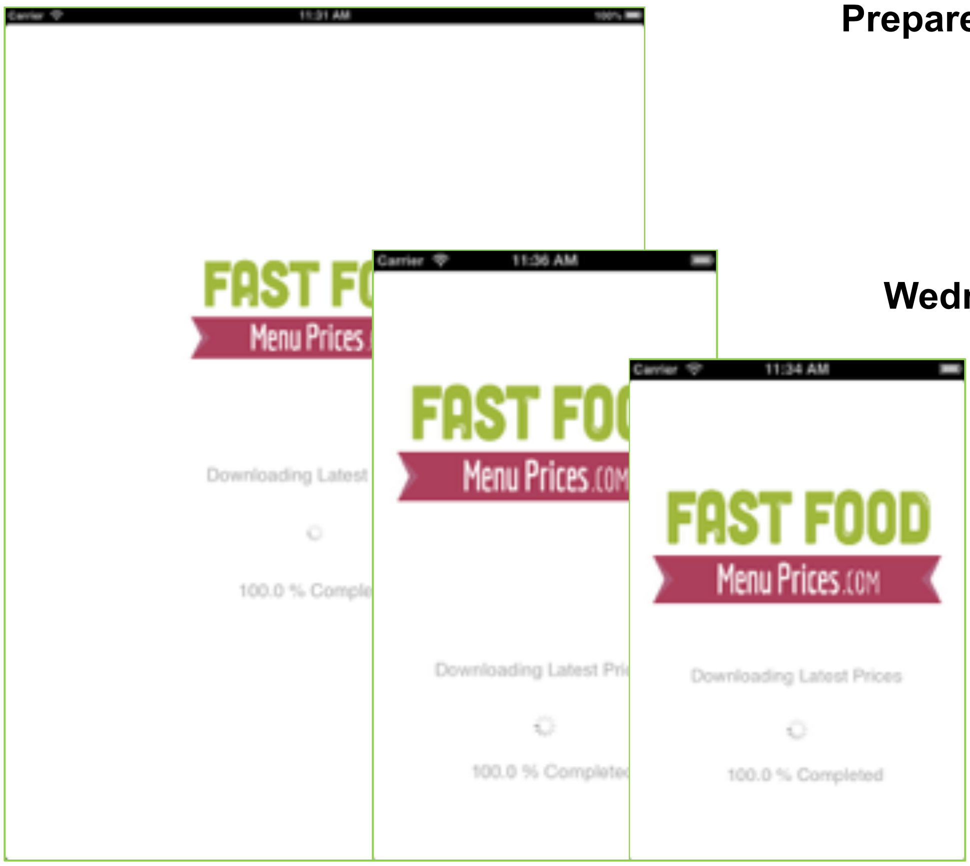
Software Requirements Specification

For

Fast Food Menu Prices

Version 2.0



Prepared by Christopher Miller

iOS Operators

Wednesday, October 9, 2013

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Christopher Miller	10/07/2013	Initial/Introductory Draft Completed	1.0
Susan van de Ven	10/07/2013	Verification 1 of Initial/Introductory Draft	1.1
Frank Tsui	10/07/2013	Verification 2 of Initial/Introductory Draft	1.1
Abi Salimi	10/07/2013	Verification 3 of Initial/Introductory Draft	1.2
Christopher Miller	10/07/2013	Update 1 to Document/Specification	1.3
Admir Salcinovic	10/07/2013	Validation of Update 1 of the SRS	1.3
Christopher Miller	10/08/2013	Update 2 to SRS, Overall Description	1.4
Sheryl Duggins	10/09/2013	Verification 4 of Overall Description	1.4.4
Christopher Miller	10/09/2013	Update 3 SRS, Overall Description	1.5
Admir Salcinovic	10/09/2013	Validation of Overall Description	1.5
Christopher Miller	10/13/2013	Update 4 to SRS, High Level Requirements	1.6
Jonathan Lartigue	11/13/2013	Verification 5 of High Level Requirements	1.6.1
Christopher Miller	11/14/2013	Update 5 to SRS, Requirements/Use Cases	1.6.2
Admir Salcinovic	11/27/2013	Final Validation of Document/ Specification	2.0

1.Introduction

1.1.Purpose

The purpose of this software specification document is to define the components/requirements for creating an iOS version of Fast Food Menu (FFM) Prices. This document will outline all of the necessary functions to complete the FFM Prices application. The design of the application is not purposed in this document.

1.2.Intended Audience and Reading Suggestions

The intended audience for this document is iOS Operator (Christopher Miller), Admir Salcinovic (the project sponsor), and Jonathan Lartigue (the development advisor).

This SRS document will be broken up into the following sections: Project Description, Glossary, System Features, External Interface Requirements, and Non Functional Requirements.

1.3.Product Scope

The FFM Prices project's resources are managed at fastfoodmenuprices.com. This data is to be saved into the iOS application's local memory for day-to-day price checking.

The objective is to take the FFM Prices existing Android application features and convert them to an iOS application. The application shall be compatible with iPhone 5, iPhone 4, and iPad screen sizes. The application shall have Google AdMob, and Google Analytics services included.

1.4.Document Definitions

Application Archive	*The executable that runs on a registered iPhone/iPad*
API	*FFMPrices Application Programming Interface with Cloud Endpoints*
Device	*The mobile device that runs on the system (e.g, iPhone/ iPad)*
iOS	*Interim Operation System (non-admitting)*
IDE	*Integrated Development Environment (i.e., Eclipse, Visual Studio, etc.)*
GIT	*A revision control and source code management (SCM) system with an emphasis on speed.*
FFM	*Fast Food Menu – The application in development*
FFMPrices	*The Fast Food Menu Prices Web* Site/*Resource*
Functional Requirement	*Defines a function of a system or its component*
Non-Functional Requirement	*Specifies a criteria that shall be used to judge the system operation*
Patron	*A fast food customer that views restaurants of their preference*
SDK	*Software Development Kit (as in iOS SDK)*
SRS	*Software Requirements Specification*
UDID	*Unique Device Identifier*
UI	*User Interface*

1.5.References

All devices that will use the application, outside of the App Store installations will need to have their device's UDID listed below for registration (and testing purposes):

Christopher Miller's iPhone	XXXXXXXXXXXXXXXXXXXXXXXXXXXXX69a

Acquire Apple UDID: https://github.com/spsumac/Tool-Kit/blob/master/README.md

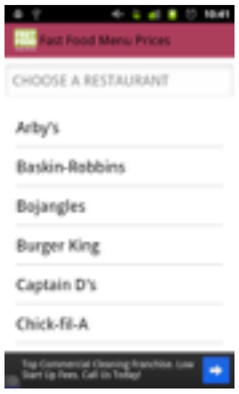
The system will use Google Analytics and Google Admob. The following are the ids for the respective web services:

Analytics Tracking ID	XXXXXXXXXX-3
Admob Unit ID	xxxxxxxxxxxxxe2

2.Overall Description

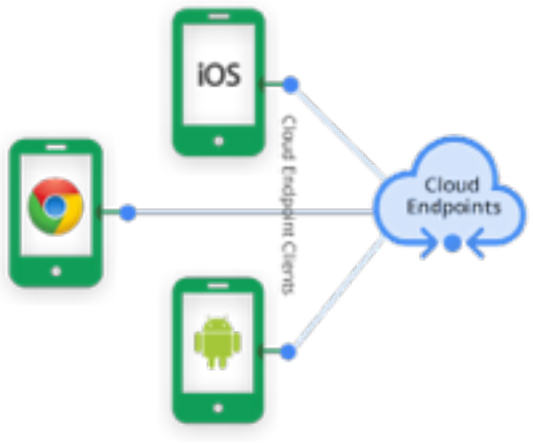
2.1.Product Perspective

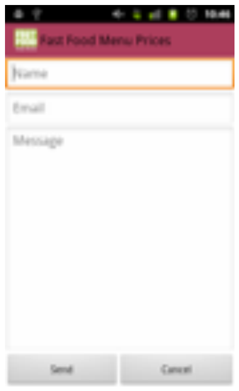
This iOS application is the third client component of the Fast Food Menu Prices System. The FFM Prices iOS app gains its origin from the site (fastfoodmenuprices.com), and the Android client implementations (as well as popular demand). The FFM Prices site has an interface in the form of a Web API (Cloud Endpoints), which holds all restaurant data and prices, and allows for immediate correspondence with the FFM Prices team.



Arby's
Baskin-Robbins
Bojangles
Burger King
Captain D's
Chick-fil-A

Top Commercial Cleaning Franchise. Low Start Up Fees. Call Us Today!





2.2.Product Functions

A Patron is given a list of restaurants to select from and has the option to view each restaurant's current menu prices. The Patron can also contact the FFM Prices team for questions or suggestions through the email screen. These actions get accomplished through online resources.

There are two files that allow for online communications with the FFM site. These files allow for the restaurant data to be downloaded (from the cloud) and displayed, as well as contact the FFM Prices team for help (through the cloud as well).

2.3.User Classes and Characteristics

Patron	A Patron is a fast food customer that views restaurants of their preference. Patrons will sometimes view multiple meals for group events or guests. All Patrons have access to the menu prices from any device. A Patron shall be able to make comments about the service, and ask questions as well. There are millions of potential Patrons, of which an estimated 100 will download from Google Play in 4 weeks time (source: current play.google.com installs) with a current average of 25 install per week.

2.4.Operating Environment

The iOS application shall be compatible with the iPhone 5, the iPhone 4, and all iPads. The operating systems that the application shall be able to run on are iOS 5, iOS 6, and iOS 7.

2.5.Design and Implementation Constraints

The application development will be completed on A Mac Book Pro (running *Mac OS X Lion 10.7.5*). The XCode IDE is the working environment that will be used for development, and this version of XCode is 4.6.2 (4H1003).

The system is dependent on the content of the PHP web server. The server must be able to process multiple requests while sharing server execution time and memory resources with other web applications and instances of the same application.

The development is in the XCode environment with a GIT repository system, for keeping a history of changes to the application and its resources. FFMPrices' dominant color is 172, 62, 92 (#AC3E5C), the accent is 159, 180, 77 (#9FB44D), & the 2nd accent is 124, 137, 74 (#7C894A).

3.Specific Requirements

3.1.External Interface Requirements

3.1.1.User Interfaces

All interfaces that are opened shall be tracked using the Google Analytics Web Service. When the screens appear in the application the name of that screen is logged. Furthermore, the system has 4 screens that can be described as follows:

Splash Screen

The splash screen is the landing screen that initially loads the restaurants.

This screen is the first screen that the Patron sees when starting the application. From the splash screen, the Patron will see the status of the content being loaded.

After the loading is completed, the system dismisses it.

Restaurants Screen

The restaurants screen displays the list of filtered restaurants.

This screen is the second screen that the Patron sees after starting the application. From the restaurants screen, the Patron shall view the list of displayed restaurants and can type into the search box to filter the list.

From this screen the Patron may select one of the restaurants. By selecting a restaurant it opens the third screen (The Menu Items Screen).

Menu Items Screen

The menu items screen displays the restaurant menu and its prices.

The menu items screen is the third screen the Patron sees after starting the application, and selecting a restaurant. From the menu items screen, the Patron will see the list of Menu Groups that organize the Menu Items.

3.1.2.Hardware Interfaces

The project shall be developed in iOS 6.1 SDK, but has a Deployment Target of iOS 5.1. This means the iOS application shall be compatible with the iPhone 5 (runs iOS 6+), the iPhone 4 (runs iOS 4+), and iPads (which run iOS 5.1+), and finally the iPad Mini (which runs iOS 6+).

The application uses network hardware to make connections with an external server with PHP scripts or cloud end points contained in two files.

3.1.3.Software Interfaces

The iOS SDK (v 5.1) is the primary API used, alongside the Google AdMob iOS API (v 6.5.1), and the Google Analytics iOS API (v 2.0), the application also uses cloud end points (PHP scripts).

The Google AdMob SDK for iOS makes it easy for native iOS developers to allow a Patron to view advertising content that interests them through banners. Google AdMob banners use a small portion of the screen to entice "click throughs" to a richer experience such as an app store page.

The Google Analytics SDK for iOS makes it easy for native iOS developers to collect Patron engagement data from their applications. We can use the Google Analytics reports to measure the number of active Patrons which are using the application, and from where in the world the application is being used, as well as many other important features.

The external server PHP scripts allow for the restaurant data to be downloaded and displayed, as well as contact the FFM Prices team for help. The names of the two external server files are dataController.php and contact_email.php. The contact_email.php script needs data but the dataController.php script does not need input to operate.

3.1.4.Communications Interfaces

On the FFM Prices Site (fastfoodmenuprices.com), the contact_email.php file is a PHP script that requires three parameters for it to work properly. The first of the three parameters is a name for the emailing Patron. The second is the Patron's email address they want to be reached at for a response. The third parameter is the actual message that the Patron wants to convey.

Application Email Resource: contact_email.php?email=some_email@email.com&name=Some%20Name&message=Hi%20there!

The usage of this file requires that the user types a valid email address, and also the server as well the contact_email PHP script have to be fully functional.

On the FFM Prices Site (fastfoodmenuprices.com), the dataController.php file is a PHP script that requires no parameters for it to work properly.

Application Data Resource: getdata/dataController.php

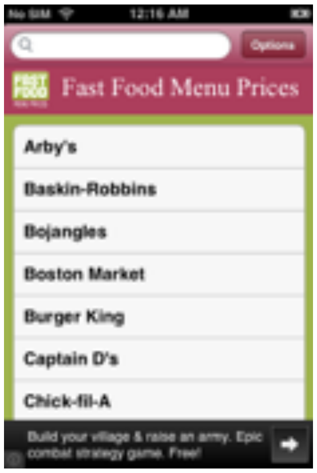
The Google Analytics SDK for iOS sends what the Patron has done within the application. One such activity is clicking on a restaurant and opening the Menu Items Screen. This is primarily possible through the use of an Analytics Tracking ID.

The Google AdMob SDK for iOS sends data that the Patron provides by "clicking through" on some banner advertisement. This function cannot exist without the use of an AdMob Unit ID.

3.2.Functional Requirements

3.2.1.Patron Use Cases

3.2.1.1.Display Restaurants



This use-case realizes the display function of the system. The Patron opens the system and can see the list of restaurants. ~~If the device is not connected to a network the data will not be downloaded initially~~.

Actors: Patron, System, Device, ~~API~~
Pre-Conditions: Patron has the System installed on device
Post-Conditions: Patron can view list of Restaurants

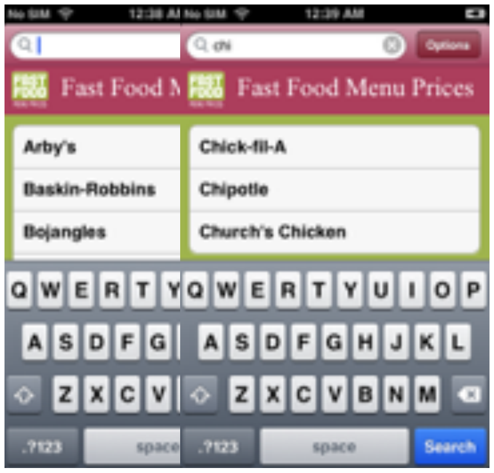
Successful Path (Redacted: Due to preload of data into app):
1. Patron clicks System icon on Device home screen.
2. ~~System retrieves list of restaurants from API endpoint~~
3. ~~API Server queries database and creates JSON object~~
4. ~~System updates Percentage of Restaurants Saved from JSON object~~
5. ~~System opens Restaurants Screen after all restaurants load~~
6. System displays list of restaurants

Alternate Path (~~Previously retrieved~~ Preloaded restaurants):
1. Patron clicks System icon on Device home screen.

2. System opens Restaurants Screen after verifying previous load
3. System displays list of Restaurants

Exception(s):
~~Device cannot initially connect to network~~
~~API endpoint cannot generate JSON object~~

3.2.1.2.Find Restaurant



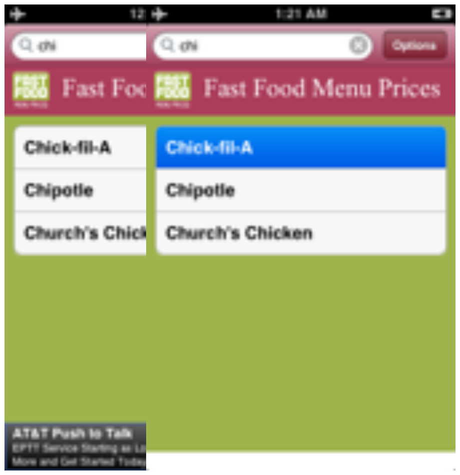
This use-case realizes the search function of the system. The application is open and the restaurants are displayed. If the Patron chooses to search for a restaurant instead of scrolling to it in the list, they can type the name and view names related to the search query.

Actors: Patron, System
Pre-Conditions: Patron has displayed restaurants
Post-Conditions: Patron finds restaurant in list of Restaurants
Successful Path:
1.Patron scrolls through list of available restaurants
2.Patron views restaurant

Alternate Path (Patron wants to filter list):
1. Patron searches application
2. Patron types a character into search
3. System updates list of Restaurants displayed
   a. Patron returns to step 2 until satisfied
4. Patron views restaurant

Exception(s):
System does not contain restaurant

3.2.1.3.Select Restaurant



This use-case realizes the select for details function of the system. The application is open and the restaurants are displayed. If the Patron chooses a restaurant after finding it in the list of restaurants, then the system opens the Menu Items Screen.

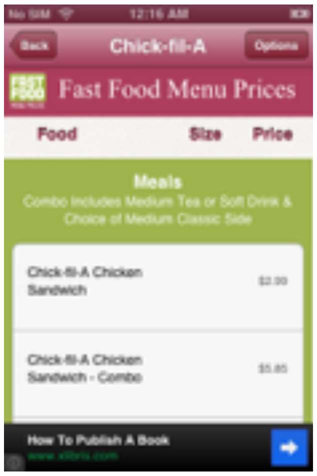
Actors: Patron, System
Preconditions: Patron has found restaurant
Post-Conditions: System displays a restaurant in Menu Item Screen

Successful Path:
1. Patron selects desired restaurant
2. System opens Menu Items Screen

Alternate Path (Wrong restaurant opened):
1. Patron returns to Restaurants Screen.
2. Patron selects correct restaurant
3. System opens correct Menu Items Screen

Exceptions:
Patron selects incorrect restaurant

3.2.1.4. Display Menu Items


No SIM
12:16 AM
Back
Chick-fil-A
Options
Fast Food Menu Prices
Food
Size
Price
Meals
Combo Includes Medium Tea or Soft Drink & Choice of Medium Classic Side
Chick-fil-A Chicken Sandwich
$2.99
Chick-fil-A Chicken Sandwich - Combo
$5.85
How To Publish A Book
www.xlibris.com

This use-case realizes the display details function of the system. The application is open, and the Patron chose a restaurant. After the restaurant is selected the system displays its related items in a list of Menu Groups.

Actors: Patron, System
Preconditions: Patron has selected restaurant
Post-Conditions: Patron can view list of Menu Items

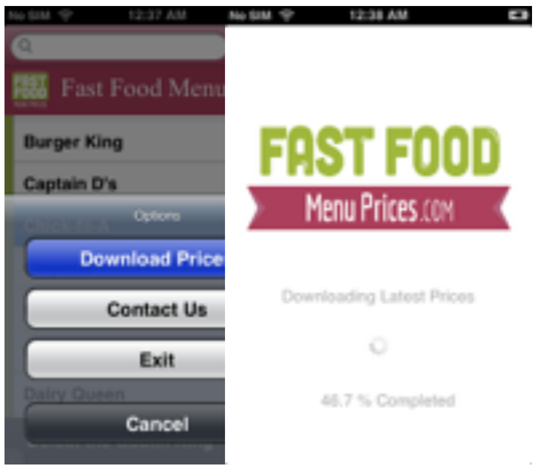
Successful Path:
1. System displays list of menu groups with items and prices in each

Exceptions:
Menu items did not load

3.2.1.5. Download New Restaurants

This use-case realizes the download function of the system. The application is open to the Restaurants Screen, and the Patron chooses options. After the Patron chooses options, the system displays the download option.

Actors: Patron, System, Device, API

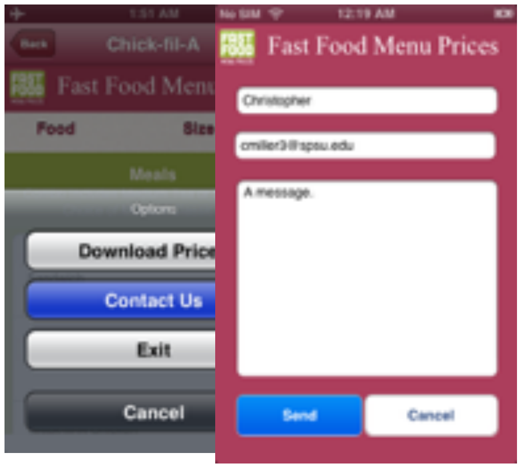


Preconditions: Patron has the system open to Restaurants Screen
Post-Conditions: Patron can view new/updated list of Restaurants

Successful Path:
1.Patron clicks options System icon on screen
2.Patron clicks Download Prices option on screen
3.System returns to Splash Screen
4.System retrieves list of restaurants from API endpoint
5.API Server queries database and creates JSON object
6.System updates Percentage of Restaurants Saved from JSON object
7. System returns to Restaurants Screen after all restaurants load
8. System displays list of restaurants

Exceptions:
Device cannot connect to network
API cannot generate JSON object

3.2.1.6. Contact FFMPrices Team



This use-case realizes the contact function of the system. The application is open, and the Patron chooses options, then the system displays the contact option.

Actors: Patron, System
Preconditions: Patron has the system installed on device and open
Post-Conditions: Patron submits correspondence to FFMPrices Team

Successful Path:
1. Patron selects options System icon on screen
2. Patron selects Contact Us option on screen
3. System opens Contact Screen
4. Patron enters name into input field
5. Patron enters email into input field
6. Patron enters message into input field
7. Patron sends correspondence

Alternate Path (Patron decides to cancel correspondence):
1. Patron cancels correspondence
2. System returns to previous screen

Exceptions:

Device cannot connect to network

3.3.Performance Requirements

Finding restaurants and individual menu items should take less than 3 seconds to complete. Finding menu items should take less than 5 seconds for menus containing 50 items or less, or less than 10 seconds for menus containing between 50 and 100 items.

3.4.Software System Attributes

The system is portable by its very nature: being a mobile application that is usable on a telephone and is available to be placed on a wide variety of devices.

The GUI will be organized in a manner that is both visually appealing and easy for the user to navigate, based upon the previous Android application's usability (but enhanced). The color coding of the application shall be used in the tinting feature of iOS UIKit components, which allows for the functions of the application to blend in color to the existing graphics.

The application will load quickly due to the data being preloaded, and will be very responsive during downloading of new restaurant data as well.