Smart Waiter System Architecture

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

January 10, 2016

Contents

1 Introduction **3**
- 1.1 Purpose . . . 3
- 1.2 Description . . . 3
- 1.3 Scope . . . 3

2 Overview **3**

3 System Architecture **3**
- 3.1 Overall Operation . . . 3
- 3.2 Order . . . 3
- 3.3 Database . . . 4
- 3.4 Processing Order . . . 4

4 Orders **4**
- 4.1 Overview . . . 4
- 4.2 Menu Class Structure . . . 4
  - 4.2.1 MenuCategories Class . . . 4
  - 4.2.2 MenuItems Class . . . 5
  - 4.2.3 User Class . . . 5
- 4.3 Camera Structure . . . 6
  - 4.3.1 Design Principles . . . 6
  - 4.3.2 onActivityResult . . . 6
  - 4.3.3 Module Decomposition . . . 7
- 4.4 Accounts Structure . . . 7
  - 4.4.1 Design Principles . . . 7
- 4.5 Transactions Structure . . . 9

5 User Interface Design **9**
- 5.1 User Interface Design Overview . . . 9
  - 5.1.1 Menu Categories . . . 9
  - 5.1.2 Menu Items . . . 10
  - 5.1.3 Confirm Order . . . 11
- 5.2 User Interface Navigation Flow . . . 12
- 5.3 Use Cases . . . 12
  - 5.3.1 Sign In Page . . . 12
  - 5.3.2 Barcode Scan Page . . . 12

5.3.3 Menu Categories Page 13
5.3.4 Category Items Page 13
5.3.5 Customize Item Page 13
5.3.6 Cart Page 13
5.3.7 Payment Page 14

List of Figures

List of Tables

1 Revision History Table 2

Revision History

Date	Comments
October 9, 2015	first draft.

Table 1: Revision History Table

Template

This document uses Volere Template for its organization.

1 Introduction

1.1 Purpose

1.2 Description

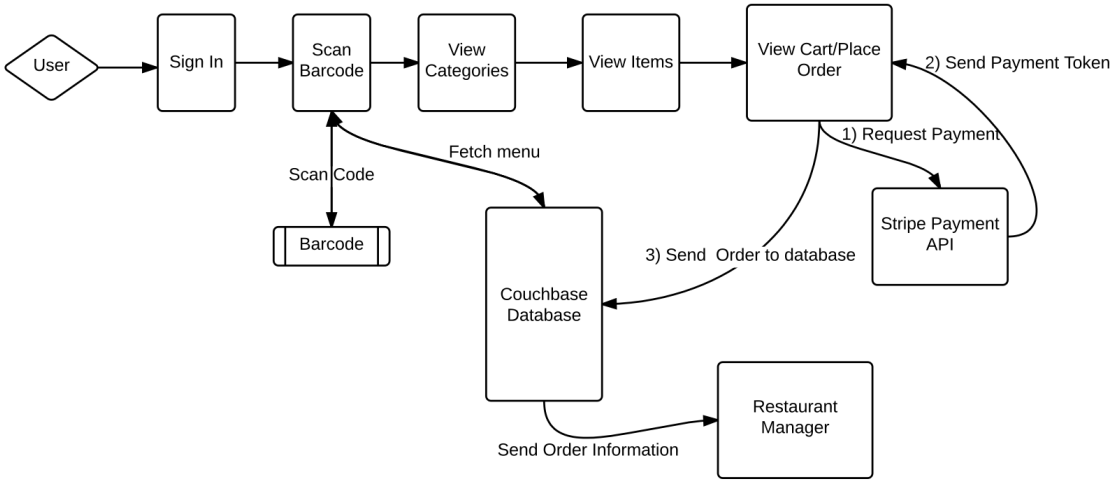
1.3 Scope

2 Overview

3 System Architecture

3.1 Overall Operation

Below is a high level overview of Smart Waiter operation. The operation is decomposed into three components which is introduced in this section and will be thoroughly explained in the following sections.



3.2 Order

This component encompasses all operations needed to allow a user to place an order. This includes, user sign in and all operations till placing the order (top segment of operation diagram). The following components are described in detail in section 4 of this document:

1. Menu Class Structure
2. Barcode Scanning
3. Accounts
4. Transactions

3.3 Database

3.4 Processing Order

4 Orders

4.1 Overview

Below are key details of ordering component for Smart Waiter application. Detailed architecture review is provided.

4.2 Menu Class Structure

There are three primary classes used to hold all vital information regarding menu information. These are: MenuCategories, MenuItems and User. Each class purpose and its correlation with each other is explained below.

4.2.1 MenuCategories Class

This class purpose is to store menu category information of a restaurant menu. This entails category name, picture and a reference to category items. To do so, there are three main variables used within this class to hold this information:

categoryName: Stores category name in the form of String

picURL: Stores URL of category picture

categoryItems: Stores an array of objects that reference MenuItems class

This class is instantiated and called upon when a user successfully scans a barcode. The JSON response acquired from the database is parsed, and information related to menu categories is stored appropriately within this class. The application uses this class to display category information when "Menu Categories" page is spawned (please see section 5.1.1 to view picture).

4.2.2 MenuItems Class

This class purpose is to store menu item information of a restaurant menu. This means, item name, price and description. Thee main variables are used within this class to hold this information.

itemName: Stores item name

itemPrice: Stores item price

itemDetail: Stores description of item

Objects of this class are instantiated from the MenuCategories class. By successfully scanning the barcode and parsing the JSON response, MenuItem objects are instantiated and used to save item details. These objects are held within categoryItems list seen in Menu Categories class. This application uses this class to display item information of a category when "Menu Items" page is spawned (please see section 5.1.2 to view picture). These objects are also used in reference to items the user would like to order. This is thoroughly discussed in the proceeding section.

4.2.3 User Class

User class is used to store menu item a user would like to order. For now, there is only one important variable to consider:

userItem: an array that stores objects of MenuItems class. This is used to save menu items a user would like to order.

Only one object is instantiated through the life time of this application. This object represents the user cart of menu items he/she would like to order. This object is called upon when a user decides to add an item to their cart. When this occurs, a copy of the MenuItem object is saved within the userItem variables. This way, there is a list maintained to hold all item

a user would like to order. When a user confirms and sends their order, the item information is extracted from userItem list, formatted into a JSON request and send back to the database for further processing. Having this class implemented allows extendibility in the future, as all vital information pertaining to a user can be stored within the class (for example, user settings).

4.3 Camera Structure

There is one main function related to the Camera structure: onActivityResult. The camera is used to read QR codes, which contain the location of where the menu data is stored in relation to the current restaurant. The QR codes are read using an embedded version of the ZXing library. When the application starts, a ZXing specific variable (IntentIntegrator) is initialized, when the user clicks the "Scan QR Code" button on the first screen, the Camera is initialized.

4.3.1 Design Principles

This component follows the Single Responsibility and the Information Hiding principle.

Single Responsibility: The responsibility of this component is to capture and parse QR code data. If this component had to be changed, it would only affect the camera functionality of the application, and not other components like the menu or accounts.

Information Hiding: The camera component is separate from every other component of this application. If any changes were to be made to the camera application, it will not affect the other components of the application. However, it is unlikely that there will be any changes applied to this component of the application.

4.3.2 onActivityResult

This function uses an integrated version of the ZXing library to scan QR codes. After the camera is initialized, the application waits for the user to pass a QR code, this is done by taking a picture of the QR code with the camera. The function onActivityResult is called as soon as a QR code is passed. It takes, as input a requestCode, resultCode, and the current intent:

requestCode: Used to parse the QR code

resultCode: Used to parse the QR code

intent: Gives the current intent, used to parse the QR code

Using the given input, the function parses and gets the contents of the QR code that was passed to the function using the ZXing function parseActivityResult. The information stored in the QR code is then saved to a local variable to be used by the other components of the application.

4.3.3 Module Decomposition

Module: onActivityResult

Secret: How the QR code is parsed and read

Service: Parses and gets the content of the QR code

4.4 Accounts Structure

There are 3 main classes related to the Accounts structure: User, Card, and Stripe. Accounts are used to associate a user identifier to a user, so that data can be used for transactions and for storing a user's credit card information securely. Credit card information transfer and storage uses Stripe API. This reduces the burden on the developers, since we can use an established API that follows the restrictions applied by credit card companies, and handles the secure transfer of sensitive information, rather than developing an efficient and secure system ourselves.

4.4.1 Design Principles

This component follows the Single Responsibility and the Information Hiding principle.

Single Responsibility: The responsibility of this component is to securely store a user's credit card information and to associate this information with a user identifier for use in transactions. If this component had to be changed, it would only affect the transactions class functionality of the application, and not other components like the menu or camera.

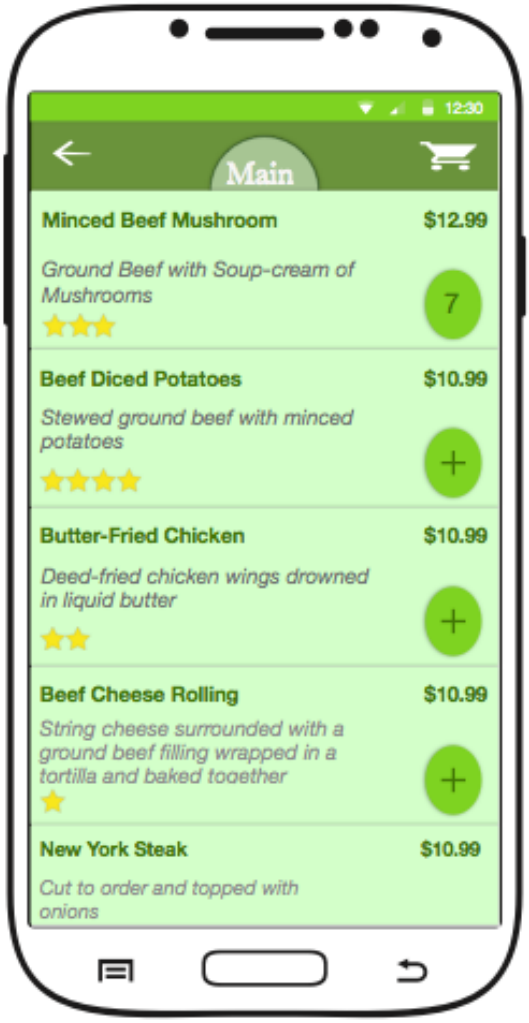
4.5 Transactions Structure

5 User Interface Design

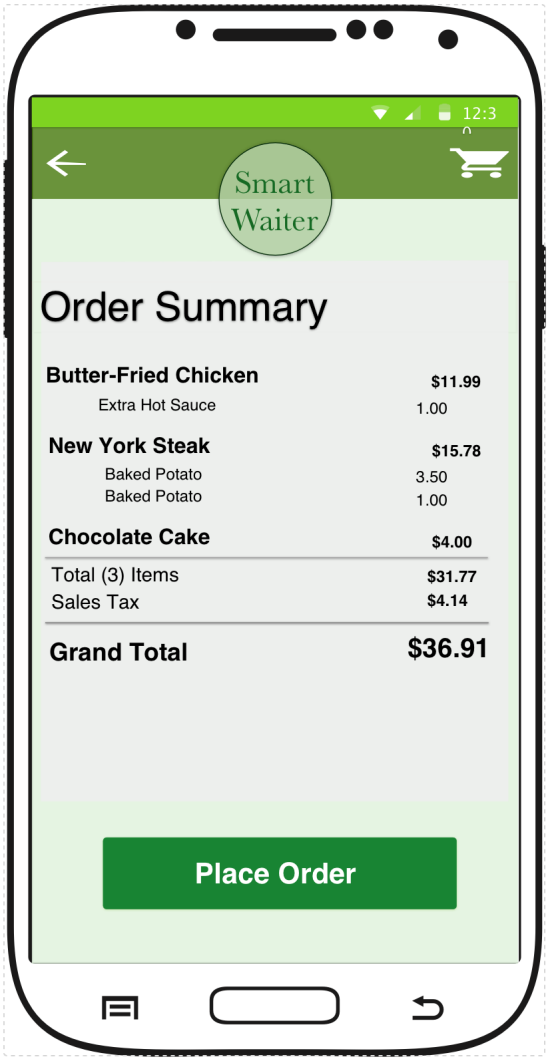
5.1 User Interface Design Overview

5.1.1 Menu Categories

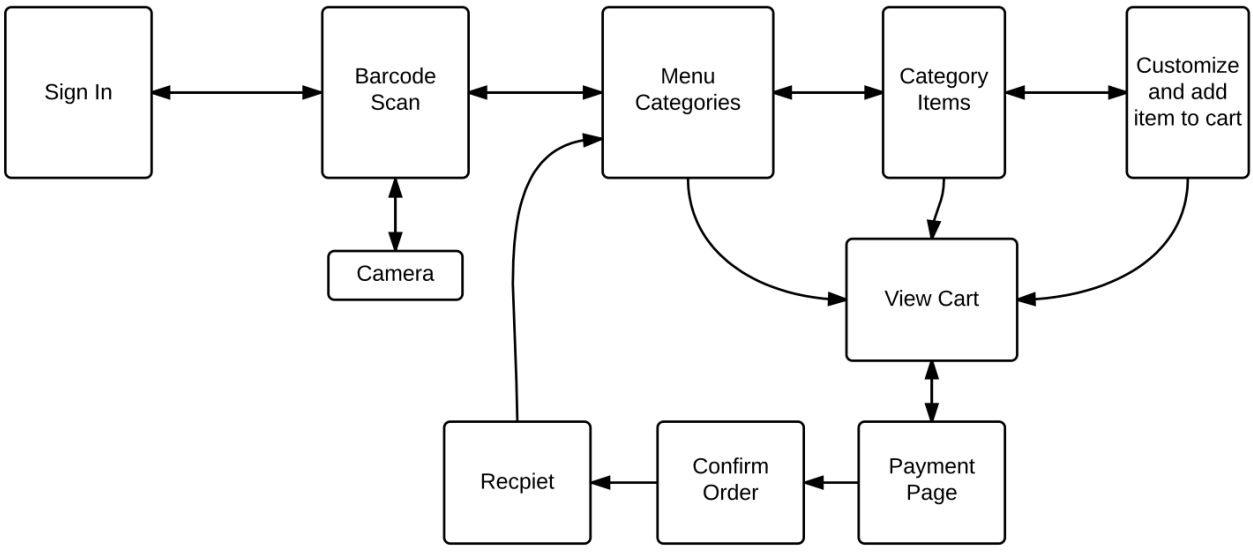
5.1.2 Menu Items



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