Smart Waiter Detailed Design

Meraj Patel #1137491
Pavneet Jauhal #1149311
Shan Perera #1150394

January 11, 2016

Contents

List of Figures

List of Tables

Template

This document uses Volere Template for its organization.

1 Introduction

1.1 Purpose

The purpose of Smart-Waiter aims to provide a solution that will allow users to order and pay through a mobile application at restaurants.

1.2 Description

This opportunity arose from the lack of a universal application in the market that allows users to walk into a restaurant, scan a code to view the menu, and proceed to order and pay through the use of a singular application. Android users will be able to walk into any restaurant that offers our solution, and have the ability to use these services.

1.3 Scope

The scope of Smart-Waiter will be limited to providing the user with the following features: viewing the restaurant's menu, creating the user's order, placing the order and paying for their order.

2 Overview

2.1 Design Principals

2.1.1 Information Hiding

Information hiding is the principle of segregation of design components that are likely to undergo changes as the lifecycle of the application progresses. This way, other key parts of the application will remain unhindered by any possible changes.

3 Development Details

3.1 Language of Implementation

The source code of the Android Smart-Waiter application will be written in Java and XML, as well as the API libraries used in development.

3.2 Supporting Frameworks/APIs

CouchbaseLite (Database) ZXing Embedded (QR Code Scanner) Stripe (Credit Card)

4 Implementation Components

4.1 Camera Function

In this section we will detail the function that relates to the camera aspect of Smart-Waiter. There is one main function necessary for QR code scanning: onActivityResult. onActivityResult is called as soon as a QR code is captured by the camera. The function is passed three variables, a requestCode (int), a resultCode (int), an intent (Intent). Using these three variables, the function calls the ZXing function parseActivityResult using the three aforementioned variables as parameters. The parseActivityResult returns the contents of the QR code.

function onActivityResult (requestCode, resultCode, intent) IntentResult contents = parseActivityFunction (requestCode, resultCode, intent) if(contents is not null) String QRContents = contents.getContents() Call populate menu functions using QRContents as a parameter

4.2 Account Function

In this section we will detail the key function that relates to account transactions and credit card charges. There is one key method related to credit card transactions: chargeParams, it involves using the Stripe class. As described in the System Architecture, given an instance of the Card class, and using Smart-Waiter's Stripe API key, a token is created and sent to the Stripe servers. The Stripe servers return a token that can be used to charge the user's credit card. The token is then used as a parameter of chargeParams, along with the information related to the credit card charge, like amount and the type of currency.

Stripe = new Stripe (Smart-Waiter's Stripe API key) createToken(card, tokenCallback())

token = request the Stripe token from Stripe server

try chargeParams(amount, integer amount in cents) chargeParams(currency, string value of type of currency) chargeParams(source, token) chargeParams(description, string value of description regarding charges to card)

charge = Charge.create(chargeParams) catch (Exception e) Error handling

4.3 Ordering Functions

Below are function descriptions that are necessary for ordering in SmartWaiter. As explained in System Architecture, there are three primary classes used to hold all vital information regarding menu information. These are: MenuCategories, MenuItems and User. These three classes are used to provide a user with a full menu of a particular restaurant and allow them to place an order.

4.3.1 Parsing Function

A parsing function was created to parse through received JSON response to save information in appropriate classes. Stated below is sudo code for this JSON parser. This function will parse through a JSON request. View if the key equals ”categoryname”. If so, save the value within MenuCategories class under . Also, if the key equals ”categorypic” save the value within MenuCategories class under pi

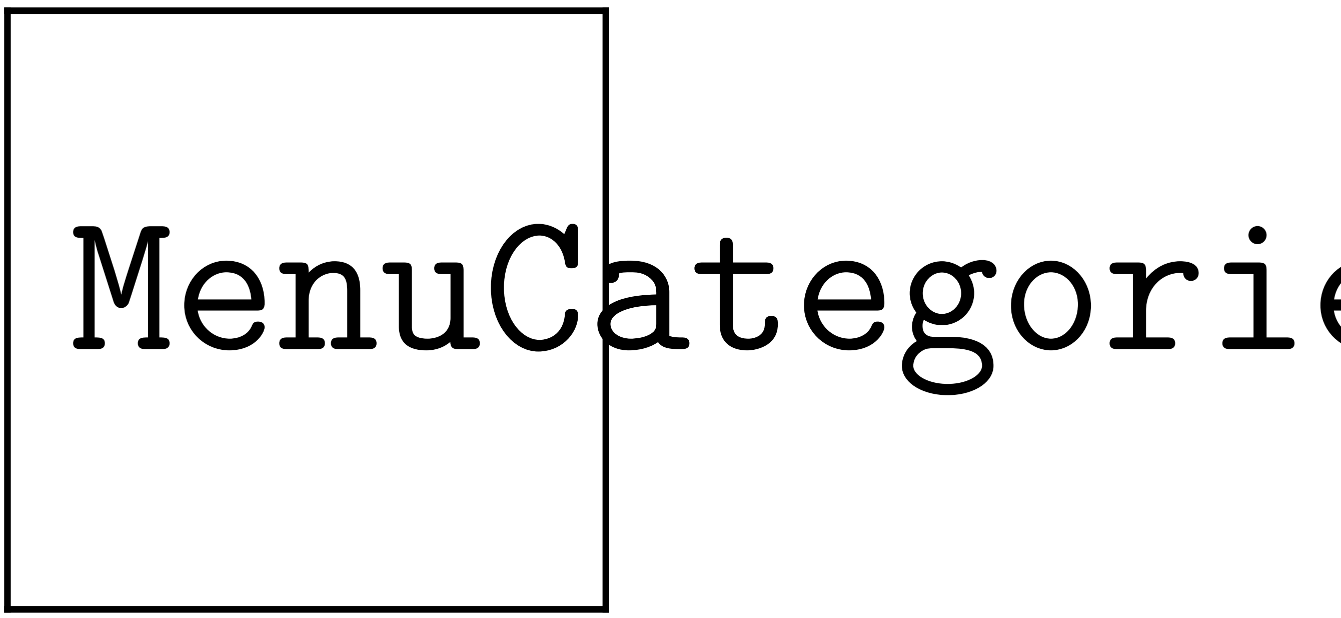
function parse throws exception corrupt define categoryArrayList Save JSON response as list of hash map array for (each element in hash map list) while (each array element with list) if(key equals categoryname) Create MenuCategories object, save category name if(key equals ”categorypic”) use MenuCategories object, save category name   add created MenuCategories object to categoryArrayList

4.3.2

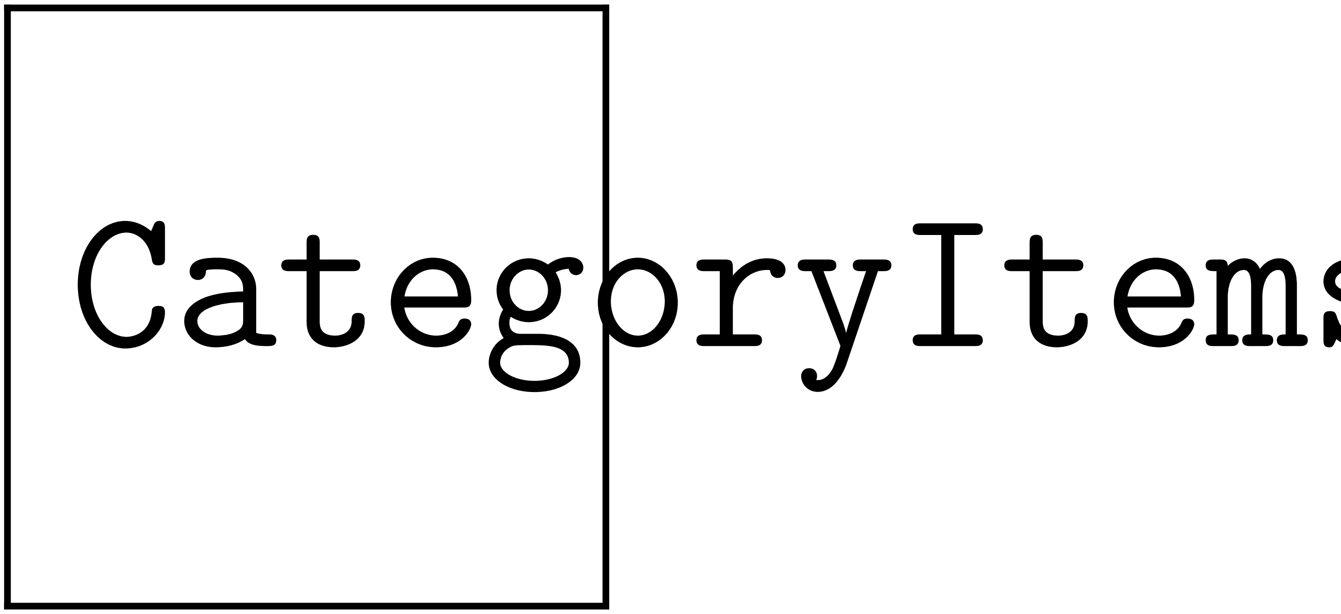
5 User Interface Design

5.1 User Interface Design Overview

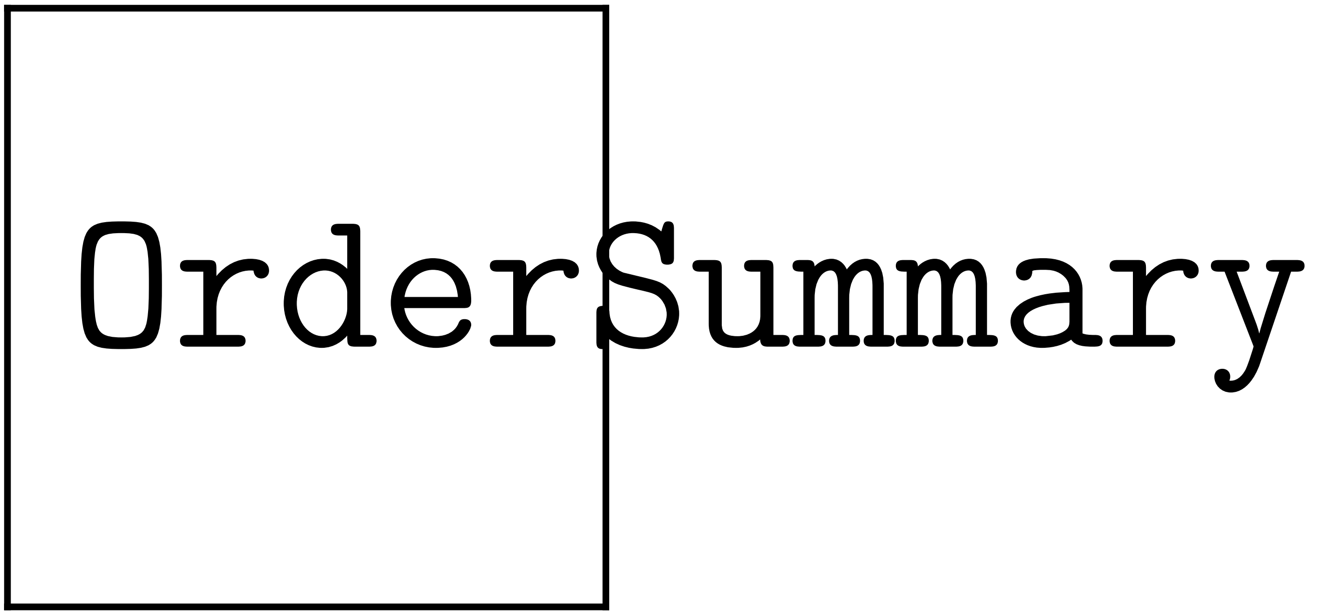
5.1.1 Menu Categories



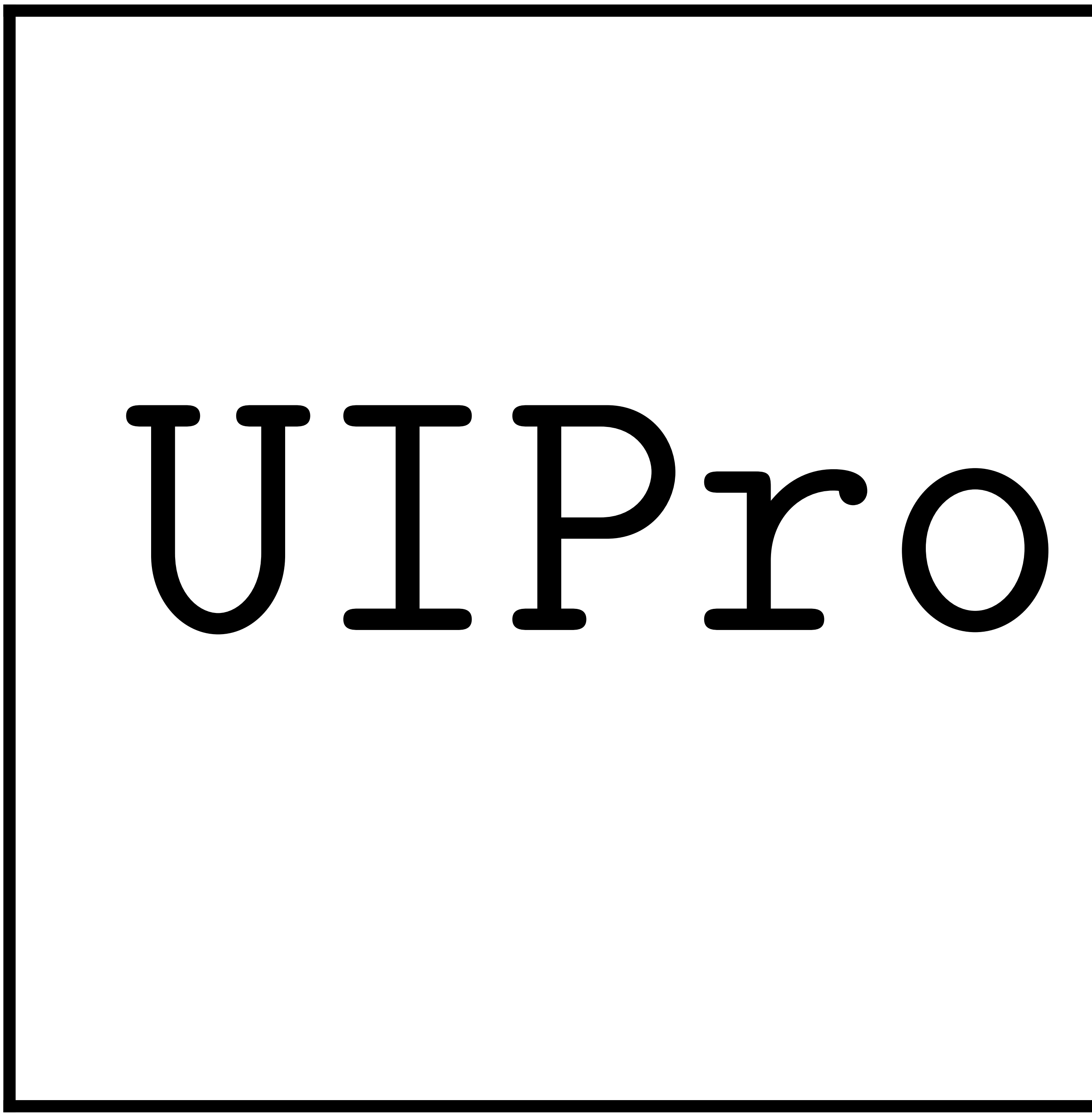
5.1.2 Menu Items



5.1.3 Confirm Order



5.2 User Interface Navigation Flow



5.3 Use Cases

5.3.1 Sign In Page

User Input	System Response
Enters correct user name and password	Application transitions to Barcode Scan page
Enters incorrect user name and password	Toaster displayed reading ”incorrect login, please try again”
Enters incorrect user name	Toaster displayed reading ”incorrect login, please try again”
Enters incorrect password	Toaster displayed reading ”incorrect login, please try again”
Clicks ”Skip Sign in”	Application transitions to Barcode Scan page
Clicks Back Button on phone	Application Quits

5.3.2 Barcode Scan Page

User Input	System Response
Clicks Scan Barcode	Application transitions to camera so user can scan code. If successful, application will transition to menu page. Otherwise will return to scanning page and display a toaster reading, ”please try again”
Clicks back button on phone	Application transitions to Sign in Page

5.3.3 Menu Categories Page

User Input	System Response
Clicks category	Application transitions Category Items
Clicks back button on phone	Application transitions to Barcode Scan

5.3.4 Category Items Page

User Input	System Response
Clicks Item	Application transitions to Customize Item
Clicks back button on phone	Application transitions to Menu Categories

5.3.5 Customize Item Page

User Input	System Response
Ticks check boxes	None
Enters special instructions in input field	None
Clicks ”Add to Cart”	Transitions to cart page and populate list with item
Clicks back button on phone	Application transitions to Menu items

5.3.6 Cart Page

User Input	System Response
Clicks ”Delete”	Deletes item from list
Clicks ”Submit Order”	Transitions to payment page
Clicks back button on phone	Application transitions to previous page

5.3.7 Payment Page

User Input	System Response
Input valid credit card and clicks ”Process”	Transitions into Confirm Order page
Input invalid credit card and clicks ”Process”	Toaster displayed reading ”invalid credit card”
Clicks back button on phone	Application transitions to previous Cart Page