regarding possible order scams, such as a third-party (infiltrator) attempting to spoof as a device, there will be a system call made to the original device logged to see if the order was actually made from that device. Thus, a spoofed order will need an actual hardware I.D. in order to be able to order anything. Another means of deterring any infiltrators will consist of limiting the number of characters in any text in the app itself, as to not allow the hackers an open entry point to perform a type of SQL injection. Finally, as the technology behind the language that the app was built on advances, we will update our own code in order to limit security risks. This is due to the fact that every language has its own flaws, but each update to the language manages to fix some, even though it could produce others.

Husen: Upcoming Features

Our future implementation will include the following three important features.

- **Message/Notification Center**
- **Employee Clock in/Clock out**
- **All Encrypting Login for all Users**

Message Center

Although there are several messaging apps on the market, none of them offer a system that can be integrated into another app. The SpeedByte built-in message center will help all employees with communication and productivity as well as a host of other benefits. These benefits include but are not limited to, profitability, efficiency, convenience, and reputation building.

Increase in Restaurant's Profitability:

The most important benefit of having a message center is that it will eliminate the needed for internal employee communications via devices such as headsets or Bluetooth. This will remove a high cost of operation and in return create a more profitable environment. Restaurants may also have to outsource and pay extra for additional services; however, in SpeedByte, this service is provided at no additional cost. The message center can also be preprogrammed by the manager to have commonly used phrases. This will eliminate any improper use by employees, as well as create an easy to use, one click to convince system.

Big help to Employees:

Another benefit is that if there are errors in taking orders for a customer, the chef could notify employees and/or the manager easily through one quick click of the message center. The message center could also act as a notification center for issues such as *order out of stock* or *inclement weather*, all the users will be notified through a generic preset message " Order Out of Stock." and "Restaurant will be closed today due to inclement weather."

Easy to use:

Only one click is to convince, this will benefit employees with time, as well as allow the less tech savvy employees to convey a message. Plus there will be no possibility of personal usage which could create future problems.

Reputation:

Now that all the communication will be done using message center, restaurant atmosphere will improve as there will be no yelling across the restaurant interactions between employees.

Encrypted Login

All the employees will have a unique user ID and a Password to perform all the actions. For security purposes, the encrypted login will allow specific user types to do their specific tasks. This will make the risk of being hacked significantly less.

Clock in/ Clock out

The Clock in/Clock out feature is a basic requirement for all the businesses. SpeedByte provides this as a built-in feature, this eliminates the cost of external devices.

Benefits of clock in/clock out:

There will be another interface at a restaurant provided from SpeedByte which will allow all the employees to clock in/clock out. Restaurants may have to outsource and pay extra for additional services; however, in SpeedByte, this service is provided free of cost. Not only is it free of cost, but it also eliminates all the hassle such as filling out a Timesheet, making an excel sheet for the payroll company, and keeping track of the employee's work log.

Easy to use:

One click to Clock in and one click to Clock out for all the employees. Manager will be able to pull out employees work log history per week to send it to the appropriate payroll company.

Database

Right now, mongo DB is being used to store data. However, in the future, if the mobile application becomes more popular and the market for the application increases, a central operating database called RDBMS will be implemented. RDBMS will keep track of all the local databases for each individual location. The individual database specifications will vary based on the restaurant's' requirements. Creating a central database will help in increasing profitability, as well as lower maintenance cost.