SRS Requirements Outline

Project Description:

DineOn is an Android application that makes the dining experience more efficient for both the server and customer, as well as makes the experience more personalized and tailored toward the user based on data collected through the application. To accomplish this, a system is in place that manages a database, which includes user data and restaurant data. With this data the application builds a personalized experience for each restaurant that the user searches and gives them the ability to interact with that restaurant without being present at the location of the restaurant. Some of these interactions include building orders from the restaurant's menu and sending a request for pickup, delivery or dining in for that order. Also, the system will eventually support a payment system that uses the user's payment information and sends the restaurant the transaction information. The user can also reserve a table at the restaurant and upon arrival can scan the QR for that table, which checks in the user with the restaurant and server and alerts them that you're ready to be served. Instead of waiting for the server to purchase additional items the user will be able to order through your application, which alerts the server of additional purchases. All of the user interaction with the restaurant will be remotely done by the user through the application.

This project will be a great addition of resources for all types of people who want to make the restaurant experience more efficient and enjoyable. Also, the application is will be an important resource for restaurant's themselves because it will make their operations more efficient and eliminate the time of customers waiting for their food. This application will be available to anyone with an android device and the user interface is designed to allow users of all technical experience to user the application easily and efficiently. The idea of automating the restaurant experience is interesting because for years the routine hasn't changed much, but with the integration of technology in society is time for that process to be improved and the Restaurant Experience is the application to do just that.

Software Toolset:

In addition to the required software tools, we are going to use a cloud service that provides data storage called Parse. Using an API provided by Parse, our application will be able to store and query any data that gets sent there by a user. There is no query language that we need to use directly. All queries are made by using Parse's Android API. This is a plus for our team because Java is the most familiar language to us, and we do not need to learn any database query languages. For additional functionality, like utilizing push notifications, we need to use Parse's Cloud Code functionality. This would require us to write a little bit of Javascript using their Javascript API. The whole idea of using parse is to abstract the network code out of our application. Although we will be writing a small amount of Javascript, we feel the benefits of using Parse to simplify the network side of our application outweighs the minor inconvenience of using a less familiar language. Plus, some of the members on our team have experience with Javascript and can help those who have never seen it before.

Feature Set

Requirements

- Users will be able to login to the application allowing them a tailored, personalized experience
- Users will be able to associate themselves with a restaurant. This allows the content to be focused on a particular restaurant. Users will also have a seamless way of changing restaurant of interest
- Once users picks a restaurant of interest they can view menu items, restaurant information, friends opinions (social network), make reservations, and place orders
- Users can physically check themselves into the restaurant to either notify the Restaurant that they have arrived at the restaurant to Dine in or here to pick up an order
- Users can check out any time while they are dining in.
- Server is able to resolve login request
- Server is able to resolve queries pertaining restaurants
    - provide necessary restaurant data
- Server should manage dining session

    - user/customer arrival
    - user order
    - User<-->restaurant interaction
- Database should track how up to date the data is(Time and date)
    - This allows for caching within the application itself
- Database Should store images
    - Restaurant Logo
    - Restaurant enviromental pictures (In restaurant pics)
- Server and database handles concurrency issues with restaurants and android phones

Core Features

- Check-in feature for restaurants
    - For this system to work for clients deciding to dine-in (see Minor Features 1a) a check-in feature will be implemented that allows clients to register with the particular restaurant that they are currently at. This will be done through either QR codes, proximity based, or by making reservations where a register code will be sent from the restaurant or manually in person. Checked-in users will be able to then order from their restaurant and seat themselves without having to wait.
    - User can make reservation that is maintained by the server.
        - Server checks restaurant and time
- Enhanced Restaurant interactions
    - Menu browsing and reviews
        - Maintained by database on restuarant side
    - Call of waiter/waitress
- Order placing at restaurants
    - Clients dining-in will be able to place their order which will be sent through the server to the restaurant machine. The chefs will then have easy access to these orders.
        - Standard way: accumulating orders, pay at the end
    - Clients selecting take-out or deliver do not have to check-in, but must still pay when they get their food (either at home or when they pick it up). Similar to 3a, orders are placed in a queue maintained by the restaurant.
- Database Personal Data stored. Persistent, restaurant-oriented user data
    - Store information regarding Login credentials
        - Non 3rd party data Restaurant Generated Username
    - Store previously visited restaurant dining session information
        - Potentially connect to facebook
    - Store past meals, entrees, beverages,
    - Rating abouts restaurant
    - Current Reservation
    - Favorites... (Restaurant, Menu item, Hot bartender)

Minor Features

- Choices of dine-in, take-out, delivery
    - Dine-in: selecting this option will allow the client to check-in to the system and order directly from the restaurant to be delivered to a designated table as determined by the user.
    - Take-out: selecting this option will allow the client the convenience of ordering from home or on the go and picking their order up from the restaurant personally. Users of this option do not need to check-in, but will be provided an order code or will provide their name for the order (TODO). Only some restaurants will support this option.
    - Deliver: selecting this option will allow the client to order ahead of time and have their meal delivered to them. Only some restaurants will support this option.
- QR Code Recognition

    - QR Codes can be placed at seating locations
    - Front desk
    - Outside restaurant
- Get directions, hours, basic information about restaurants
    - Clients will be able to browse participating restaurants and learn basic information as provided by the restaurant such as directions, hours, menu sample or the entire menu.
    - This will be handled by having participating restaurants provide most if not all of this information when they sign up for the service.

Extended Features

- Augmented Reality (Vuforia)
    - Recognizes registered restaurant signs and opens the custom page of the restaurant in the application
- Proximity based check in
    - NFC
        - Probably not bluetooth are Alljoyn
- Additional login functionality
    - Facebook authentication access tokens or user id
    - Google authentication access tokens or user id

Phase Breakdown

There is no way we can forecast this accurately. it is more of a plan that we can reference later when we need to adjust.

Phase Alpha:

- This segment of time will be focused on building a lite end to end system with minimal features.
- Define interfaces between android apps and the server/database (Parse)
- Android specific: Have basic UI elements in place. Have end to end activity negotiation. Have asynchronous network protocol implemented with server
- Parse
    - Basic implementation answering queries from the database
    - Implement the database structure, capable of storing data for all planned features
- Clearly identify abstractions
- A user (customer app) will be able to browse restaurants, view their menus, and if they say they are at the restaurant, gain access to calling their waiter or requesting the bill.
- A restaurant user will be able to select their restaurant and edit their stored info, including menu items and hours.

Phase Beta

- This segment of time will focused on increasing the complexity and number of features
- Android (customer app)
    - application specific Database for caching.
    - Complete the User experience, QR Code
- Android (restaurant app)
    - Implement management procedures for the restaurant to interact with the client side
    - View and manage current dining sessions
    - View and manage check ins
- Parse
    - Be able to React and track a start of a dining session.
    - React to and track “Check in”s.
    - Manage basic requests (such as orders) from users.
- Define interfaces between db and server, server and client.
- A user will be able to sign in, then select a restaurant, as well as being able to view reviews. After confirming check in, the user can place an order and pay through the app, in addition to previous features. Users who are not checked

in can place a reservation for later, in addition to previous features. QR code recognition is available for check ins.
- A restaurant user will be able to sign in, then view all current dining sessions, and perform basic actions on them, such as deleting the order, or marking it complete. New accounts can create a new restaurant. In addition, they will be able to generate specific QR codes for users to check in with.

Phase V1:
- Ensure end to end behavior.
- Android customer app: 3rd party login. Facebook integration. Google Maps integration. Google Search.
- Android restaurant app
    - Full customer tracking and editing
- Full implementation of core and minor features, including the dine in, takeout, and delivery tracks for customer app
- Detailed restaurant information
- Facebook integration for personalized recommendations
- Any extended features

Adjustments

- Cut: Remove extra 3rd party features.
    - Remove Facebook integration
    - Remove 3rd party login
    - QR Code
- Cut: Remove Restaurant Employee Application Experience
    - We will postpone the ability for users to communicate with waiters and waitresses on their respective mobile devices. Instead we will just maintain a queue on the server to acknowledge that the request are tracked.
- Cut: Take-out/Delivery/Dine-in
    - Shift to an exclusively in-restaurant application. Remote/advance ordering will not be available. Such features compliment our app, but are already available through other channels (i.e. Eat24, Websites, etc) and do not necessitate a mobile device.
- Cut: Making Reservations
    - Similar to above, making reservations is not critical to in-restaurant usage of the app.
- Adjustment: Menu Complexity
    - Reduce complexity of menu system. No framework for placing special orders (outside of a ‘extra notes’ input), no custom pizza toppings, etc. Just binary selection
- Adjustment: Payment system
    - Change payment system to “ask waiter for bill”
- Adjustment: Login functionality
    - Login with Facebook/Google authenication
- Cut/Adjustment: Back-end authentication
    - (Worst case scenario) Reduce strictness of back-end authentication - Server doesn’t have to validate user “Check-ins”. i.e. No restriction on ordering menu items from different or non-checked-into restaurants (on the back end - client still shouldn’t display more than 1 restaurant at once).
- Augmented Reality (Vuforia)
    - Recognizes registered restaurant signs and opens the custom page of the restaurant in the application

Team Dynamics:

- We have a simple communication hierarchy. There is a PM, Two technical leads, and a librarian. As a whole we are the development team for our mobile service. In terms development, we are divided into two sub teams a user end android application and a restaurant (client) end android application. There are individuals per team that will have direct control and dictation of the user interface, server communication, and back end process.
- Each team has their own technical lead. Tech leads and PM will meet on a need basis over a variety of communication platforms (Email, Conversation Facebook/Google Hangout, Physical meeting, etc). Tech Leads and

PM have a mandatory, weekly customer meeting. Group scrums are schedule once a week. At these meetings, individuals will have the ability summarize their weekly progress, raise concerns, and place input regarding project progression. These roles were picked based off developer experience and interests. If disagreements arise, they will attempted to be resolved at the level of the communication hierarchy that the conflict occurred. If members of a sub team disagree it will be handled within that sub team while the respective manager mitigates. Cross team communication, including disagreements, will be mitigated through technical leads and the PM.

- Assigned Roles
    - (Michael Hotan) The PM is the coordinator between the development team and the customers. PM also acts as the water tester ensure the scope and schedule of the project is still feasible. PM directly coordinated between the respective Technical Leads and the librarian.
    - (Mark Rathjen, Jordan McNeal) Tech leads are responsible for disseminating team specific information. They are also responsible for guiding their respective teams ensuring tasks and milestones stay within the overall scope.
    - (Vince Blas) Librarian: Overall responsible for the integrity of the documentation. Everyone is responsible for contributing valuable documentation for their respective work. The librarian has authority to contact any member of the team with any issues regarding their documentation contributions.
    - User interface: User end - Garrett Lee, Restaurant end - Vince Blas. Michael Hotan will mitigate coordinations between the two teams.
    - Server administrators: User end - Mark Rathjen, Restaurant end - Jordan McNeal. Coordinations will be done directly between the two teams
    - Back End: User end - Nicholas Johnson, Restaurant end - Zach Rathbun. Michael Hotan will mitigate coordinations between the two teams.