SRS Use Cases

Formal Use Case: Placing an Order

Primary Actor: Customer
Scope: Restaurant Experience App (DineOn)
Level: User Goal
Stakeholders and Interests:

Customer—wants to order food for their table from their phone
Restaurant—wants the customer's order information

Precondition: User has logged in with the correct credentials, has selected the restaurant to dine at, and is checked into the restaurant with an established session.
Minimal Guarantee: Sufficient logging information will exist so that the restaurant can detect that something went wrong, and ask the user to provide details. The user will then have to order manually.
Success Guarantee: Parse has acknowledged the order, the restaurant receives the order and the client receives confirmation that the restaurant received their order.
Trigger: User has the application open and decides to order food on their phone.
Main Success Scenario:

1. Customer opens up the menu.
2. Customer adds desired menu items to their cart.
3. Customer places order.
4. Parse relays the order to the restaurant application.
5. DineOn notifies the user that their order has been successfully placed and received.
6. Restaurant cooks and brings user their ordered items.

Extensions:

3a. Synchronization failure. Restaurant menu state changes between initial load of restaurant data and order placement.

3a1. Parse identifies synchronization via time stamp.
3a2. Parse sends a push notification to the user notifying the user application of most up to date data.
3a3. User application notifies user of the order difference and asks for the user's approval to change their order.
3a4. User application sends an updated request to the restaurant application completing order request or the user cancels their order.

3b. Network failure. User application sends order request but it is lost in the net.

3b1. 3rd party dependency on Parse to handle network failures, malformed requests.

5a. User gets a phone call or logs out of the application during the middle of the request.

5a1. Parse saves the success notification in the background and loads it when the app is reloaded.

5b. User wants to cancel or modify their order after they placed it.

5b1. The application tells the user to notify the restaurant to cancel or change the order. The restaurant changes or cancels the order using the restaurant app.

6a. Restaurant runs out of one or more ordered items between the order placement and food delivery.

6a1. It is the restaurant’s responsibility to notify and edit the user’s order through the restaurant app and to let them know that they are out of an ordered item.

Formal Use Case: Restaurant Check-in

Primary Actor: Customer
Scope: DineOn App
Level: User Goal
Stakeholders and Interests:

Customer—wants to check in to a restaurant
Restaurant--wants to know what dining sessions are active

Precondition: User has logged in with the correct credentials and has selected the restaurant to dine at.
Minimal Guarantee: Sufficient logging information will exist so that the restaurant can detect that something went wrong, and ask the user to provide details.
Success Guarantee: Parse acknowledges the check-in, sends the check-in to the restaurant application, relays confirmation back to the user application, and establishes a session between the user and the restaurant.
Trigger: User is at a restaurant and decides to check-in on their phone.
Main Success Scenario:

1. User asks to check in to the restaurant.
2. Parse relays the check-in request to the restaurant application.
3. Parse establishes a session between the user and the restaurant and displays the session on the restaurant app.
4. User receives confirmation that the session has been established.

Extensions:

1a. User receives phone call or application closes after check in request is made.

1a1. Parse saves requests in the background and resubmits once the app is reloaded.

1b. Network Failure: user application sends check in request but it's lost in the net

1b1. 3rd party dependency on Parse to handle network failures, malformed requests.

Casual Use Case: Searching for a Restaurant

The user is logged into the DineOn application. The user specifies the name of a restaurant they want to search for. Parse receives the search request and returns the names of any restaurants that meet the user's search criteria from the database. The user can navigate to the resulting restaurants' pages to see more information about the restaurants. If the user logs out or receives a phone call in the middle of their request, Parse will save their request in the background and resubmit it once the app is reloaded. Our application has a 3rd party dependency on Parse to handle network failures or malformed requests.