Restaurant Locator and Booking – Yummy!

Mobile App Project Code 06 – SRS Document

IBM Career Education

Disclaimer

This Software Requirements Specification document is a guideline. The document details all the high level requirements. The document also describes the broad scope of the project. While developing the solution if the developer has a valid point to add more details being within the scope specified then it can be accommodated after consultation with IBM designated Mentor.

Table of Contents

INTRODUCTION

A fine dining restaurant chain, operating from multiple locations in various cities, wanted its customers to locate the nearest restaurant quickly and also see the current waiting time in those restaurant. As an option, a customer could also book a table from the phone only. Therefore, they planned to develop a restaurant locator and booking application, “Yummy!”.

Yummy! will display the nearest restaurant locations on a map on the basis of the user’s current location. Each location will display the approximate waiting time (if any) in minutes. The user may select a location and book a table. The table will be kept reserved up to predetermined time duration from the time of booking. If the customer does not arrive in time, the same will be cancelled; but Yummy! Will provide a push notification 5 minutes before the expiry of booking.

The solution has to be developed using IBM Worklight Studio and will be deployed on IBM WAS, DB2, and Worklight Server. The target device will be an Android Phone. The development will follow the IBM’s Rational Unified Process.

This document is the primary input to the development team to architect a solution for this project.

System Users

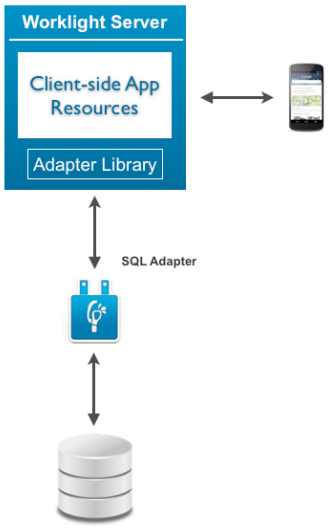
The customers frequenting this chain of restaurants will be the primary user of Restaurant locator and booking system, Yummy!. They will use this app on their smart phone.

Assumptions

1. The scope will be limited to customers viewing the availability of restaurants and booking a table. Features like viewing and ordering from menu are not included in this scope.
2. The application will be targeted for one device type only. From the perspective of ease of availability, an Android device is recommended.
3. The application assumes the availability of an appropriate back-end system. Therefore, it will be essential to create a bare bone skeleton of the same in order to make Yummy! work. All data will be entered from the DB2 backend or uploaded directly in to tables using CSV files.
4. Users of the application are assumed to be pre-registered for accessing the making a booking and accessing queue status.

SYSTEM ARCHITECTURE

Yummy! will be a HTML5 application. Its high-level architecture is illustrated through the following diagram that highlights the key system components and their interactions.



1. The HTML5 based mobile web application is loaded from Worklight Server. It communicates the current location to the server for fetching the restaurants in and around the current location of the user; bookings if any; and booking that have been marked as expired or in waiting with wait time on the mobile device.
2. A SQL adapter uses location information/user id to fetch details/update booking status if any, from/to “Yummy backend data base”.
3. The database in DB2 holds the “Yummy” tables and performs the actions requested by the adapter viz. a) fetching locations, user’s booking request, and b) updating booking status. The booking status through this application can be ‘Waiting’, ‘Ready to Serve’, ‘Expired’.

Please refer to section on Use Cases for more details.

FUNCTIONAL REQUIREMENTS

The high level functional requirements for the Yummy! Mobile App are outlined in this section.

Authentication

Yummy! provide a secure user-id/password based secured login mechanism to access booking status. The development team is expected to create these keeping in mind the general practices followed by the mobile applications.

To maintain focus on the primary objective, you may store user-id/password and role details in clear in the database and also enter data directly in to the table from the DB2 backend.

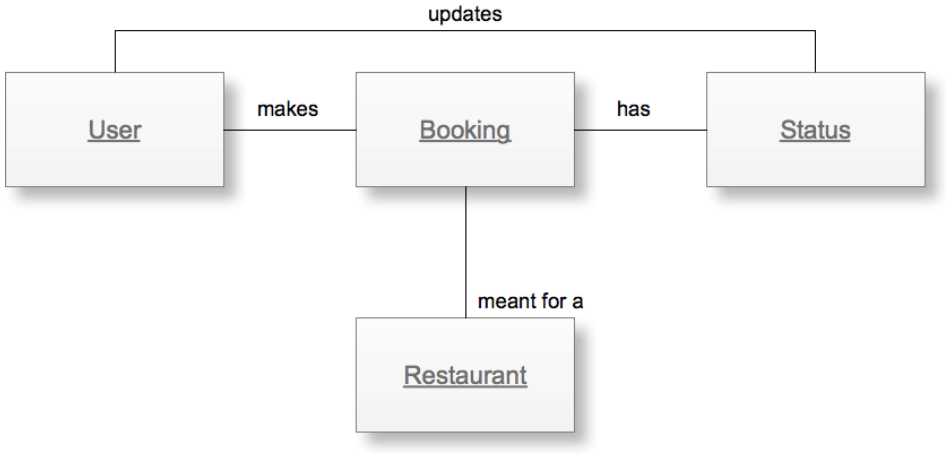
Note: Login to application is required for booking purpose.

Internet-based Restaurant Booking

Yummy! assumes availability of an existing web based restaurant booking application. Therefore, in order to develop and test, it will be essential to have a bare bone booking app that Yummy! can extend to smart phones. It is suggested to create a complete database model on the basis of the object diagram described in the subsequent sub-section.

Object Diagram

A high-level logical object model of the system is shown below. During technical design it will be transformed into a physical model covering all system entities. Such a diagram will include their relationship and its cardinality.



1. User makes a booking for a restaurant in a location.

2. Locations are pre-defined in terms of various localities in the city. To reduce the complexity, assume a single city operation!
3. Restaurant details include address, phone number, location name, latitude, and longitude. The latitude and the longitude are obtained using Google Geocoding API located at https://developers.google.com/maps/documentation/geocoding/. The application displays restaurant address and phone number on touching one of the location from the list displayed on the users mobile.
4. A booking has a user id, date & time, waiting time if any, (defined when the booking is accepted by the restaurant) and a status.
5. Status of a booking keeps getting updated based on the average waiting time of a restaurant.
6. Booking status appears, when the user access the application on their mobile, if the user has made a booking for a restaurant.

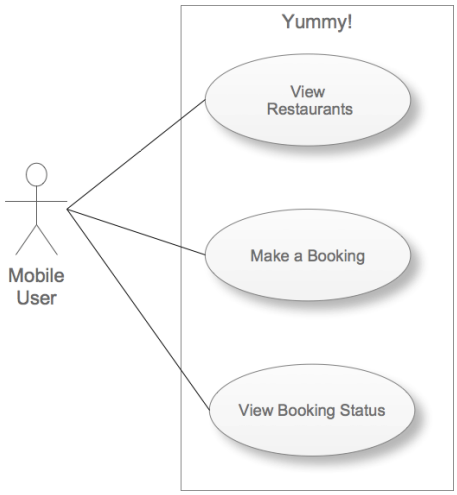
To simulate system operation, dummy entries are made in the tables using DB2 back-end. Yummy! fetches and/or updates the required information from these tables using SQL adapter as highlighted in the section on System Architecture earlier.

Push Notifications

Push Notifications are triggered by the back-end system to notify the user of the booking expiry. The back-end system explicitly pushes notification(s) and the same are retrieved by the Worklight adapter's event source. A push also updates the user's booking status to 'Ready to Serve' or 'Expired' in the app. In absence of necessary prerequisites for notifications, the actual pushing of notification may be ignored.

Use Cases

The following figure illustrates the Use Case diagram for the system.



View Restaurants

Use Case Element	Description
Number	UC.01
Application	To display the list of restaurants falling in and around user's current location.
Primary Actor	Mobile User
Secondary Actor	None
Pre-condition	None
Trigger	User accesses the restaurant list using appropriate gesture on touch screen of the smart phone.

Use Case Element	Description
Basic Flow	a) Smart phone obtains the latitude and the longitude of the current location using the GPS service of the smart phone. b) This information is sent to Worklight server. Server computes minimum and maximum latitudes & longitudes to ensure that these cover a 5Km area around the current location. This is done using Haversine Formula as described in a later section. c) Using minimum/maximum latitudes and longitudes, a list of locations is obtained from the location table. d) Using the list of locations obtained at step (c), user's booking (if any) for those locations are obtained and sent to the smart phone. e) If there is a booking, then Status is displayed on the smart phone.
Alternate Flow	If the GPS services of the smart phone are switched off, the user is prompted to turn on the services.
Output	None

Make a Booking

Use Case Element	Description
Number	UC.02
Application	To make a booking
Primary Actor	Mobile User
Secondary Actor	None
Pre-condition	None
Trigger	User invokes restaurant for the current location using appropriate gesture on touch screen of the smart phone.
Basic Flow	User enters the number of guests and sends the request using appropriate gesture on touch screen of the smart phone. Selected location id along with guest count is sent to the Work light server, where the server computes the waiting time based the average time per booking. The booking wait time is also updated on the smart phone and alerted to the user. The booking status is updated as 'Waiting' in case there is a wait time else its updated as 'Ready to Serve'

Use Case Element	Description
Alternate Flow	None
Output	None

View Booking Status

Use Case Element	Description
Number	UC.03
Application	View Booking Status
Primary Actor	Mobile User
Secondary Actor	None
Pre-condition	None
Trigger	User invokes restaurant for the current location using appropriate gesture on touch screen of the smart phone.
Basic Flow	Selected location along with user id is sent to the Work light server, where the server computes the waiting time based on the restaurant capacity and seats booked and the average time per booking. The same is also updated on the smart phone.
Alternate Flow	None
Output	None

ABOUT HAVERSINE FORMULA

The Haversine formula is an equation important in navigation, giving great-circle distances between two points on a sphere from their longitudes and latitudes. The basic details can be found at Wikipedia entry located at http://en.wikipedia.org/wiki/Haversine_formula URL.

TESTING GUIDELINES

Quality of the software can be achieved with basic hygiene and consistency followed during design and development of User Interface (UI), Navigation, Validations as per the business process requirement. Here are some essential guidelines:

1. Plan on unit testing mobile applications early and frequently during development. This is simply because mobile testing is challenging, given the wide range of possible devices and platforms for even a single application.
2. IBM Worklight Server provides an app preview service that enables you to simulate mobile web artifacts in a desktop web browser. App preview also enables the simulation of Worklight client APIs and it is available from the Worklight Studio test server console. Use it as a quick and easy way to test your app during early stages of development.
3. Use native mobile SDK simulator to perform near real life like testing on your desktop.
4. Finally use the real device to test your app. For example, when an Android device is connected to the computer via a USB cable, the Eclipse ADT plug-in automatically recognizes it and attempts to deploy applications onto it.
5. In order to test your mobile application with full back end enterprise system integration, the application needs to be installed in an integrated environment having all the components. Plan for setting up such environment at an early stage.
6. Testing push notifications, if applicable, will require necessary account set up for the target device(s). Some key examples are Android Cloud to Device Messaging Framework (now Google Cloud Messaging – GCM) or Apple Push Notification service (APNs for short).

SUGGESTED READING

The project is aimed at making the student understand concepts of (a) Design and Development using IBM Worklight, IBM Rational tools, Web Sphere Application Server and DB2 Database; and (b) User Interface Design for Mobile Devices.

Tools

The following reading reference is easy to understand and should be read to get a clear understanding of capabilities of the tools and how you would leverage them to execute a project.

Resource	URL
RAD - Tackling challenges of software development with Rational ApplicationDeveloper for WebSphere Software	http://www.ibm.com/developerworks/rational/library/08/0926_ackerman-mahate/index.html
IBM Education Assistant – IBM Worklight V5	http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.worklight/worklight/5.0/Back-End_Connectivity/Worklight_Backend_Connectivity/player.html
Getting started with IBM Worklight	https://www.ibm.com/developerworks/mobile/worklight/getting-started/

User Interface

Designing a UI for mobile devices is not identical to desktop or web application UI design. It is not about simply adding controls, some text and color to each screen. Here gestures play a vital role, chrome has to be minimized. The following UI design guidelines, developed by Google for Android and by Apple for iOS, will give you insight in to how to create great UI for mobile applications.

Resource	URL
Google Android UI Guidelines	http://developer.android.com/guide/topics/ui/index.html
Apple iOS Human Interface Guidelines	http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html