2/23/2014

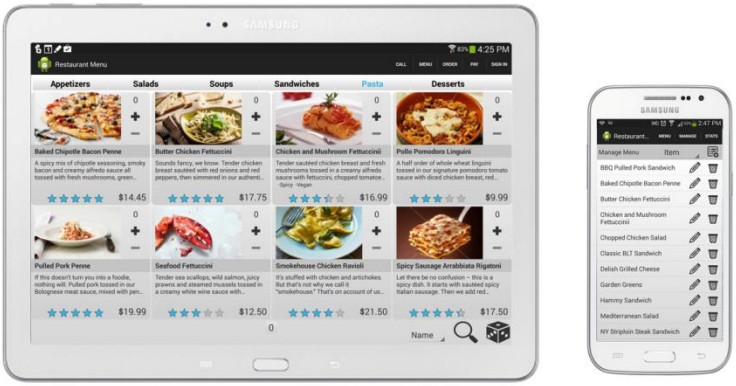
CARLETON UNIVERSITY

RESTAURANT MENU AND MANAGEMENT SYSTEM

COMP 4905 | Trevor Oberhammer

ABSTRACT

Many people have experienced going to a restaurant where the service is poor and there is a lack of attention from the wait staff. The paper menus can be flimsy, hard to navigate, and outdated. To leverage the growing mobile industry, the solution we have developed uses mobile devices in order to interact more with customers. This restaurant menu and management system will replace the paper waste, is more maintainable, and allows for greater customer engagement.



This project consists of the development of two of the mobile apps designed for this system. A particular focus for this project was on user interface design and how Wi-Fi Direct can be used with such a system. This project has provided good insight and experience in the development of user interfaces for mobile apps and the key design issues that must be considered. Wi-Fi Direct was successfully used to allow mobile devices to communicate and manage information between them, but some limitations discovered may not make it an ideal solution.

ACKNOWLEDGMENTS

I would like to thank my project supervisor, Dr. Tony White, for the support, feedback and encouragement he provided; which helped to remove much of the pressure I experienced while undertaking this project. I would also like to thank my project partner, Quyen Le, for her collaboration on this project and the support she was able to provide. Finally, I would like to thank the Android team for providing a comprehensive and helpful source of information and tutorials for their SDK and all those who contribute on sites such as stackoverflow.com who provide simple code examples that can help get things up and running much quicker.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Visiting a restaurant traditionally involves selecting a meal from a paper menu and being waited on by the restaurant's wait staff. A busy restaurant or inattentive staff can leave customers waiting to have their orders taken, drinks refilled or to receive their bill. If the restaurant is busy the customer is left occupying a table longer than they need leaving other customers waiting. Any unnecessary waiting can reduce customer satisfaction and ultimately result in lost business.

To reduce customer wait times, management must ensure sufficient staff is present during peak hours and that they are properly trained to provide excellent customer service. These staffing issues can lead to substantial costs for the business.

Paper menus are problematic. The restaurant may have a large number of menu items which can make the menu appear overwhelming to look through. As a result, customers may not see all the items they would have been interested in. When changes to the menu are required, such as price adjustments or item updates, the costs and environmental concerns associated with reprinting need to be considered.

Menu changes are often left to accumulate until enough are required to justify the costs of reprinting. Changes may be required frequently and a paper menu would quickly become outdated. Waiting until a reprint is done before implementing the changes in the restaurant may not be a sound business practice. Manually updating menus instead of reprinting can lead to inconsistencies and may make the restaurant appear cheap and low quality.

We have designed and built a restaurant menu and management system that provides an interactive tablet based menu that replaces the paper menu entirely and removes much of the need to be waited on by the restaurant's wait staff. This tablet based menu app also provides additional features designed to enhance the customer's overall experience. A separate management app allows the restaurant's management to quickly make changes to the menu and provide a snapshot of the restaurant at any given time. The restaurant menu and management system consists of the menu app, the management app, the server and a database. Other apps, intended to be used by the restaurant's kitchen and wait staff were not developed for this project. The development of this system was decomposed into two independent projects: the menu and management apps; and the server and database. This report covers the development of the menu and management apps.

MOTIVATION

The mobile market continues to grow each year replacing the demand for traditional desktop applications. This makes software development for mobile devices an interesting and attractive industry to work in. Motivation for this project stems from the desire to learn and gain experience in mobile app development as well as an interest in the design and development of distributed systems.

LEARNING OBJECTIVES

The development of the menu and management apps will provide a number of opportunities to learn and gain experience in many different areas of software development. The two main learning objectives for this project are to explore user interface design for mobile devices and Wi-Fi Direct. Upon completion of this project, I hope to have a better understanding of the key user interface design issues that must be considered when developing for mobile devices; some of the various sensor technologies provided by these mobile devices and how they can be used to enhance the UI experience; and how Wi-Fi Direct can be used to communicate, organize and share information between devices as well as any limitations that may exist. I will also expect to have developed a much better understanding of software development for the Android platform and experience working with RESTful Web Services, Google Protocol Buffers and building components of a distributed software system.

BACKGROUND

ANDROID DEVELOPMENT

Android is one of the world's most popular mobile platforms. Built on the contributions of the open-source Linux community as well as hundreds of hardware, software and carrier partners, Android provides a powerful development framework for building mobile applications. Using a single application model, Android allows the deployment of apps to users across a wide range of mobile devices. Android Developer Tools (ADT) is an Eclipse plugin that offers a full Java IDE with its own set of features to assist in the development of Android apps (Android, 2014). Android apps are written in the Java programming language. Android's open development platform allows developers to use third party tools in their apps opening the door to a wide range of options when considering development solutions. Android also provides an excellent testing environment with a

complete set of testing tools available with the ADT (Reuters, 2010). Both the menu and management apps for the restaurant management system are developed for Android Jelly Bean (4.1 - 4.3 APIs) devices.

JERSEY RESTFUL WEB SERVICES

Jersey is an open source, production quality framework for the development of RESTful Web Services in Java. Jersey provides support for JAX-RS APIs, serves as a JAX-RS Reference Implementation, provides its own API to extend the JAX-RS toolkit and offers additional features and utilities [Jersey, 2014].

GOOGLE PROTOCOL BUFFERS

Protocol buffers are a language-neutral, platform-neutral, extensible way of providing a flexible, efficient and automated mechanism for serializing structured data [Google, 2012]. The data is structured and defined in a .proto protocol buffer message file. The structure for each message is a logical record of information contained in a series of name-value pairs. The .proto file is compiled for the Java language, producing data access classes. These generated classes provide simple accessor fields as well as methods to serialize and parse the entire data structure. Protocol Buffers are used to serialize data sent between the various devices in the menu management system.

WI-FI DIRECT

Wi-Fi CERTIFIED Wi-Fi Direct is a certification mark for devices that support a technology that enables it to connect with each other directly without the need to connect to a traditional network hotspot. Wi-Fi-Direct connections are protected by WPA2 security technology and allow one-to-one connections or a group of devices to connect simultaneously with one another (Wi-Fi Alliance, 2014).

Android's Wi-Fi P2P framework complies with the Wi-Fi Direct certification program and allows Android 4.0 or later devices with the appropriate hardware to connect to other devices that are Wi-Fi Direct capable. The Android Wi-Fi P2P framework is used to allow menu app devices to connect to other menu app devices at the same table in the restaurant to provide the ability to share information and manage orders and bill payments.

USER INTERFACE DEVELOPMENT FOR MOBILE

Developing intuitive and easy to use apps for mobile devices can be a challenging undertaking. The Android platform is used on devices that vary in screen size, density, and locale and apps need to be able to support these variations so that the quality of the user experience provided by the app is stable across all supported devices. The very nature of mobile apps has a great effect on UI design. The screen size of mobile devices are generally much smaller than the typical screen size users are familiar with for desktop applications. Information must be condensed and the UI simplified so that the user experience doesn't deteriorate. Mobile devices also introduce a number of features and capabilities that are not available or are not commonly seen on desktop machines. Various sensors, such as an accelerometer, GPS, gyroscope and touch screens are standard with mobile devices. These sensors open a world of possibilities in how users are able to interact with an app in many ways not possible with desktop machines.

DESIGN

MENU APP OVERVIEW

The menu app is intended to be used by the restaurant's customers in place of a paper menu. Each customer at the table can be provided with a device running the menu app. These digital menus can be assigned to the same table to provide a logical group of devices that will be able to communicate with one another. For the scope of this project, the menu app is designed to be used on 10" tablets only and will not be supported on smaller screen sizes. A tablet with a large screen size was chosen to provide a better user experience. The actual device used for this project was a 10" Samsung Galaxy Note 2014 Edition.

The menu app includes a number of features intended to simplify and improve the customer's dining experience. Basic browsing of the restaurant's menu will be provided to replace the paper menu's functionality. Customers will also be able to search and sort the menu to quickly find and identify menu items of interest. To further enhance the customer's experience, provide greater convenience and service, and to reduce wait times, customers will be able to order directly from the menu app when they are ready to do so. All menu devices at the same table will be able to manage their individual orders and/or combine their orders to be sent at the same time. Bill management will also be provided, allowing customers at the same table to decide how they would like to split and combine the payment of their orders prior to requesting the bill. Smaller features designed to

personalize the experience will also be developed. These include support for a customer loyalty program, social media features, and a meal generator.

MANAGEMENT APP OVERVIEW

The management app is intended to be used by the restaurant's management staff and is designed for use on a Smartphone or a tablet device. To reduce the time spent supporting all the various screen sizes that Android devices are available in, only the 10" Samsung Galaxy Note 2014 Edition and the Samsung Galaxy S3 Smartphone was used for the development and testing of this app for the project. The management app will provide users with the ability to change the menu while the system is running. Elements of the menu, such as menu items, categories and item options can be added, modified and removed; with changes to the menu picked up through periodic updates by other running menu and management devices.

The management app will provide the same browsing, search and sorting functionality that the menu app provides. This will allow management to see the current look of the menu as users of the menu app are able to see. The management app will also provide the simple tracking of key business information such as the restaurant's revenue history, item popularity and customer traffic information. Requirements and detailed use case information for both the menu and management apps can be found in Appendix A.

SYSTEM DESIGN

The restaurant menu and management system is composed of several different components. The two components developed for this project are the menu app and management app. A REST architecture style is used to provide communication between the app components and the system's server component using the Jersey RESTful Web Services framework. Device-to-device communication between menu apps is accomplished through Wi-Fi Direct. The server connects to the database to retrieve and store the restaurant's menu, customer and business data (See Figure 1).

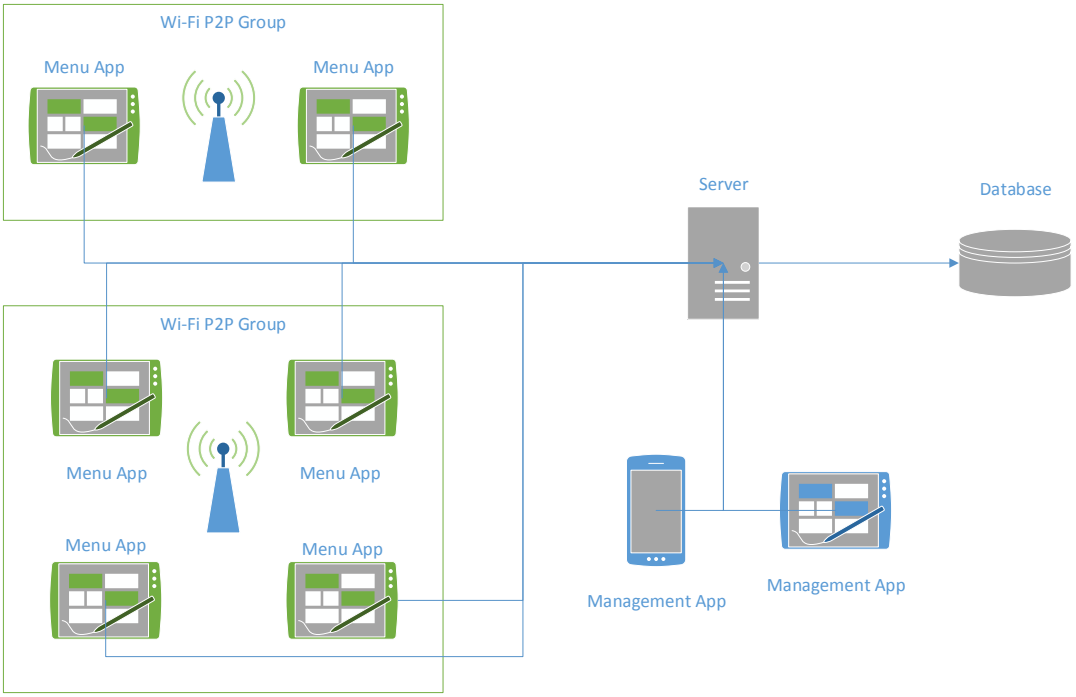


FIGURE 1- SYSTEM NETWORK DIAGRAM

While the menu and management apps are two different and independent programs, they both offer some common core functionality such as the ability to browse and search through the restaurant menu and how they communicate with the system's server. Both app are developed for the Android platform. These common features allowed for much of these two apps to be developed using the same underlying frameworks.

FRAMEWORKS

The Android framework is what the menu and management apps are primarily built on. The Jersey framework is used for RESTful communication between system components, however; use of the Jersey API was not required by the menu and management apps as the Android framework provided built-in alternatives. The Protocol Buffer framework is used to serialize data sent between the system components. The apps are event-driven in nature. User interaction creates events that trigger changes in the system. Events that require communication with the system's server or other menu app devices are managed by a framework using the Pipe and Filter and Reactor patterns. This framework was developed for this project and is used by both the menu and management apps. A class diagram of this framework is shown in Figure 2.

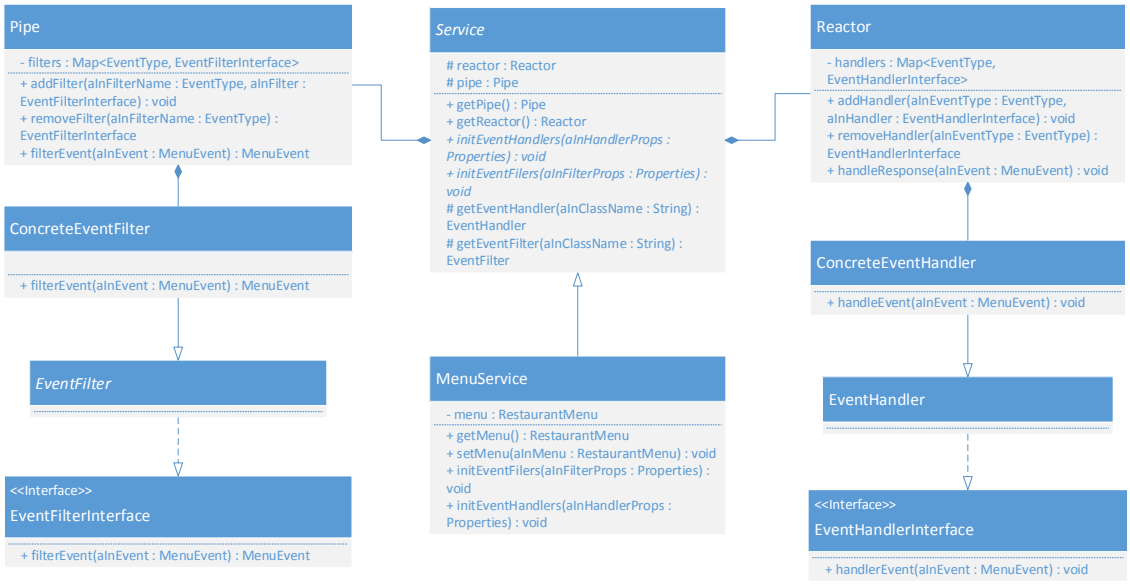


FIGURE 2 - CLASS DIAGRAM OF EVENT FRAMEWORK

REACTOR: The reactor handles events in the system. Event handlers are registered with the reactor at runtime through the use of a configuration file. Each handler is designed to handle a specific type of event. The handler processes event data and makes the necessary system changes. The reactor also provides a way in which asynchronous incoming events can be processed by the handlers in a synchronized manner. This helps reduce potential concurrency issues.

PIPE AND FILTER: The Pipe and Filter pattern provides a way of filtering events into a serialized form to be sent across the network and to filter serialized messages from the network back into events. Event filters are registered with the pipe at runtime through the use of a configuration file. Each filter is designed to filter a specific type of event either into its serialized form or back again. Filters serialize event specific data based on the protocol being supported by the system. For this project, Protocol Buffers are used to represent and structure the data being serialized. The Pipe and Filter provides a modular way of serializing data that can easily be configured to handle different protocols should the system requirements change without affecting the rest of the system.

USER INTERFACE DESIGN

MENU APP: The target users of the menu app will be primarily the restaurant's customers. The app is preloaded onto the device and is running and ready to go when the customers sit down at their table. A number of factors played an influential role in how the user interface was designed for the menu app.

DEVICE OWNERSHIP: The restaurant's customer is not using their own mobile device and is required to use the Android based tablet provided by the restaurant. Customers may not be familiar with the device and Android based apps. Android encourages developers to use certain features of their UI framework to build a UI that is consistent with the look and feel of other Android apps. A consistent look and feel between different apps is beneficial in that new users of an app will immediately feel more comfortable using the app for the first time and will require less time to learn how to use it. However, because not all users of the menu app will be familiar with Android some of these Android specific UI design choices may not be best suited for this app. This also applies to the Back, Home, and Options buttons provided on Samsung devices. These buttons are located outside of the main screen area. The Home button is a physical button that is visible to the user. The Back and Options buttons flank the Home button on either side, but are touch based and invisible to the user. While the Home and Options button will not be needed by the restaurant's customers, the back button provides a built in way for navigating back, much like in a web browser. Users who are not familiar with a Samsung device may not know these buttons exists and could accidently touch them without realizing it, triggering unwanted actions that would lead to confusion. Thus the use of and reliance of this button likely should not be considered.

APP USAGE: Users of the menu app are likely first time users; they are being asked to use the app and may not ever use the app again. This contrasts with a typical mobile app user whom willingly downloads and installs the app and has enough interest and motivation to at least try to learn how to use the app over a number of repeated uses. While all UI design should strive to be intuitive, easy to learn and use, the fact that users may only ever use the menu app once and are being asked to use it makes this goal a much more crucial one. The menu app’s user interface needs to be designed to be intuitive and easy to learn so first time users of the app are able to quickly figure out how to use the app without any or little assistance. This is especially important as the menu app is intended to provide much of the service needed to successfully order and pay for a meal at the restaurant.

MANAGEMENT APP: Target users for the management app will be the restaurant's management staff. The app can be loaded onto a tablet device or a Smartphone. This could be a device provided by the restaurant which is dedicated to the usage of the management app or a personal device belonging to the user. The same concerns as with the menu app don't apply for the management app. Users are expected to use the app frequently and will be able to learn how to use to app over

time. Users are also expected to be motivated to learn how to use the app as it will be a vital tool in operating the business.

SCREEN SPACE: Mobile devices inherently have smaller screens. Less content can be displayed on the screen at any given time. Making the most of the available screen size is crucial to a successful UI design. With fewer visible options at any given time, users are forced to rely on their short-term memory to build an understanding of the content being displayed. Less room is available for advanced features that may require a complex layout (Nielson and Budiu, 2013). If the UI is to support devices of different screen sizes and orientation, a UI that works well on one device may not be suited for another.

CHROME: Chrome in UI design are the visual design elements that provide users information about the screen's content or the commands to operate on the content. Chrome can provide a fixed set of commands that are always visible and available to the user and promotes consistency and standards in the UI. This facilitates learnability and can make users feel more in control (Nielson and Budiu, 2013). Chrome for the menu and management app will consist of the navigation bar at the top of the screen that will allow for quick navigation to the major UI components. This navigation bar will always be present and provide a steady set of options the user will be able to select at any time. Navigation bars are commonly found on website pages and is a UI element that most new users will be familiar with. Chrome specific to individual components that will not always be visible will be kept at a minimum to preserve screen space.

VIRTUAL KEYBOARD: Typing on a virtual keyboard is a headache for most. Error prone, users must visually attend to the keyboard as they type and many hunt-and-peck rather than touch type which can slow things down (Nielson and Budiu, 2013). To reduce the amount of typing required, only essential information will be requested from the user and form values will be pre-populated if the information is already available. Input controls provided by the Android framework that can be used as an alternative to the virtual keyboard will also be considered.

DOWNLOAD TIMES: Mobile apps that rely on information that is to be sent across a network can suffer in usability if the connection is too slow or users have to wait for large files to download before they can continue using the app. Communication with the system's server is crucial to the functionality and usability of both the menu and management apps. The menu and management system uses its own dedicated server and will have relatively few clients to that of a typical web server connected at any time, so slow and unreliable connections should not be an issue. To ensure

scalability, measures will be taken to reduce the number of interactions required between the menu and management apps with the system's server as well as reduce the amount of data sent between these components.

WI-FI-DIRECT

Wi-Fi-Direct was used to allow menu devices located at the same table in the restaurant to communicate with one another. The three features supported by Wi-Fi Direct that were developed during this project were menu item sharing, combined orders, and bill management. Menu app devices will be assigned to a unique table id and each device will be assigned its own unique menu id.

Network communication on wireless devices has an impact on battery life and Wi-Fi Direct is no exception. Ideally Wi-Fi Direct communication would only be established between devices when the devices were in use and not sitting idle, however a limitation with Android's Wi-Fi P2P framework is that user permission is required for a connection to be made between devices. This would require the customer to manually accept P2P connections when the device became active which is not ideal. To avoid burdening the customer with this issue, Wi-Fi Direct communication will be configured and established between devices by the restaurant staff prior to being used by the restaurant's customers and will remain connected throughout the day while the restaurant is open for business. This comes at an obvious cost to battery life; however, the data sent between devices will be minimal in size and will only occur when the features supported by Wi-Fi Direct are in use.

When a Wi-Fi P2P connection is established between two devices, one device is randomly designated as the group owner and the other the client or peer device. Any additional devices that connect to one of these two devices are automatically designated as another peer device in the group and will be directed to connect to the existing group owner directly. The Wi-Fi P2P group configuration will be similar to a client-server model in which the group owner will act as the server and all connected peer devices will act as clients; however, because the group owner device is randomly chosen and supports the same features as the client devices, it will behave as somewhat of a hybrid client/server in which it will receive requests from client devices and handle them much like a regular server would, but will also be able to send its own requests to client devices. TCP socket connections will be created between the group owner and each peer device using the Acceptor-Connector pattern.

Figure 3 demonstrates how four menu app devices would be configured in a Wi-Fi P2P group. Device B has been designated as group owner and devices A, C and D are the peer devices. If A wants to send a request to device B and D, the request would be sent to B, whom would handle the request and redirect the same request to D. If C wanted to send a request to A, the request would still go to B first, whom would then simply redirect the request to A. Now if B wanted to send a request to C, being the group owner, it simply sends the request directly to C.

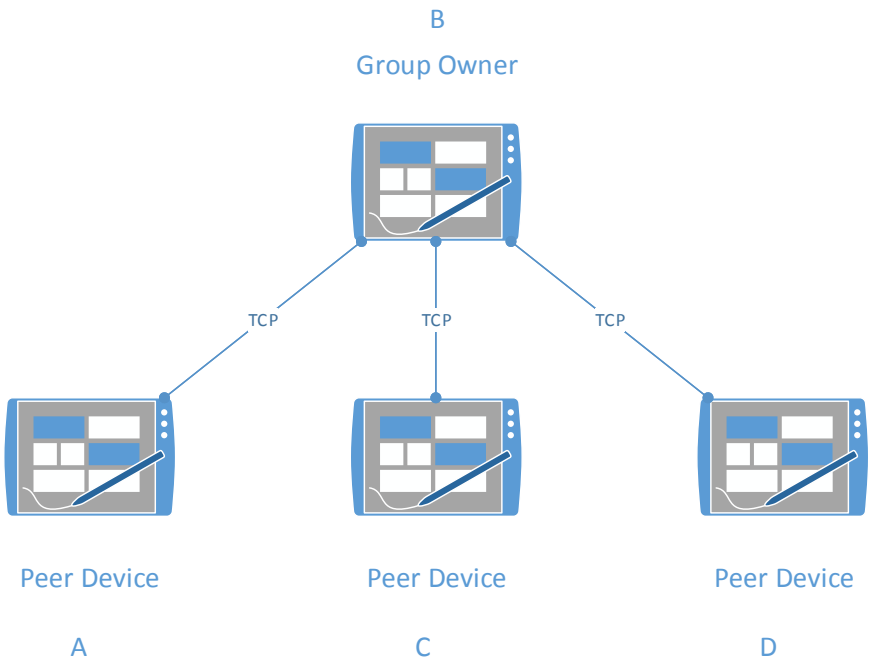


FIGURE 3 - WI-FI P2P NETWORK

IMPLEMENTATION

Many of the features proposed for the menu and management apps were completed for this project. Basic browsing of the restaurant's menu, including the ability to search and sort was implemented for both the menu and management apps. Additional features for the menu app that were implemented include the random meal generator, menu item sharing, ordering and bill management features. For the management app, additional features included the management of the restaurant menu, including the ability to add, edit and remove menu elements.

The main learning objectives of this project was to gain a better understanding about UI design for mobile devices and Wi-Fi Direct so this section of the report will focus on the main features that provided the most insight to these learning objectives. These include menu browsing, menu item

sharing, ordering and bill management. A general discussion of how the menu and management apps were organized and configured as Android projects will also be provided.

ANDROID PROJECTS

The menu and management apps are two separate Android projects, named Menu and Management respectively. However, because of the core functionality they support as described in the design overview of this report, a third project, MenuManagement, was created to allow the Menu and Management projects to share common code and resources. This project is another Android project which has been declared as a library project. The Menu and Management projects reference the MenuManagement library project and include the MenuManagement's compiled sources into their .apk files at build time (Android, 2014).

PACKAGES

Source code for the Menu, Management and MenuManagement projects is grouped into packages that describe the type of class or a common functionality in the system. All packages across the three projects begin with com.tjhammer.honours, each project then has its unique identifier such as .menu, .management and .common. These are the base packages for the Menu, Management and MenuManagement projects respectively. Java classes that don't fall into any specific grouping are contained in the base package. Common packages across all projects include .activities, .adapters, .filters, .fragments, and .handlers. The .activities package contains the main activity for each project. The .adapters contains all adapter classes used with Android's ListView layouts. The .filters package contains all request and response filters used by the Pipe for serializing and deserializing messages sent between the menu and management system components. The package .fragments contains all the Fragment classes used in the app and the .handlers package contains all the event handlers the Reactor uses to dispatch events. Generated Google Protocol Buffer classes are contained in the com.tjhammer.honours.rest.menu package and Wi-Fi Direct specific source files are in the com.tjhammer.honours.menu.p2p package.

LAYOUT RESOURCES

Static UI layouts for Android apps are commonly defined in XML files and are stored in the project's res/ directory. Separate directories for each screen size/orientation combination supported by the app was created. The directory res/layout-sw720dp-land uses a standard naming convention

recognized by the Android framework to load layouts designed for screen sizes that are larger than 720dp in width. This directory contains UI layouts designed for the menu or management app while used on a tablet in the landscape orientation. The res/layout-sw600dp directory is designated for UI layouts designed for tablet's in the portrait orientation and the res/layout directory is for UI layouts used by management app on a Smartphone in portrait orientation.

The Android framework detects the size of the screen and the orientation of the device at runtime and automatically applies the appropriate layouts. If a layout is not supplied in a particular folder, it is searched for in the next lowest supporting directory. This allows layouts that can be applied across all screen sizes and orientations to be kept in a single XML file in the lowest supporting directory. For example, if a layout is inflated while the tablet is in landscape orientation and the layout file is not contained in the res/layout-sw720dp-land directory, the res/layout-sw600dp directory is searched for the layout. If the layout is not found then the res/layout directory is searched. If the tablet was in portrait orientation, the res/layout-sw720dp-land directory would be ignored.

XML layout resource files were created for all static UI elements for both the menu and management apps. When possible, layout elements were defined in such a way to allow for general use across all screen sizes and orientations. This meant that width and height dimensions were not hardcoded to a specific dp value and were allowed to dynamically adjust to accommodate the element's content or the device's screen size. This ensured maximum support by varying screen sizes and orientations and made maintaining the layout easier. In many cases, particularly with fragment layouts, this was not possible and multiple layouts were required.

ACTIVITIES, FRAGMENTS AND THE ACTION BAR

Both the menu and management apps only use one main activity, called MenuActivity and ManagementActivity respectively. These activity classes inherit from MenuManageActivity which inherits from the Android Activity class directly. A simple class diagram is shown in Figure 4. Please note that not all fragment classes are shown in the diagram.

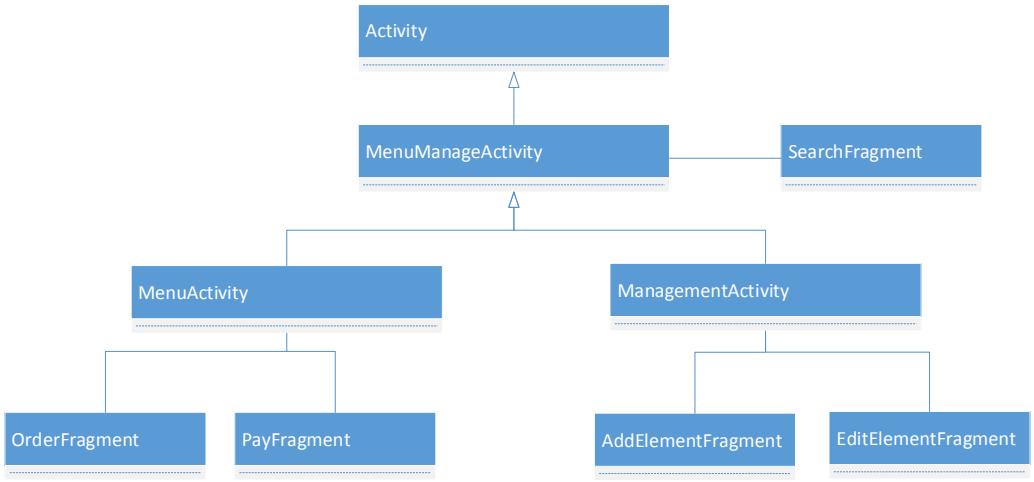


FIGURE 4 - CLASS DIAGRAM FOR ACTIVITY AND FRAGMENT RELATIONSHIP

The UI layout for the MenuActivity and ManagementActivity is made up of an Action Bar and a FrameLayout. The FrameLayout is just an invisible frame for fragment layouts, analogous to a picture frame. Fragments represent the UI for all the various UI components required by the apps. For example, when the MenuActivity first loads up it automatically places the fragment containing the menu layout into the FrameLayout. This layout contains all the layout elements needed to display the restaurant's menu. If the user selects the Pay button in the Action Bar, the menu fragment is replaced by the fragment containing the layout elements needed to display all the payment and bill components. Figure 5 shows how fragments are placed into the FrameLayout.

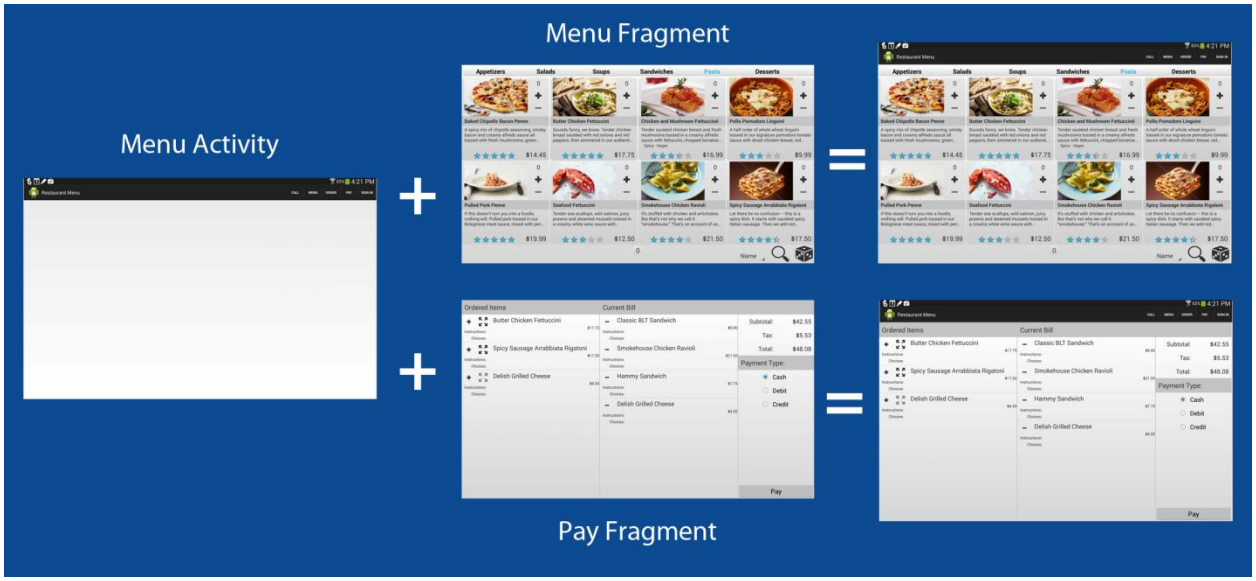


FIGURE 5 - ACTIVITY SWAPPING FRAGMENTS

A back stack is maintained of all fragments that have been previously loaded into the Activity. When replacing an existing fragment, the fragment being replaced can be pushed onto the back stack allowing the user to navigate back to this fragment by pressing the device's back button. For the menu and management apps, the fragments loaded by selecting a menu option on the Action Bar are not added to the back stack and are completely replaced by the newly loaded fragment. This prevents the user from unintentionally adding numerous instances of the same fragment to the back stack while navigating with the Action Bar. In the cases where navigation choices are limited to the current fragment, for example, viewing a specific item in the restaurant's menu is only accomplished by selecting the item from the list of menu items in the menu fragment, the current fragment is placed in the back stack so if the user presses the Back button they are returned to the menu fragment in the state they had previously left it.

The Action Bar is the navigation bar commonly seen on Android apps. The app's icon, name and all navigation and actions that need to be constantly available and visible are placed in the Action Bar. Actions that users should be able to perform regardless of where they are in the app were chosen to be in the Action Bar. For the menu app this included Call, Menu, Order, Pay and Sign In / Sign Out. For the management app, the options are Menu, Manage and Stats.

It was of particular importance for the menu app that the core features needed to successfully select, order and pay for a meal, ask for assistance and perform account actions were always visible and available so the user could easily find them. A small sacrifice was given to screen space as the Action Bar is always visible and therefore requires a set amount of space to be displayed; however, considering that most users would be new to the menu app, the benefit of simplifying navigation to core features vastly outweighed the small amount of screen real estate allotted to the Action Bar.

RESTAURANT MENU BROWSING FEATURE

OVERVIEW: The restaurant menu is the central feature for the menu app as almost all other features are based on the ability to browse the restaurant menu. The restaurant menu is the first fragment loaded when the menu app opens and should be the first thing that restaurant customers see when they begin to use the menu app. An option for navigating to the restaurant menu is provided in the Action Bar so customers can quickly return to it at any time. The management app also supports this feature providing management with the ability to view the restaurant menu as it is displayed on the menu app and to see additions and modifications they have made to the menu.

To make the menu and management system as flexible as possible in terms of the variation of menus supported, many of the common elements typically found in a restaurant menu were included. These elements include: items, categories, tags, options and choices. Items refer to the actual menu item being offered by the restaurant. Categories are provided as most restaurant menus employ some logical categorization of their menu items. There is a one-to-many relationship between categories and menu items. A category can be assigned to any number of menu items; however, a menu item can only be assigned to one category. Tags are labels that can be created and assigned to any number of menu items. An example of a tag could be "Spicy" or "Gluten Free". A many-to-many relationship exists between tags and menu items. Options are simply categories for choices. Many restaurants offer choices with their menu items, for example, a hamburger often comes with a choice of sides which could include fries, salad, onion rings, etc. In this example the option is "Sides" and the choices are "Fries", "Salad" and "Onion Rings". A many-to-many relationship exists between menu items and options; and options and choices. Tags, options and choices are optional and are not required in the creation of the restaurant menu. At least one category is needed as all menu items need to be assigned to a category for display purposes.

UI: The menu_fragment.xml defines the static layout of the restaurant menu. The MenuFragment class is an extension of the Android Fragment class. Restaurant menu items are displayed by category using a GridView layout. This layout is populated at runtime based on the menu information stored in the RestaurantMenu class. In landscape orientation, a 4x2 grid can fit on the screen; for portrait orientation, a 2x3 grid is supported. Additional menu items can be viewed by vertical scrolling. Swiping in a horizontal direction will cycle the menu to the next / previous category as will selecting a category. Categories are listed in a horizontal fashion directly above the menu items. If there are too many categories to fit within the screen width, the categories scroll horizontally so the user can simply swipe on the categories to scroll over to see more categories. To avoid crowding at the top of the screen a panel spanning the width of the screen was added below the menu item grid at the bottom of the screen. This panel contains the menu device's id and additional feature choices such as search, sort and the meal generator. Figure 6 shows the menu fragment in both portrait and landscape orientations.

Each menu item displayed in the menu is a condensed view of the item's information. This includes a thumbnail sized image of the item, the item name, an abbreviated version of the item's description, the item's price and any tags that the item has been assigned. The menu and management system uses a recommender system deployed on the system's server. As customers

provide ratings for the menu items they have tried, the system produces an overall rating for the item. This rating is displayed as a score out of 5, represented by the 5 stars shown for each item. A small panel on the right side of each menu item provides a quick way for customers to add and remove items from their order. A counter is displayed in the top-right corner of each item to inform the user as to how many instances of that particular item have been selected to order. This counter is automatically adjusted when the item has been ordered.

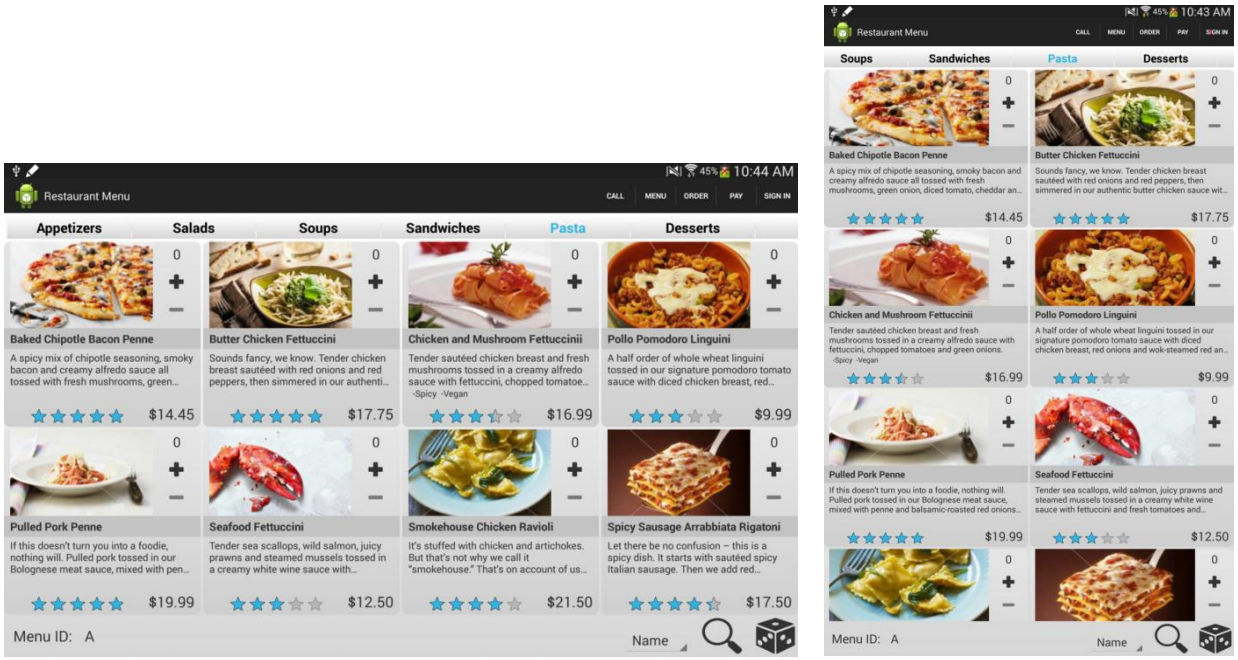


FIGURE 6 -MENU FRAGMENT IN BOTH ORIENTATIONS

Selecting a menu item will open the MenuItemFragment which provides a full view of the selected item. Attempting to add an item that contains options and choices that need to be selected before the item can be added to the order from the MenuFragment will also open the MenuItemFragment. All the previous information provided about the item in the MenuFragment are also shown in the MenuItemFragment with the exception of the current order counter, the add/remove button panel and item rating. A full description is provided and any options and their choices are listed. Choices are provided as RadioButtons as the user will only be limited to one choice from each option. Every item has a "Special Instructions" option in which the user can specify any special instructions they would like the kitchen to know when preparing the item. The user simply needs to add their instructions in the EditText view provided. A button panel is provided at the bottom of the screen allowing the user to return back to the MenuFragment, use the Share feature, or add the item to their current order. An example of the MenuItemFragment can be seen in Figure 8.

To improve user interaction while browsing the restaurant menu, all of the restaurant's menu information is downloaded from the system's server to the device when the menu app is first started and at periodic intervals. This interval is configurable through a configuration file. Periodic updates are used to pick up any changes to the menu that may have occurred. All of the menu items, categories, tags, option and choices are retrieved from the server then organized and stored in an instance of the RestaurantMenu class. While downloading the entire menu on each update may seem unnecessary due to the fact that restaurant menus are unlikely to change frequently and there will undoubtedly be redundant information being sent, the menu information consists of primitives and String objects and will be relatively small in size. Sending all the menu information each time also simplifies the effort needed to ensure that all menu app devices have the most up-to-date menu information. Each menu item is associated with an image name. This image name refers to an image file stored on the menu app device. A check is made each update to see if the image can be found on the device. If not, a separate request is sent to the system's server requesting the image file. This way an image only needs to be downloaded once from the server, reducing download time and improving the user's experience. The downloading of the menu information is performed on a background thread by the DownloadMenuTask.

DownloadMenuTask: The DownloadMenuTask is a subclass of Android's AsyncTask and is tasked with requesting the menu information from the system's server then performing the local check for each image file. The DownloadMenuTask requests and downloads each image file not found on the device from the server individually. The DownloadMenuTask is used for all menu updates including the initial update when the app is first opened. A boolean flag silentUpdate can be specified each time an instance of the DownloadMenuTask is instantiated. This flag is set to false for the initial menu download and true for all periodic updates after. When false, a progress dialog is shown notifying the user that the menu is being downloaded. The progress bar is also updated to display the image name of every image as it is downloaded. Once the download task is complete, the MenuFragment is opened to display the restaurant menu. The displaying of the progress bar and opening of the MenuFragment is not performed when the silentUpdate flag is set to true. This way the updating of the restaurant menu is performed without disrupting the user.

RestaurantMenu: The RestaurantMenu is a thread safe class that contains all restaurant menu information provided by the system's server and is used to supply menu information to the various UI components that need it. An Android SparseArray is created for each element type in the menu and another for mapping menu items to their category. The RestaurantMenu class provides accessor methods to retrieve and replace the menu information. The SparseArrays containing the menu information are not updated each time the menu is downloaded from the system's server, but are completely replaced. This was done to reduce the processing overhead that would have been needed to compare the old menu information with the new menu information and only update and/or remove the necessary element items. Since the entire menu is downloaded each time, completely replacing the old information was a simpler, cleaner and more robust implementation.

WI-FI DIRECT SETUP

To configure the menu app to use the Android Wi-Fi P2P framework a number of steps need to be taken. Request permissions need to be declared in the Android manifest file to allow the menu app to use the device's Wi-Fi hardware. A number of different objects are needed to manage and create Wi-Fi Direct connections with other devices. These are the WifiP2pManager, Channel and WifiP2pBroadcastReceiver. The WifiP2pManager is part of the android.net.wifi.p2p API and provides the API necessary for managing Wi-Fi P2P connections. The Channel connects the menu app to the Wi-Fi P2P framework and is obtained when the WifiP2pManager is initialized. The WifiP2pBroadcastReceiver is a simple extension of the Android BroadcastReceiver. It receives Wi-Fi Direct specific broadcasts that are filtered with an IntentFilter. The WifiP2pManager and the Channel object are created during the MenuActivity's onCreate() method and the WifiP2pBroadcastReceiver is instantiated during the onResume() method. The IntentFilter is registered with the WifiP2pBroadcastReceiver at this time as well and is unregistered during the MenuActivity's onPause() method. This ensures that the WifiP2pBroadcastReceiver will not receive intents when the menu app is paused reducing system overhead. The P2pTableService class was created to encapsulate all the necessary methods and objects required to establish a Wi-Fi P2P group between menu app devices and to manage these connections. The P2pTableService is instantiated during the MenuActivity's onCreate() method. Figure 7 shows the class diagram of the Wi-Fi P2P framework developed for the menu app.

FIGURE 7 - CLASS DIAGRAM OF WI-FI P2P FRAMEWORK

TABLE SETUP (CREATING / CONNECTING TO A WI-FI P2P GROUP)

The process of creating a Wi-Fi P2P group is a complicated process and this section will describe the process at a high level. An options menu provides the ability for users, the restaurant staff in this case, to configure the table id and menu id assigned to the device. The table id is used to determine which devices are to join which Wi-Fi P2P group, and the menu id is used to uniquely identify each device in the group. The option menu also contains a "Join Table" and "Leave Table" button. Selecting the "Join Table" button on a device will initiate a call to the P2pTableService's joinTable() method. Initially, two menu devices must initiate this call. Each device will first create a TableService that includes the table id assigned to the device and then broadcasts the service to all devices that may be listening. Each device will then attempt to connect to each other's service.

Once one of the devices (it doesn't matter which one) picks up the other's service, the service is first checked to verify that the table id provided matches the table id assigned to the device. If the table ids match then a connection with the service is created. Once a connection is established, one of the devices is randomly designated as the group owner and the other the peer device. The group owner will create a PeerAcceptor thread that is tasked with listening for and accepting connection requests on a ServerSocket. The peer device will create a GOConnector thread tasked with requesting a Socket connection with the group owner device. Once the socket connection between group owner and peer devices is established a TableMemberConnection is created on each device. The TableMemberConnection manages all further communication on the socket.

At this point, two devices have now connected and formed a Wi-Fi P2P group. Any additional devices that wish to connect to the group, need only to configure their table id to match and provide a unique menu id, then request to join the table by selecting the "Join Table" option in the options menu. Since a Wi-Fi P2P group has already been established, newly connected devices are automatically designated as peer devices and will create socket connections with the group owner device.

MANAGING TABLE MEMBER INFORMATION

It is important that each menu app device connected to a Wi-Fi P2P group is aware of what other menu app devices are connected to the same group. Each peer device that connects to the group owner will immediately send a JOIN_TABLE_REQUEST to the group owner. This request contains the menu id of the peer device. When the request is received by the group owner, the request is filtered into a MenuEvent that is dispatched to the Reactor. The handler for a JOIN_TABLE event will first check that the device is in fact the group owner, then will add an entry into the TableManager for the peer device. The TableManager on the group owner essentially keeps a master list of all the menu app devices connected to the group. Each peer device in the Wi-Fi P2P group is then sent a UPDATE_TABLE_REQUEST containing all the menu ids of the devices connected to the group. When the request is received on the peer device, the request is handled and the peer device's TableManager is updated with the list of menu ids.

All communication between menu app devices over Wi-Fi Direct is handled through the event framework discussed in the Design section of this report. Each request that is sent from a device is converted to a response event on the receiving device and is handled by an event handler in the Reactor.

TABLE SHUTDOWN (LEAVING A WI-FI P2P GROUP)

The Wi-Fi Direct connection is kept open until the "Leave Table" option in the options menu is selected. If the device wishing to leave the table is a peer device in the Wi-Fi P2P group then a LEAVE_TABLE_REQUEST is sent from the peer device to the group owner. The group owner will remove the device from the TableManager and then send another LEAVE_TABLE_REQUEST back to the peer device. The group owner device will also check to see if there are still devices connected to the table. If so, the group owner will broadcast an updated list of menu ids to all remaining peer devices in the list. If there are no remaining devices connected to the table, the group owner will initiate a call to the P2pTableService's leaveTable() method. When the peer device that initiated the request to leave the table receives the LEAVE_TABLE_REQUEST from the group owner, it too will call the P2pTableService's leaveTable() method. The leaveTable() method is responsible for disconnecting the device from the Wi-Fi P2P framework and closing resources.

A limitation discovered with Wi-Fi Direct is that the Wi-Fi P2P group is dependent on the group owner device. If the group owner device leaves the group, either by choice or if the connection was lost, the entire Wi-Fi P2P group would be disassembled. All peer devices will be disconnected and another Wi-Fi P2P group would need to be reestablished. In the case where the group owner device is leaving the group by choice, an orderly process is initiated in which the group owner first broadcasts a TABLE_CLOSE_REQUEST to all peer devices. Upon receiving this request, each peer device sends a LEAVE_TABLE_REQUEST back to the group owner triggering the same process as if the peer device initiated the action themselves. This allows all devices in the Wi-Fi P2P group to cleanly shutdown their P2P resources when the leaveTable() method is called.

MENU ITEM SHARING FEATURE

OVERVIEW: The menu item sharing feature was implemented for the menu app and is a Wi-Fi Direct only feature. If the menu app is not connected to a Wi-Fi P2P group then this feature is disabled. The menu item sharing feature allows users to share a single menu item with another device. The idea is that if the user, the restaurant customer in this case, were to discover a menu item they felt someone else at the table would be interested in, they could "send" the item to the other person's menu app device so they could immediately view it. This would save the customer from having to explain how to navigate to the particular item or physically hand their menu app device over to the other person.

DESCRIPTION OF UI FLOW: While viewing a particular menu item the "Share" button will be enabled if the device is connected to a Wi-Fi P2P group as shown in Figure 8. Selecting this option will open a fragment listing the menu ids of all the other menu app devices connected to the same group. This is a checkbox list and any number of devices from the list can be selected. Once the user has chosen which devices they would like to share the menu item with they simply select the "Share" button once again to send out the request. The user is returned back to the menu item fragment and a Toast message is displayed informing them that the share item request has been sent.

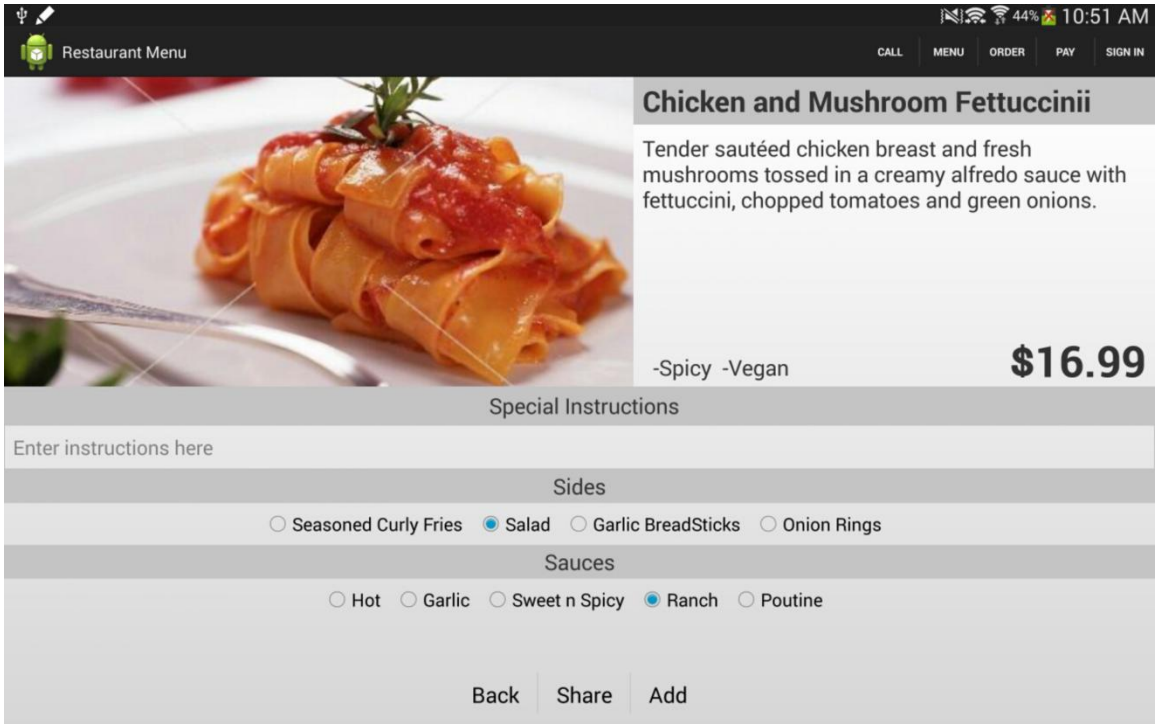


FIGURE 8 - MENU ITEM FRAGMENT WITH SHARE FEATURE ENABLED

The menu app devices that have been selected to receive the share request will present the user with a dialog box explaining that a menu item has been shared with them and asking if they would like to view the item. If the user selects "Cancel" to deny the share request, the dialog disappears and the user is returned to their current activity. If the user chooses to accept the share request, the menu item fragment is immediately opened displaying the menu item.

WI-FI DIRECT IMPLEMENTATION: This was the simplest of the Wi-Fi Direct features to implement. The TableManager is used to provide a list of the menu ids associated with the menu app devices connected to the Wi-Fi P2P group, which is displayed for the user to select from. Once the user has selected the devices to send the share request to, a MENU_SHARE_REQUEST is sent to the selected devices. If the device initiating the share request is the group owner of the Wi-Fi P2P group, then the request is simply sent directly to each device. If the device is a peer device in the group, then the request is first sent to the group owner. Each share request contains a list of the menu ids of the devices that the request is intended for and the menu item id of the item in the restaurant menu that is being shared. The list of menu ids is used to verify that the incoming share request is in fact intended for the device that receives it and is also used by the group owner to redirect the request to all other devices the request is intended for. Upon receiving and verifying the share request was intended for the device, a dialog box is opened allowing the user to decide if they would like to see the menu item being shared. If the user confirms, the menu item id contained in the request is used to retrieve the menu item from the RestaurantMenu and the item is displayed for the user to see.

ORDERING FEATURE

OVERVIEW: This feature was implemented specifically for the menu app and provides users with the ability to order menu items directly from the menu app device. This eliminates the need for the restaurant's wait staff to physically come to the table to take the order. This is considered to be a vital feature of the menu app. This feature is intended to work with both individual menu app devices and devices that are connected to the same Wi-Fi P2P group. Devices connected to a Wi-Fi P2P group have the option to combine their individual orders together to be sent in at the same time. This was considered an important detail as it implicitly signals to the restaurant's kitchen staff that the items on the order should be prepared and ready to go at approximately the same time. Orders can also be sent out individually even while a combined order is being created. This way, for example, if customer A has combined their entree selection with customer B and C, but C is still deciding what they would like and is not ready to send the order in, customer A can still submit an order for drinks or an appetizer in the meantime. An order history is also kept on each menu app device so customers can review the orders they have sent out. Both individual and combined orders are recorded.

DESCRIPTION OF UI FLOW: Once a user has decided on a menu item they would like to order they can select to add the item to their current order as described in the menu browsing feature section of this report. As many items can be added to their order as they would like. Once they are ready to order the user simply selects the "Order" button in the Action Bar. This will display the order fragment. All of the menu items that have been selected to order are individually listed in a ListView which will be referred to as the selected items list (left side of the screen in Figure 9). Each listing displays the menu item's name, any special instructions that were specified and any choices that were selected. Each listing also contains a checkbox and an icon of a trash bin. The trash bin is provided so the user can remove the item from the list if they decide they no longer wish to order the item.

If the user would like to submit an individual order, they simply select each item they wish to order by checking the checkbox for that item then select the "Order" button. The selected items are removed from the list and a Toast message is displayed on the screen notifying the user that the order has been submitted.

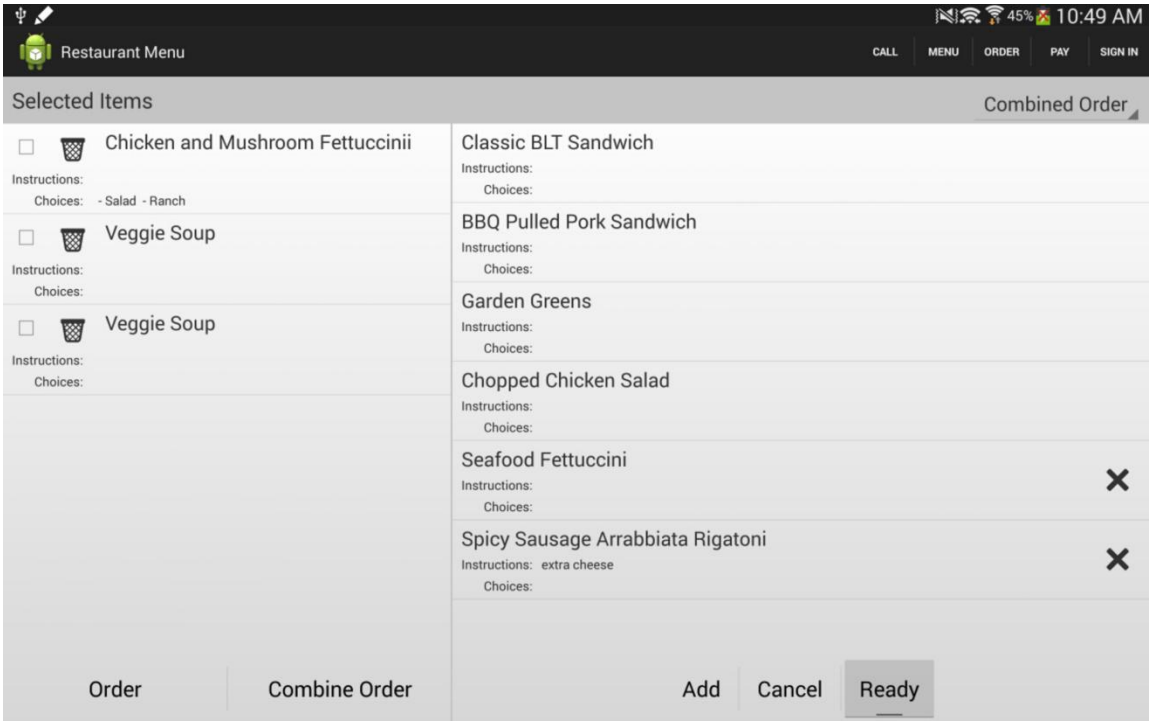


FIGURE 9 - ORDER FRAGMENT

If the user would like to combine their order with others at the same table and a combined order has not yet been created, the user simply selects the items they would like to add to the combined order, the same way as they would for an individual order, then press the "Combine Order" button. This will result in a fragment much like the one described in the share menu item feature that lists all the devices that are connected to the Wi-Fi P2P group. The user selects which devices they would like to combine their order with and selects the "Combine" button. The user is then returned back to the order fragment and can now see the items they have selected to combine in the Combined Order panel of the order fragment. Menu app devices that were selected for the combined order will have the items in their order fragment as well.

To reduce the number of steps required for the users and simplify the process, it is assumed that the customers have agreed amongst themselves as to who will be combining orders. This avoids the need to have confirmation dialogs popping up on everyone's screens asking if they want to confirm the combine order requests. Naturally, a cancel option is available in case a user decides they no longer wish to participate in the combined order.

Once a combined order has been created, additional options will become available in the order fragment. These are: Add, Cancel and Ready. When any of the users participating in the combined order wish to add items, they simply check the items from the selected items list and press the "Add" button. The chosen items will be removed from the list and added to the combined order. All changes made to the combined order will be immediately reflected on all devices. Items that a user has added to the combined order will have the option to be removed by the same user. This is shown as an "X". For example, if customer A adds item 1 and 2 and customer B adds item 3, then on customer A's menu app device, items 1 and 2 will display the "X" button, and item 3 will not. The opposite is true for customer B. Removed items are returned back to the list of selected items. Cancelling a combined order will remove all of the items the user has added to the combined order and place them back in the selected items list and they will no longer be included in the combined order.

When the user is ready to submit their combined order, they simply press the "Ready" button and no further action is required by them. The combined order will only be submitted once all customers participating in the combined order have selected the "Ready" button. They do not all have to press it at the same time. For example, customer A and B would like to combine their orders. If customer A knows what they want to order they can create and add their items to the combined order then select "Ready" while customer B is still browsing the menu. Eventually,

customer B decides what they would like and adds their items to the combined order. Once they select the "Ready" button the order will be submitted automatically and a Toast message notifying the user the order has been sent will be displayed.

WI-FI DIRECT IMPLEMENTATION: This feature, specifically the combined order capability, was the most complicated of the Wi-Fi Direct features to implement. A subclass of the MenuService class was created called OrderAndPayService to provide the data structures and methods needed to manage order and billing information. Objects such as OrderItem, MenuOrder and CombinedOrder were also created. The biggest challenge was synchronizing the combined order information between the menu app devices that were participating in the order. This was aggravated by a number of factors.

1. Not all devices in the Wi-Fi P2P group were necessarily participating in the combined order.
2. While a menu app device could only participate in one combined order at any given time, different combined orders could potentially be built simultaneously between different devices.
3. Users may wish to join an existing combined order after the order had already been created between its initial participants.

These factors meant that a single combined order could not be simply shared and synchronized between all devices in the same Wi-Fi P2P group, but that combined order information would have to be shared and synchronized only between the participating devices. Doing so added a layer of complexity to the implementation, but solved all the potential problems.

A COMBINE_ORDER_REQUEST is sent out from the menu app device after the user first initiates the creation of a combined order. This request contains the order item information for each item initially selected to add to the combined order, including the menu item id, any special instructions and selected item choices; and the menu ids of the menu app devices in the Wi-Fi P2P group they would like to combine the order with, including the device's own menu id. The request is distributed throughout the group in the same fashion as discussed in the menu item share feature, with one difference. If the request was sent by a peer device, the group owner will check that it does not redirect the request back to the sender. Since the MENU_SHARE_REQUEST sent for the menu item share feature did not contain the sender's menu id this was not an issue. Upon receiving the request, each device will retrieve the CombinedOrder object from the OrderAndPayService and adds all the menu ids and order item information contained in the request to their existing orders.

Each device then sends an ORDER_UPDATE_REQUEST to all menu app devices participating in the combined order. This request contains all the menu ids and order item information in the combined order as well as a count of how many devices have signaled that they are ready to submit the order. This counter is synchronized between participating devices and is used to determine when to submit the combined order. Upon receiving the ORDER_UPDATE_REQUEST, any existing combined ordering information is replaced by the information contained in the ORDER_UPDATE_REQUEST. The device that initially sent out the COMBINE_ORDER_REQUEST removes each item that was initially chosen to add to the combined order from the selected items list in the order fragment at this time as well.

Further updates, such as adding or removing an item, to the combined order at this point by any device are a little more straightforward. A request is sent out to all menu app devices participating in the combined order containing the order item information of the items to be added or removed. Upon receiving the request, the items are simply added or removed from the combined order.

The "Ready" button is a ToggleButton and has an on / off state. Each time this button is pressed, the local counter representing the number of devices that are ready to submit the combined order is adjusted. The counter is incremented once when the ToggleButton's state is set to on, and decremented once when its state is set to off. A check is made to determine if the counter is equal to the number of menu app devices participating in the combined order. If it is, then all devices have signaled they are ready and the combined order is submitted. If it isn't, then a READY_TO_ORDER_REQUEST is sent to all menu app devices participating in the combined order. A boolean value representing the state in which the "Ready" button was set to is packaged with the request. Upon receiving the request, menu app devices either increment or decrement their local counter depending on the boolean value sent. This ensures the counter is synchronized between all participating devices and that only when the last device is ready, the order is sent out.

Once the combined order is sent, a COMBINED_ORDER_SUBMITTED_REQUEST is then sent to all participating devices by the device that submitted the order. This is only done once a confirmation from the system's server is received verifying the order was successful. An order id is returned in the confirmation and is packaged with request. Upon receiving this request, each menu app device makes a call to the OrderAndPayService's processCombinedOrder(int aInOrderId) method. This methods resets the combined order information and creates an entry in the order history.

BILL MANAGEMENT FEATURE

OVERVIEW: This feature provides customers with the ability to manage the items that were ordered from their table in respect to creating and requesting their bills. All items, including both items ordered directly from the device as well as those ordered by other devices connected to the same table are visible. Customers can select the items that they wish to pay for and add them to their bill. Each menu app device has a bill that can be created. This provides maximum freedom in allowing the customers at a table to determine who is paying for what. For example, if a family is dining at the restaurant, they may all make their own item selections and orders on their individual menu app devices, but when it comes time to pay for the meal, it is likely that only one person will pay for all the items that were ordered and can do so from one menu app device. In another case, it may be a group of friends or acquaintances dining out together and each person will pay for the items they have ordered. In any case, the flexibility to choose the items to be added to their bill is provided. An option to split the cost of an item is also provided. This allows customers to pay for a portion of an item that was ordered and was likely shared amongst the group.

As items are added or removed from the bill, bill totals are calculated so the customer can see the current cost of their bill. Once they are ready to pay they can select their preferred payment method, such as cash, debit or credit, then submit their bill.

DESCRIPTION OF UI FLOW: When a customer would like to pay for their meal they press the "Pay" button in the Action Bar. This will open the pay fragment displaying all the items that were ordered at the table (see Figure 10). This will be referred to as the ordered items list. Each item contains the menu item's name, any special instructions or choices that were specified when the item was ordered and the menu item's price. Two buttons are provided for each item. An add button, displayed as a "+" and a split item button. The user can add each item they would like to pay for to their bill by pressing the item's "+" button or swiping on the item in the direction of the bill. Items that have been added to the bill display the same information, however, the "+" and split item button is replaced by a "-" button. Pressing this button will remove the item from the bill and return it back to the ordered items list.

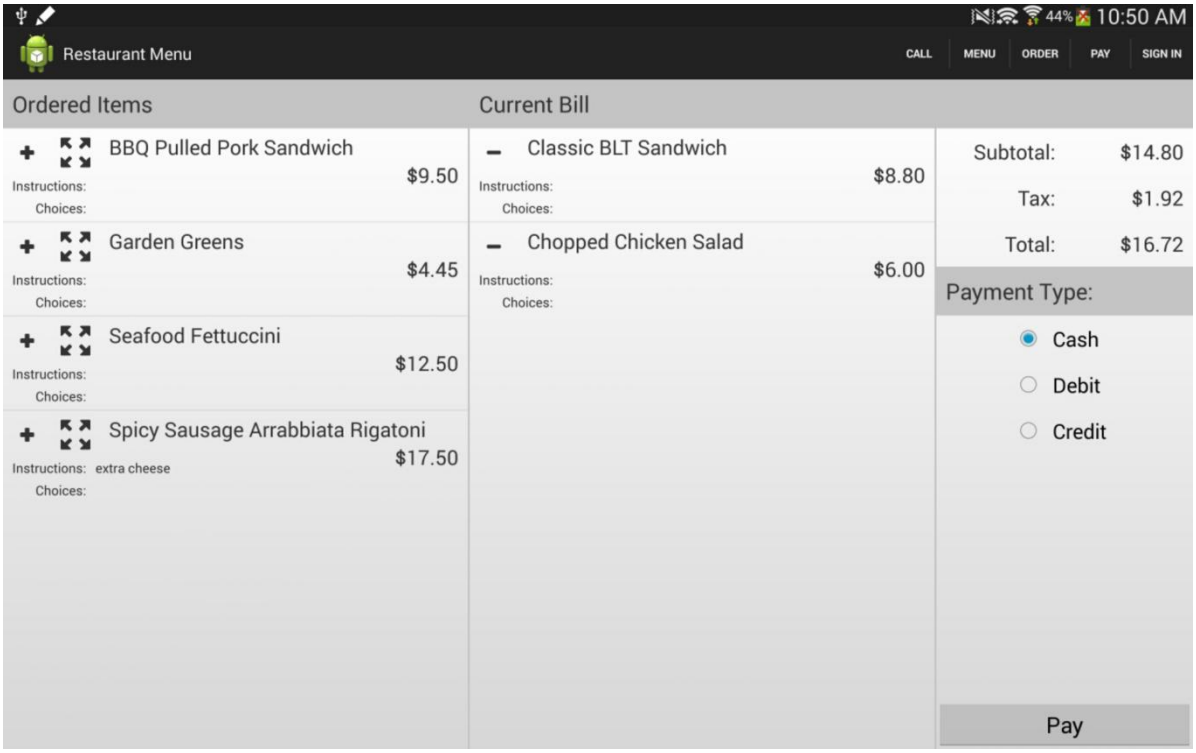


FIGURE 10- PAY FRAGMENT WITH ORDERED ITEMS ADDED TO BILL

If the user selects the split item button, a dialog will be shown providing the user with a list of integers representing the number of times they would like to split the item. The minimum value is 2 and the maximum value will be set to the number of menu app devices that are connected to the same Wi-Fi P2P group. Thus, if 4 menu app devices are connected to a table, any one item can only be split into a maximum of 4 equal parts. Once an item has been split it cannot be split again, nor can it be recombined. This simplifies the process of item splitting for the user. Each portion of a split item is displayed as its own entry in the ordered items list. The values are identical and the item's original price is adjusted to show its split value allowing users to easily see how much each portion will cost (see Figure 11).

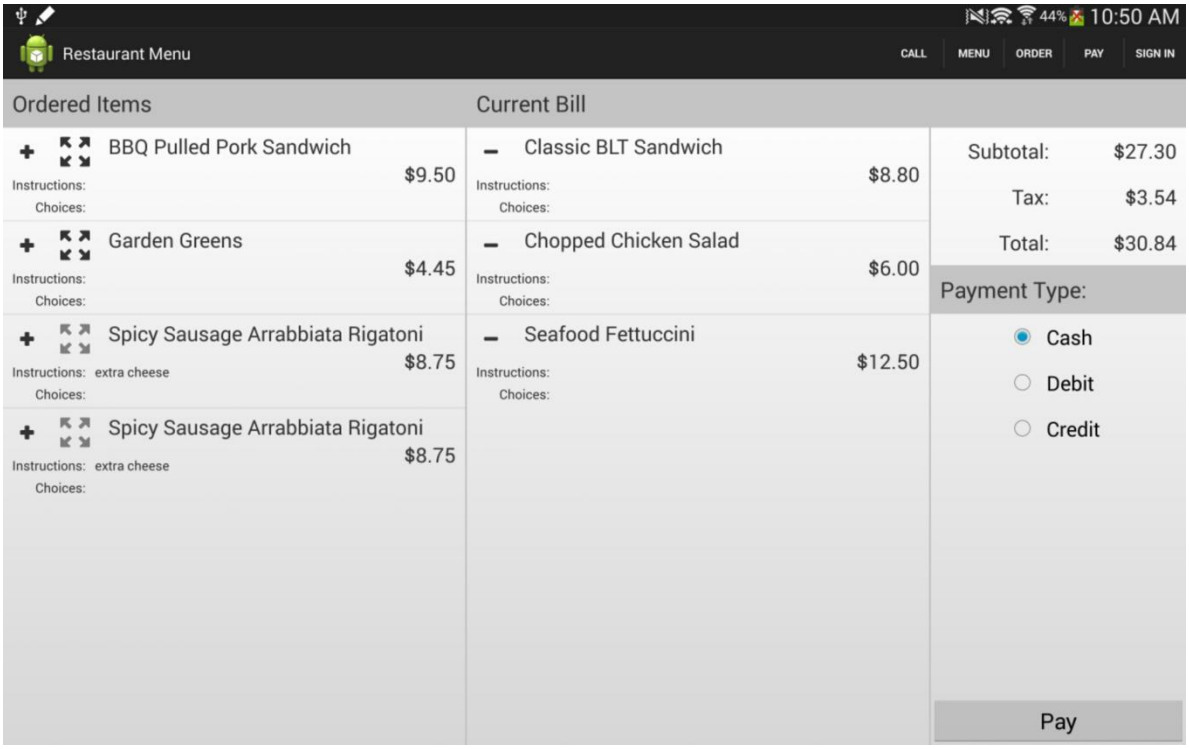


FIGURE 11- SPLIT ORDERED ITEM

The ordered items list is synchronized between all menu app devices connected to the same Wi-Fi P2P group. Once an item is added to a bill on one device it is removed from the list on all devices preventing another user from adding the same item to their bill. If an item is removed from a bill and returned the ordered items list it will be added back to the list on all devices. The same is true for item splitting. An item only needs to be split on one device and it will be reflected onto all devices.

Once the customer has selected all the items they wish to pay for, they simply choose a payment type from a radio button panel, then press the "Pay" button to submit their bill request. A Toast message is displayed to notify the customer that the bill request has successfully been sent.

WI-FI DIRECT IMPLEMENTATION: Implementation of this feature was fairly straightforward as the information shared between menu app devices is shared amongst all devices. The group owner device is responsible for keeping track of a master list of all the menu items that were ordered from the devices at the table. Each time an order is made, a request is sent out containing the ordered item information so that the list of items to be paid for could be synchronized between all devices. If the device that made the order was the group owner device, an UPDATE_PAY_ITEMS_REQUEST is sent to each peer device. Upon receiving this request, peer devices replace their current list of

ordered items with the updated list. If an order is made on a peer device, an ADD_PAY_ITEMS_REQUEST is sent to the group owner. Upon receiving this request, the group owner adds each order item contained in the request to the master list of ordered items and then broadcasts an UPDATE_PAY_ITEMS_REQUEST to all peer devices with the updated list.

When an ordered item is added to a bill and the item is removed from the ordered items list, the master list of ordered items needs to be updated. If the device is the group owner, the master list is updated immediately and an UPDATE_PAY_ITEMS_REQUEST is broadcasted to the peer devices with the updated list.

A slightly more complicated process is performed for customers using peer devices to account for a special case where more than one customer may select the same ordered item to add to their bill at the same time. If the selected item were to be simply added to the device's bill and then a request sent out to the group owner to update the master list, the customers who had selected the same item would now all have that item on their individual bills. Not only would the same item be paid for by multiple customers, but if the customers were to remove the item from their respective bills, more than one entry of the same ordered item would now appear in the list of ordered items.

To solve this problem, when an ordered item is selected to be added to a bill by a peer device, a REMOVE_PAY_ITEMS_REQUEST is sent to the group owner. Upon receiving this request, the group owner will attempt to remove the ordered item from the master list. If the item cannot be removed, simply because another REMOVE_PAY_ITEMS_REQUEST was just processed and the item is no longer in the master list, then the order item information is returned back to the peer device that issued the request in the form of a REMOVE_PAY_ITEMS_CONFIRM_REQUEST. If the item can be removed, the same request is returned to the peer device, but in this case the ordered item info is not included. Upon receiving this request the peer device checks if the ordered item information is contained in the request. If it isn't, the ordered item is added to that device's bill. In either case an UPDATE_PAY_ITEMS_REQUEST is also broadcasted by the group owner device to all peer devices with the updated master list.

When an ordered item is split by a group owner device, the master list is immediately updated and an UPDATE_PAY_ITEMS_REQUEST is broadcasted to the peer devices. If the split is done on a peer device, a SPLIT_ITEM_REQUEST is sent to the group owner containing the order item info. This info contains the number of times an item is split. Upon receiving the request, the group owner will check the number of times the item is split and attempt to split the same order item contained in

the master list. A check is done to ensure that the item has not already been split, avoiding a similar issue where more than one customer tries to split the item at the same time. If the item has not already been split, it is then split into equal portions and each portion is added to the ordered items list. The group owner will then broadcast the updated master list back to the peer devices.

TESTING

Testing for the menu and management apps was limited to mostly functional testing. This was done to ensure that the functional requirements outlined for these apps were being met and to verify that the UI layouts were being displayed as intended.

To test communication with the menu and management system's server, a simple Tomcat server was used. This mock server was configured with the same RESTful endpoints that the real system's server will have except the resources provided by these endpoints were hardcoded values. This allowed all aspects of the communication protocol between the menu and management apps with the system's server to be tested while the actual server was being developed. This also allowed the test server to be easily swapped out with the real server with relatively minor adjustments. As the project reached its final stages and the system's server and database became functional, testing was done to verify that the menu and management apps functioned with these components as intended.

The menu app is intended to be used with 10" tablets and no support for smaller devices was planned; however, due to only having one tablet available while testing the Wi-Fi Direct features, rough layouts were developed for the Samsung S3 Smartphone that the management app was being developed on. This way communication between menu app devices via Wi-Fi-Direct could be tested.

CONCLUSION

The development of the menu and management apps was a great learning experience for me. I have developed a good understanding of mobile development on the Android platform, UI development for mobile apps and Wi-Fi Direct using the Android Wi-Fi P2P framework. Some of the limitations found while working with the Android Wi-Fi P2P framework, specifically the need for users to manually accept each connection and the dependency on the group owner, may not make it the ideal solution in terms of providing the menu apps with a way to communicate and manage information with each other.

I have also gained a greater appreciation for the amount of work and time required to develop apps such as the menu and management apps and the difficulties that arise when developing a distributed system. This project also provided me with experience working with a RESTful architecture, Google Protocol Buffers and TCP sockets. Overall, I am quite satisfied with the outcome of this project and the knowledge I have gained from it.

FUTURE STUFF

MISSING FEATURES: The menu and management apps are far from complete. As with many software development projects, requirements change and often activities take longer than expected and this project was no exception. The Wi-Fi Direct features took longer than expected to implement and more effort was required to support the system's recommender service than was originally planned. Features that didn't add a lot to the main learning objectives and were considered lower priority were not implemented. These include the social media and restaurant statistics features. The social media feature would allow users of the menu app to share status updates on their Facebook and Twitter accounts. The status updates would allow users to share their experiences at the restaurant and hopefully provide some positive word of mouth advertising. The restaurant statistics feature is a significant feature for the management app, but wasn't crucial to meeting the learning objectives defined for this project. These two features should definitely be implemented in the future if the menu and management system is to be released.

EXTENSIVE TESTING: Comprehensive test suites using Android's testing framework should be created. This is a fairly big undertaking and was not done for this project. This would help flush out any existing bugs not caught with the functional testing that was performed and help to ensure the code is robust. Usability testing should also be performed to identify any user interface issues that may exist and to ensure the menu and management app will be easy to use upon release. Concurrency and scalability testing should also be performed to ensure the restaurant's data will not be corrupted and that the menu and management system can smoothly handle any size of restaurant in terms of menu size and number of devices connected to the system. For the Wi-Fi Direct features, only three devices were available to test with, so testing with additional devices to create larger Wi-Fi P2P groups should also be done.

KIOSK MODE: Due to the fact that the menu app devices are provided to the restaurant customers, it would be preferable that the customer does not have access to the Android OS and is not able to leave the menu app or shutdown / restart the device. A kiosk mode could possibly be implemented

that would prevent the user from being able to close the menu app and return to the Android OS. If this is not possible or ideal to implement then a special case could possibly be provided that will cover the device's home and power buttons.

BILL PAYMENT SUPPORT DIRECTLY ON MENU APP DEVICE: Ideally, the customer should be able to pay for their bill directly through the menu app device to further avoid dependency on the restaurant's wait staff. With the growing availability of third party card readers that support this functionality, the menu app could be integrated with existing products to provide this feature to restaurant customers. Of course, this feature may not be ideal for the restaurant owners who are purchasing the restaurant menu and management system, depending on their payment system and what options and card companies they are willing to support.

AESTHETICALLY PLEASING GRAPHICS: Creating UI layouts in Android can be a time consuming and tedious task. Creating custom graphics and fancy layout designs and/or color themes was not in the scope of this project, but could add a lot of visual appeal to the menu and management apps in the future.

ADDITIONAL SORTING OPTIONS FOR MANAGEMENT APP: The same restaurant menu sorting functionality is provided on both the menu and management apps. The sort criteria is predefined and is based on what the restaurant's customers would most likely use. Additional management specific sort criteria could be provided to help management quickly identify menu items based on criteria that provides greater business value.

USER ACCOUNT SECURITY: User account security should be provided for both the menu and management apps. For the management app, security measures could be to simply require the user to log in each time the app is opened to prevent unauthorized users from making changes to the restaurant's menu or accessing sensitive business info. The menu app should take care that customers who have signed into their loyalty accounts are protected if they have forgotten to sign out before leaving the restaurant and that access to the options menu is restricted to the restaurant's staff.

REFERENCES

About. (2014). Jersey.Retrieved February 20, 2014, from https://jersey.java.net/.

Developer Guide. (2012). Google Developers.Retrieved February 20, 2014, from https://developers.google.com/protocol-buffers/docs/overview.

Wi-Fi Direct. (2014). Wi-Fi Alliance.Retrieved February 23, 2014, from http://www.wi-fi.org/discover-wi-fi/wi-fi-direct.

Welcome. (2014). Android Developers. Retrieved February 23, 2014, from http://developer.android.com/about/index.html

Developer Tools. (2014). Android Developers. Retrieved February 23, 2014, from http://developer.android.com/tools/index.html

Managing Projects. (2014). Android Developers. Retrieved April 3, 2014, from http://developer.android.com/tools/projects/index.html

Developing for the iPhone and Android: The pros and cons. (2010). Reuters. Retrieved February 25, 2014, from http://www.reuters.com/article/2010/07/06/urnidgns002570f3005978d885257758005a352e-idUS332198778220100706

Neilson, J., Budiu, R. (2013). Mobile Usability, New Riders, pp. 55

APPENDIX A

REQUIREMENTS

Functional Requirements:

FR-BROWSE-01	Allow user to browse all the currently available menu items by category
FR-BROWSE-02	Displayed menu items will display item name, abbrev. description, price, category, tag s, count of items currently on order
FR-SEARCH-01	Allow user to search menu for specific menu items by keyword
FR-SEARCH-02	Allow user to search menu for specific menu items by price range
FR-SEARCH-03	Allow user to search menu for specific menu items by category
FR-SEARCH-04	Allow user to search menu for specific menu items by tag
FR-SORT-01	Allow user to sort displayed menu items by name
FR-SORT-02	Allow user to sort displayed menu items by price
FR-SORT-03	Allow user to sort displayed menu items by popularity

Functional Requirements (Menu App):

Id	Description
FR-MENU-CONFIG-01	Allow restaurant staff to assign the menu app to a restaurant table id
FR-MENU-CONFIG-02	Configuration of menu app should be hidden from customers and secured
FR-MENU-CONFIG-03	Allow restaurant staff to perform necessary setup of device-to-device commun ication
FR-MENU- HELP-01	Allow customer to "call" the restaurant staff for assistance at any time
FR-MENU-BROWSE-01	Only currently available menu items will be shown (ie. time of day, in stock)
FR-MENU-SELECT-01	Selecting a menu item will make all item's info available and allow customer to customize the order (choose options, input special instructions, etc.), choose the quantity, and add item to order
FR-MENU-SELECT-02	Once item selection is complete (item placed on order, or item selection is can celled), the most recent list of items the customer was looking at should be displayed
FR-MENU-SELECT-03	Allow customer to have a meal built from menu items randomly chosen from selected menu categories
FR-MENU-SHARE-01	Allow customer to share a single menu item with customer(s) at the same table
FR-MENU-ORDER-01	Allow customer to add menu items to their order
FR-MENU-ORDER-02	Allow customer to remove menu items from their order
FR-MENU-ORDER-03	Allow customer(s) at the same table to combine orders for submitting
FR-MENU-ORDER-04	Customer should be able to display all order information (before and after order made) at any time prior to payment of the order
FR-MENU-ORDER-05	Customer must be able to submit their order
FR-MENU-PAY-01	Allow customers to pay for any item(s) that were ordered at the table
FR-MENU-PAY-02	Allow customer(s) at the same table to split payment for a menu item

FR-MENU-PAY-03	Allow customer to see bill total
FR-MENU-PAY-04	Allow customer to specify payment type
FR-MENU-LOYAL-01	Customer must be able to sign in to their loyalty account at any time
FR-MENU-LOYAL-02	Customer must be able to sign out of their loyalty account at any time
FR-MENU-LOYAL-03	Customer must be able to create a new loyalty account at any time
FR-MENU-LOYAL-04	Once signed into their loyalty account, all saved preferences will be automatically applied
FR-MENU-LOYAL-05	Customer's should be automatically signed out after a payment has been made
FR-MENU-LOYAL-06	Customer's account should be protected in case they have forgotten to sign out of their account
FR-MENU-LOYAL-07	Once signed out of their account, all saved preferences and customer info will be 'forgotten' and the device will 'reset' to its default settings
FR-MENU-SOCIAL-01	Allow customer to share experience on their Facebook account
FR-MENU-SOCIAL-02	Allow customer to share experience on their Twitter account

Non - Functional Requirements (Menu App):

Id	Description
NFR-MENU-01	The menu app will follow a REST architectural style to communicate with the system's server
NFR-MENU-02	The menu app will use WI-FI-Direct technologies to communicate directly with other devices running the menu app
NFR-MENU-03	The menu app will be developed for a 10" Android 4.3 based tablet
NFR-MENU-04	Communication between menu devices at the same restaurant table should be hidden to the customers using the devices
NFR-MENU-05	The menu app should be designed for low power consumption
NFR-MENU-06	The menu app should be intuitive and easy to use (minimal learning curve)

Functional Requirements (Management App):

Id	Description
FR-MANAGE-01	User must be able to view current menu configuration
FR-MANAGE-02	User must be able to add new menu item
FR-MANAGE-03	User must be able to remove an existing menu item
FR-MANAGE-04	User must be able to modify an existing menu item
FR-MANAGE-05	User must be able to add a category to the menu
FR-MANAGE-06	User must be able to remove a category from the menu
FR-MANAGE-07	User must be able to rename a category from the menu
FR-MANAGE-08	User must be able to add a tag to the menu
FR-MANAGE-09	User must be able to remove a tag from the menu
FR-MANAGE-10	User must be able to modify a tag from the menu
FR-MANAGE-11	User must be able to add an option category to the menu
FR-MANAGE-12	User must be able to remove an option category from the menu
FR-MANAGE-13	User must be able to modify an option category from the menu

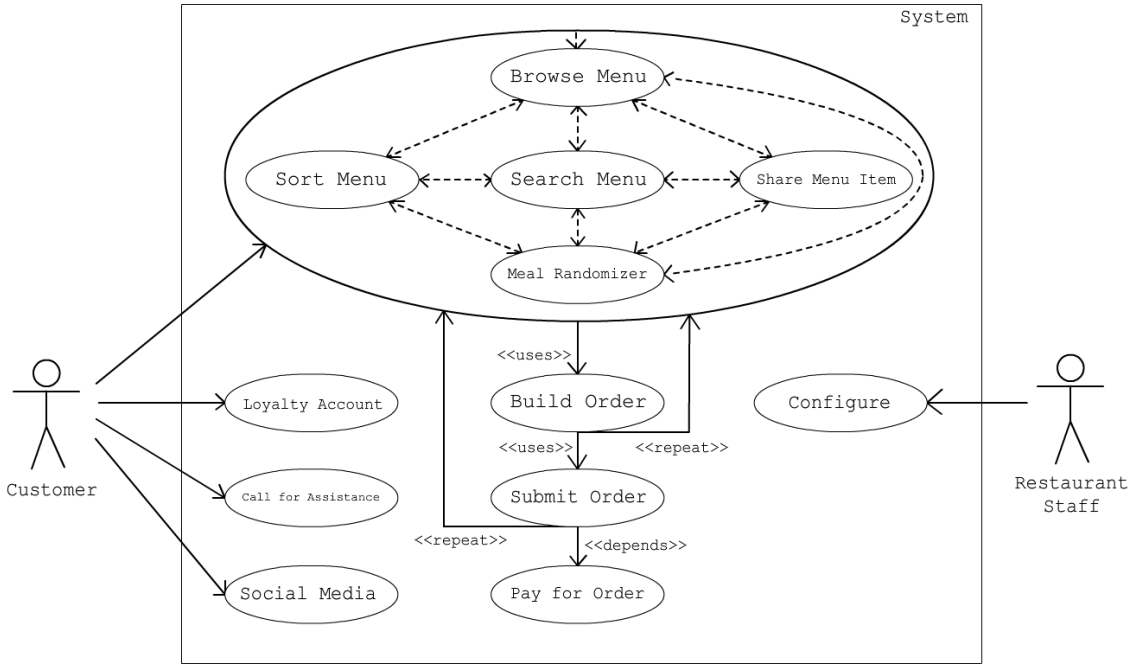
FR-MANAGE-14	User must be able to add an option choice to the menu
FR-MANAGE-15	User must be able to remove an option choice from the menu
FR-MANAGE-16	User must be able to modify an option choice from the menu
FR-MANAGE-17	User must be able to specify time of day menu item should be available to order
FR-MANAGE-18	User must be able to view revenue information over a specified period of time
FR-MANAGE-19	User must be able to view menu item popularity over a specified period of time
FR-MANAGE-20	User must be able to view customer traffic information over a specified period of time
FR-MANAGE-21	User must log in before using the application

Non - Functional Requirements (Management App):

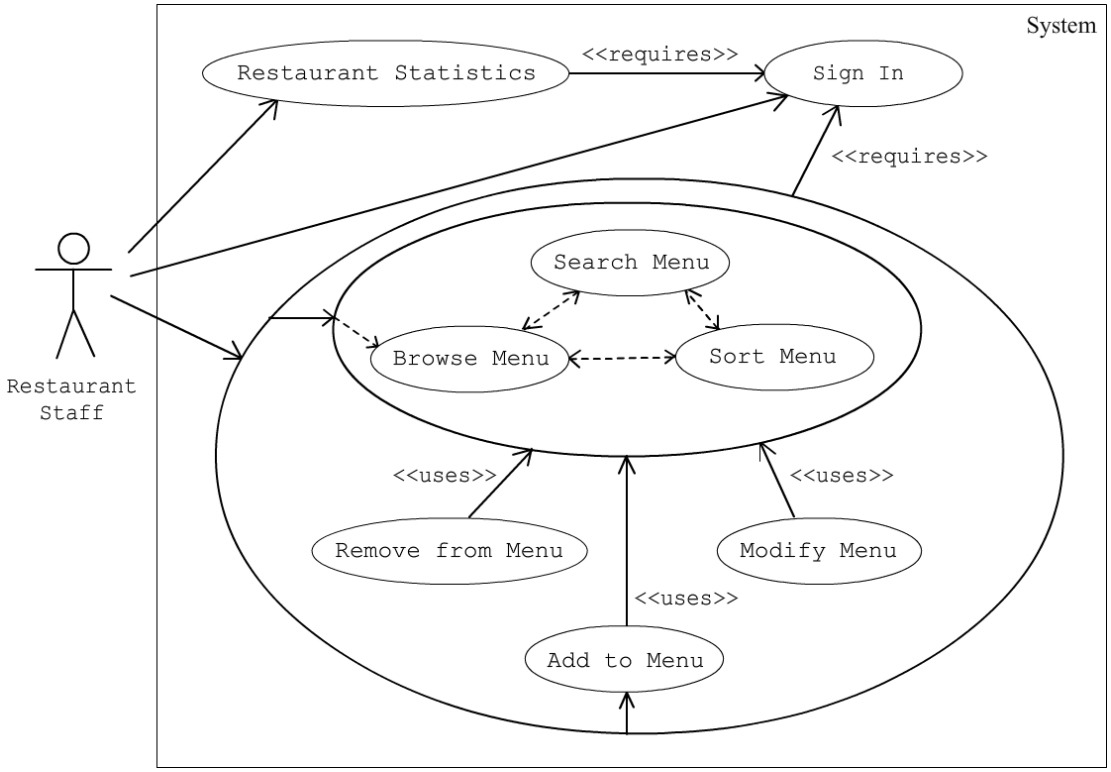
Id	Description
NFR-MANAGE-01	The management app will follow a REST architectural style to communicate with the system's server
NFR-MANAGE-02	The management app will be developed for a 10" Android 4.3 based tablet and an Android 4.3 based Smartphone
NFR-MANAGE-03	The management app may use a 3rd party library to render graphs and charts

USE CASES

Menu App Use Case Diagram:



Management App Use Case Diagram:



UC-01	Browse Menu	Traceability
Summary	User browses menu items by category	
External Actors	Customer, Restaurant Staff	
Triggering Event	User wants to look through the menu	
Pre-Condition		
Main Sequence	1. User selects menu option	
	2. System displays default category (recommended items)	
	3. User selects a menu category (if not already in category)	FR-BROWSE-01
	4. System displays all menu items that belong to the selected category	FR-BROWSE-02
Result	User can now browse menu items from the selected category	
Post-Condition	System is displaying all items from selected category	
Alternatives	**UC-01-ALT-01:** If the user is using the menu app, then only the currently available menu items are displayed (items that can and are available to be ordered at that time)	FR-MENU-BROWSE-01

UC-02	Search Menu	Traceability
Summary	User searches available menu items	
External Actors	Customer, Restaurant Staff	
Triggering Event	User wants to search for certain menu item(s)	
Pre-Condition	System is displaying menu	
Main Sequence	1. User selects to search menu	
	2. System displays simple search box	
	3. User enters keyword(s) to search menu for	FR-SEARCH-01
	4. System displays search results	
Result	User has searched for certain menu item(s)	
Post-Condition	Menu items matching search criteria are displayed	
Alternatives	**UC-02 -ALT-01:** If the user would like narrow their search they can choose additional search criteria (advanced search)	FR-SEARCH-02 FR-SEARCH-03 FR-SEARCH-04
	UC-02-ALT-02: If the user is using the menu app, then only the currently available menu items are displayed (items that can and are available to be ordered at that time)	FR-MENU-BROWSE-01

UC-03	Sort Menu	Traceability
Summary	User sorts the menu items that are currently being displayed	
External Actors	Customer, Restaurant Staff	
Triggering	User wants to sort the menu items	

Event		
Pre-Condition	There are menu items to sort being displayed	
Main Sequence	1. User selects sort criteria	FR-SORT-01 FR-SORT-02 FR-SORT-03
	2. System displays the sorted results	
Result	Displayed menu items have been sorted	
Post-Condition	Menu items are now displayed in sorted order	

Menu App Specific Use Cases:

UC-04	Loyalty Account	Traceability
Summary	Customer signs into loyalty account, possibly updates their loyalty account information, and eventually signs out	
External Actors	Customer	
Triggering Event	Customer selects to sign in to loyalty account	
Pre-Condition	No customer is currently signed into their account	
Main Sequence	1. Customer selects to sign into loyalty account	FR-MENU-LOYAL-01
	2. System displays loyalty sign in screen	
	3. Customer enters account credentials	
	4. System verifies account credentials and retrieves customer's account information	
	5. System returns customer to where they were before they selected to sign in to account	
	6i. Customer selects to configure their account	
	6ii. System displays customers account information	
	6iii. Customer updates information	
	6iv. Systems updates and saves customer account info and returns to step 5	FR-MENU-LOYAL-04
	7. Customer selects to sign out	FR-MENU-LOYAL-02
	8. System signs customer out of their account	
Result	Customer is signed out of their account	
Post-Condition	Menu is returned to default settings and customer account information is no longer available	FR-MENU-LOYAL-07
Alternatives	**UC-04-ALT-01**: If the customer does not have a loyalty account, they may create a new account at the sign in screen	FR-MENU-LOYAL-03
	UC-04-ALT-02: If the customer does not interact with the menu for a period of time, they will be required to enter their password to remain signed in the next time the menu is used	FR-MENU-LOYAL-06
	UC-04-ALT-03: The customer will be automatically signed out once an order has been paid for	FR-MENU-LOYAL-05

UC-05	Share Menu Item	Traceability
Summary	A customer shares a menu item they are currently looking at with another customer at the same table	
External Actors	Customers	
Triggering Event	Customer would like to share a menu item	
Pre-Condition	Customer is looking at a particular menu item	
Main Sequence	1. Customer selects to share item	
	2. System displays list of all devices to share with	
	3. Customer selects each devices they want to share with	FR-MENU-SHARE-01
	4. System notifies customer that menu item has been shared and returns customer to display of shared menu item	
	5. System asks customers whom menu item is being shared with if they want to view the shared item	
	6. Customer accepts share request	
	7. System displays menu item	
Result	Customer has shared a menu item	
Post-Condition	Customers whom menu item has been shared with have received the share request	
Alternatives	**UC-05-ALT-01:** If the customer declines the share request the system simply does not show the shared menu item	
	UC-05-ALT-02: Customer may swipe in direction of the device they want to share with at step 3 instead of selecting devices from the list	

UC-06	Meal Randomizer	Traceability
Summary	Customer chooses to have system randomly generate a meal	
External Actors	Customer	
Triggering Event	Customer wants to see a randomly generated meal	
Pre-Condition	Customer is looking at menu	
Main Sequence	1. Customer selects meal generator	
	2. System displays categories	
	3. Customer selects categories they want a menu item from	
	4. System randomly selects and displays a menu item from each category	FR-MENU-SELECT-03
Result	A meal has been randomly generated	
Post-Condition	Generated meal is displayed for the customer	

UC-07	Build order	Traceability

Summary	Customer builds an order to submit	
External Actors	Customer	
Triggering Event	Customer wants to order	
Pre-Condition	Customer has found a menu item they would like to add to their order	
Main Sequence	1. Customer selects to add menu item to order	FR-MENU-ORDER-01
	2. System displays menu item details including order options	
	3. Customer selects options they would like and specifies any special instructions	FR-MENU-SELECT-01
	4. System adds item to order and confirms item has been added	
	5. If customer would like to add additional items to the order they must find another item and repeat from step 1	
Result	Order has been built	
Post-Condition	There are menu items ready to be ordered	
Alternatives	**UC-07-ALT-01:** If the customer wishes to remove an item from their order they may do so selecting to remove the item at step 1 (assumes item has been previously added to the order). If a menu item has been added multiple times, the system will provide the customer with a choice of which of those items to remove.	FR-MENU-ORDER-02
	UC-07-ALT-02: If the menu item has no available options step 2 and 3 skipped. If later the menu item is asked to be removed and more than one of same item has been added, then the system automatically removes 1 of the items	

UC-08	**Submit order**	**Traceability**
Summary	Customer(s) at a table submit their order	
External Actors	Customer(s)	
Triggering Event	Customer(s) want to submit their order	
Pre-Condition	An order has been built	
Main Sequence	1. Customer selects order	
	2. System displays order info, this includes items they have selected to order and all pending and previous orders	
	3. Customer selects items they would like to order	
	4i. If customer wishes to combine their order with other customers at the same table, they choose this option now	FR-MENU-ORDER-03
	4ii. System displays a list of all devices to combine order with	
	4iii. Customer selects devices they want to combine order with	
	5. Customer submits order	FR-MENU-ORDER-

		05
	6. System submits the order when all orders that are to be combined with this order have been submitted	
Result	An order has been submitted	
Post-Condition	All items that have been ordered are now marked as ordered and require payment	
Alternatives	**UC-08-ALT-01:** If the customer would like to add items to a pending order they select this option at step 4. This will add the items to the pending order and update other devices that are combines with this order.	
	UC-08-ALT-02: If the customer would like to cancel a pending order they can select this option at step 3. The system will remove them from the combined order and update the other devices combined with the order.	
	UC-08-ALT-03: If the customer would like to remove an item from a pending order the can select this option at step 3. The system will remove the item from the pending order and update the other devices combined with the order.	
	UC-08-ALT-04: The customer may submit a pending order at step 3 by indicating that they are ready for the pending order to be submitted. They then skip to step 6.	
	UC-08-ALT-05: If a pending order is still waiting on another device to indicate they are ready to submit, the customer may cancel their ready state by repeating the same steps as **UC-08-ALT-04**.	

UC-09	**Pay for Order**	**Traceability**
Summary	Customer(s) pay for their order(s)	
External Actors	Customer(s)	
Triggering Event	Customer(s) would like to pay for their order	
Pre-Condition	An order has been submitted and has not been paid for	
Main Sequence	1. Customer selects to pay	
	2. System lists all items that have not been paid for that have been ordered from the table, as well as current bill info and payment method options	FR-MENU-PAY-03
	3. Customer selects item(s) they wish to pay for	FR-MENU-PAY-01
	4. System updates bill	
	5. Customer selects payment method	FR-MENU-PAY-04
	6. System sends notification to Restaurant Wait Staff app	
	7. System informs customer that the restaurant staff has been notified of payment request	
Result	Payment for menu items has been requested	
Post-Condition	Menu items that are on bill are no longer marked as unpaid	
Alternatives	**UC-9-ALT-01**: Customer may want to split payment for a single menu item. The can select this option at step 3. The	FR-MENU- PAY -02

	system will allow the customer to specify number of bills to split the item with. The system will split the item. Customer can return to step 3 and select split item portion to add to bill.	

UC-10	Call for Assistance	Traceability
Summary	Customer calls the restaurant staff to the table for assistance	
External Actors	Customer	
Triggering Event	Customer would like to speak with the restaurant staff	
Pre-Condition	Menu app is open	
Main Sequence	1. Customer selects "Call" button	FR-MENU- HELP-01
	2. System sends notification to Restaurant Wait Staff app	
	3. System informs customer that the restaurant staff has been called	
Result	Restaurant staff has been called to the table	
Post-Condition	A notification has been sent to the Restaurant Wait Staff app requesting assistance at the customers table	

UC-11	Social Media	Traceability
Summary	Customer shares experience on social media account	
External Actors	Customer	
Triggering Event	Customer wishes to share experience with a social media	
Pre-Condition		
Main Sequence	1. Customer selects social media platform to share with	
	2. System shows social media log in screen	
	3. Customer enters in social media account credentials	
	4. System displays social media platform specific share options	FR-MENU-SOCIAL-01 FR-MENU-SOCIAL-02
	5. Customer enters relevant info	
	6. System posts onto social media	
Result	Customer has shared their experience on their social media	
Post-Condition		

UC-12	Configure	Traceability
Summary	Restaurant staff configures the app	
External Actors	Restaurant staff	
Triggering	Configuration change is required	

Event		
Pre-Condition	Menu app is open	
Main Sequence	1. Restaurant staff opens the settings menu	FR-MENU-CONFIG-02
	2. System asks for user's credentials	
	3. Restaurant staff enters credentials	
	4. System verifies credentials before opening the settings menu	
	5. Restaurant staff selects setting to configure	FR-MENU-CONFIG-01 FR-MENU-CONFIG-03
	6. Restaurant staff makes configuration change	
	7. Restaurant staff saves and exits settings menu	
	8. System saves configuration change	
Result	Configuration change has been made	
Post-Condition	Menu app is configured	

Management App

UC-13	Add to Menu	Traceability
Summary	User adds to the menu	
External Actors	Restaurant staff	
Triggering Event	User would like to add to the menu	
Pre-Condition	User is building the menu	
Main Sequence	1. User selects to add an element	FR-MANAGE-02 FR-MANAGE-05 FR-MANAGE-08 FR-MANAGE-11 FR-MANAGE-14
	2. System displays add item template	
	3. Restaurant staff fills in item information	FR-MANAGE-17
	4. System verifies item is not already on the menu	
	5. System adds item to the menu	
	6. System confirms item has been added	
Result	Menu item has been added to the menu	
Post-Condition	Menu item is on the menu	

UC-14	Remove from Menu	Traceability
Summary	User removes an item from the menu	
External Actors	Restaurant staff	
Triggering Event	User would like to remove an item from the menu	
Pre-Condition	Item is currently on the menu and has been selected	

Main Sequence	1. Restaurant staff selects to remove the item	FR-MANAGE-03 FR-MANAGE-06 FR-MANAGE-12 FR-MANAGE-15
	2. System displays remove item confirmation	
	3. Restaurant staff confirms removal	
	4. System removes item from the menu	
	5. System confirms item has been removed	
Result	Menu item has been removed from the menu	
Post-Condition	Menu item is no longer on the menu	

UC-15	**Modify Menu**	**Traceability**
Summary	User modifies an item on the menu	
External Actors	Restaurant staff	
Triggering Event	User would like to modify an item on the menu	
Pre-Condition	Item is currently on the menu and has been selected	
Main Sequence	1. Restaurant staff selects to modify the item	FR-MANAGE-04 FR-MANAGE-10 FR-MANAGE-13 FR-MANAGE-16
	2. System displays modify item template	
	3. Restaurant staff updates item information	
	4. System displays modify item confirmation	
	5. Restaurant staff confirms modification	
	6. System updates item information	
	7. System confirms item has been modified	
Result	Menu item has been modified from the menu	
Post-Condition	Menu item is modified with new information	

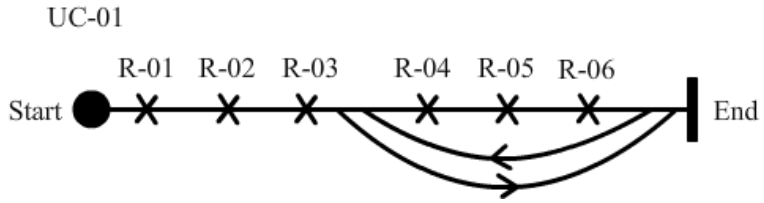
UC-16	**Restaurant Statistics**	**Traceability**
Summary	User looks at restaurant statistics	
External Actors	Restaurant staff	
Triggering Event	User would like to view restaurant statistics	
Pre-Condition		
Main Sequence	1. Restaurant staff selects to view restaurant statistics	
	2. System displays statistics	
	3. Restaurant staff selects statistic type	FR-MANAGE-18 FR-MANAGE-19 FR-MANAGE-20
	4. System generates and displays graph for default date range and the respective report	

	5. User may specify date range	
	5i. System generates and displays graph for specified date range and the respective report	
Result	User can now view restaurant statistics	
Post-Condition	Restaurant statistics and report are being displayed	

UC-17	**Sign In**	**Traceability**
Summary	User would like to use the management app	
External Actors	Restaurant staff	
Triggering Event	User starts app or resumes using app after a period of time	
Pre-Condition		
Main Sequence	1. System displays sign in	
	2. User enters credentials	FR-MANAGE-21
	3. System verifies credentials	
Result	User can use the management app	
Post-Condition	Restaurant menu is being displayed	

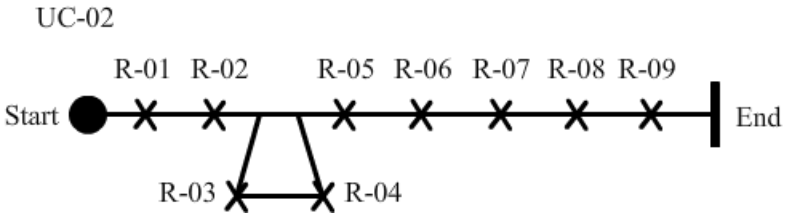
USE CASE MAPS

UCM-01 - Browse Menu



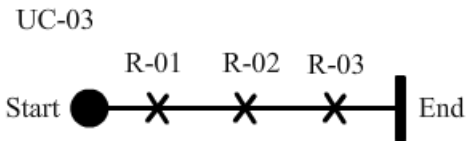
Start	
R-01	User selects Menu
R-02	System retrieves default menu category items
R-03	System displays default menu category items
R-04	User selects category
R-05	System retrieves menu category items
R-06	System displays menu category items
End	System is displaying all menu items that belong to the selected category

UCM-02 - Search Menu



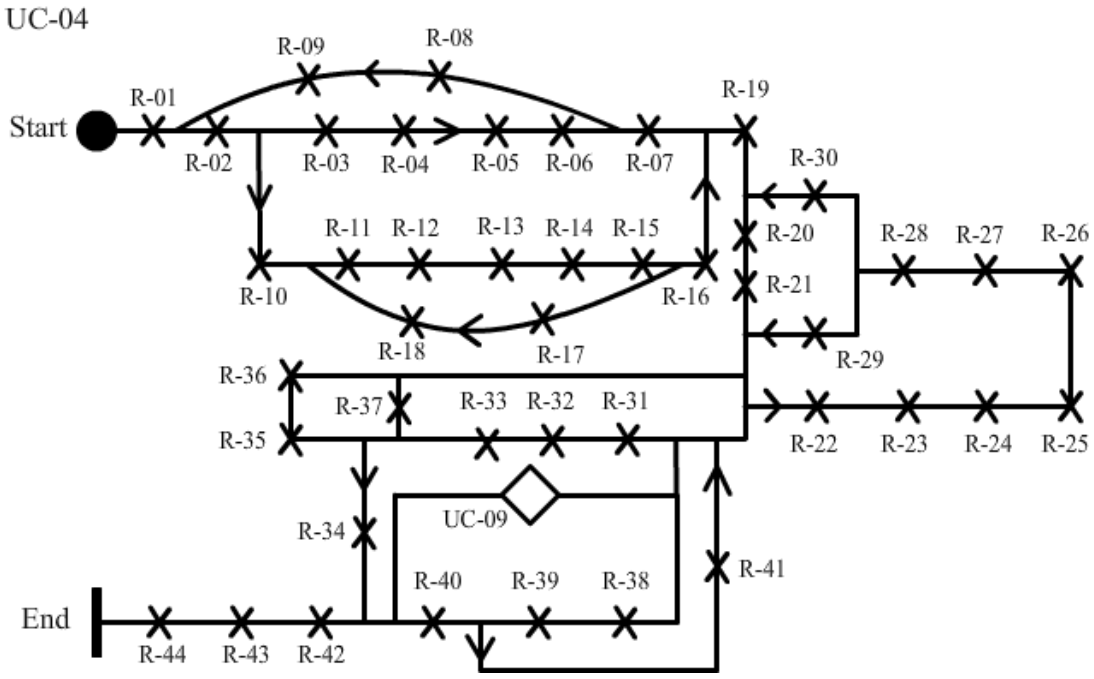
Start	
R-01	User selects Search
R-02	System displays basic search
R-03	User selects advanced search
R-04	System displays advanced search
R-05	User enters search criteria
R-06	User submits search query
R-07	System sends search query to server
R-08	System receives query results from server
R-09	System displays result
End	Menu items matching search criteria are displayed

UCM-03 - Sort Menu



Start	There are menu items to sort being displayed
R-01	User selects sort criteria
R-02	System sorts menu items that are currently being displayed
R-03	System displays sorted menu items
End	Menu items are now displayed in sorted order

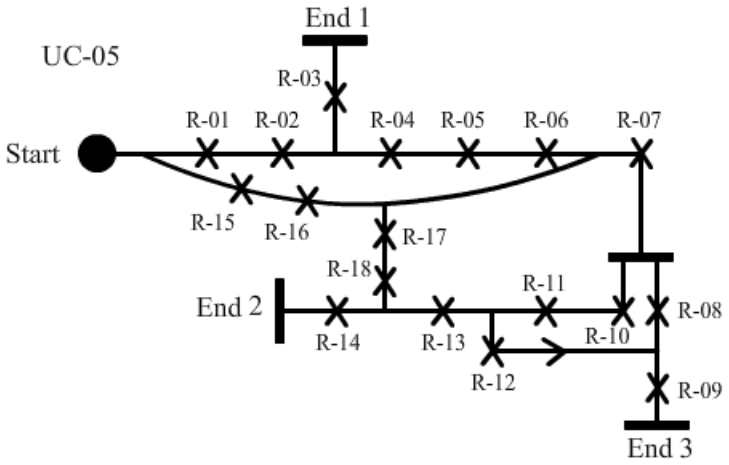
UCM-04 - Loyalty Account



Start	
R-01	User selects Loyalty Sign In
R-02	System displays loyalty sign in
R-03	User enters account credentials
R-04	User submits sign in requests
R-05	System sends sign in request to the server
R-06	System receives sign in response from the server
R-07	System displays welcome message
R-08	System displays sign in error
R-09	User acknowledges error
R-10	User selects to create loyalty account
R-11	System displays loyalty account creation

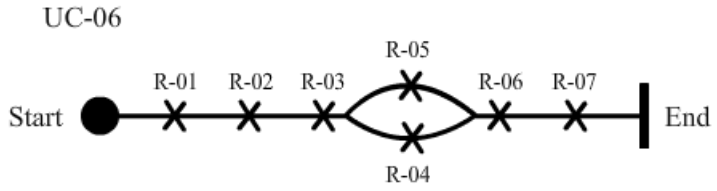
R-12	User enters account creation info
R-13	User submits create account request
R-14	System submits create account request to the server
R-15	System receives create account response from the server
R-16	System displays welcome message
R-17	System displays create account error
R-18	User acknowledges error
R-19	System displays previous screen before user selected Loyalty Sign in
R-20	System stores account info
R-21	System updates display with customer info
R-22	User selects Loyalty Account
R-23	System retrieves account info
R-24	System displays account info
R-25	User updates account info
R-26	User submits update account request
R-27	System send update account request to the server
R-28	System receives update account response from the server
R-29	System displays error message
R-30	System displays update confirmation
R-31	User does not interact with system for some time
R-32	User starts to interact with system
R-33	System prompts user for account credentials
R-34	User selects to sign out
R-35	User enters account credentials
R-36	System verifies account credentials
R-37	System displays verification error
R-38	User selects Loyalty Sign Out
R-39	System asks user for sign out confirmation
R-40	User enters account credentials
R-41	System verifies account credentials
R-42	System displays verification error
R-43	User selects Loyalty Sign Out
R-44	System asks user for sign out confirmation
End	User is signed out of their loyalty account

UCM-05 - Share Menu Item



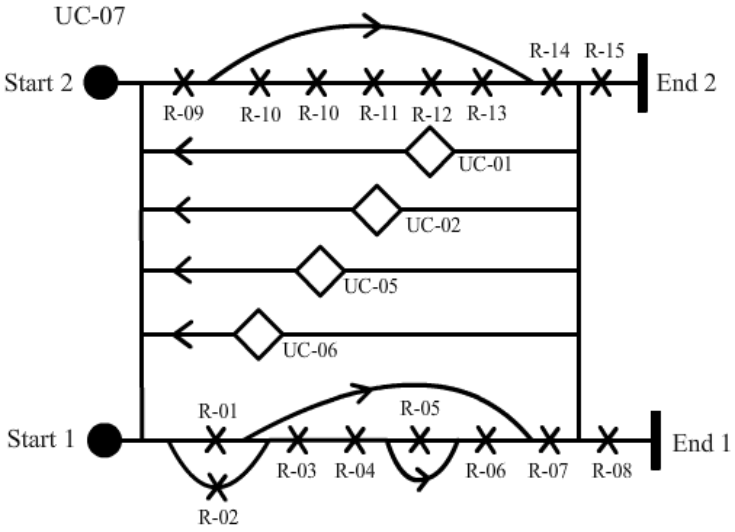
Start	Menu item is currently being displayed
R-01	User selects Share Item
R-02	System retrieves names of devices on the same table Wi-Fi network
R-03	System displays message informing user there are no devices to share with
R-04	System displays names of devices on the same table Wi-Fi network
R-05	User selects devices to share menu item with
R-06	User submits share request
R-07	System send share request
R-08	System informs user share request has been sent
R-09	System displays menu item that was shared
R-10	System receives share request (sent from another device)
R-11	System informs user of share request and asks whether they would like to accept
R-12	User accepts
R-13	User declines
R-14	System returns user to previous activity
R-15	User swipes in direction of another device at the same table
R-16	System determines which device is in direction of swipe
R-17	System displays error message if no device is identified
R-18	User acknowledges error message
End 1	User is returned to menu item display and no item has been shared
End 2	User is returned to previous activity
End 3	Shared menu item is displayed

UCM-06 - Meal Randomizer



Start	
R-01	User selects Meal Generator
R-02	System displays meal generator categories
R-03	User selects categories
R-04	User shakes the device
R-05	User requests meal generation
R-06	System generates random meal based on users category selection
R-07	System displays menu items that make up the generated meal
End	Generated meal is displayed

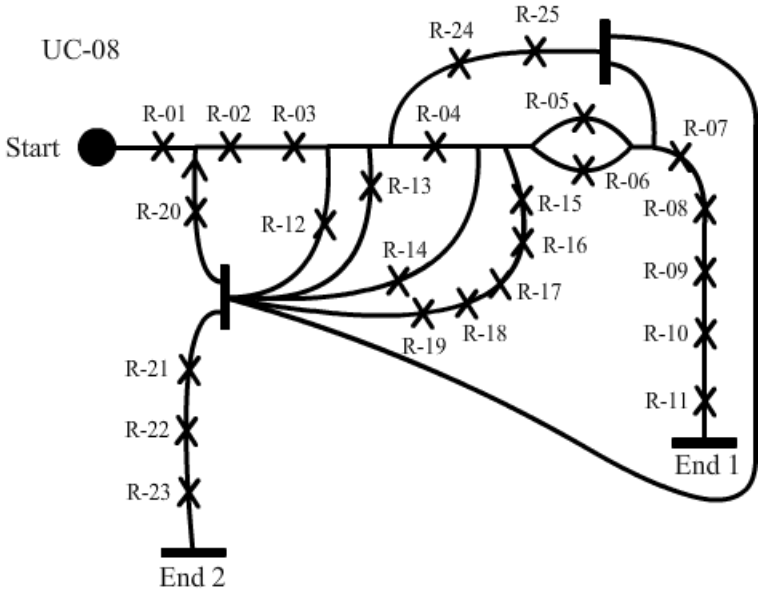
UCM-07 - Build Order



Start 1	Menu item is currently being displayed
Start 2	Menu item is currently being displayed and has been previously added to the order
R-01	User selects add item
R-02	User selects menu item
R-03	System retrieves item info
R-04	System displays item info
R-05	User selects item options
R-06	User requests to add item to order
R-07	System adds item to order
R-08	System confirms item has been added to the order

R-09	User selects to remove item
R-10	System retrieves previously added items of the same type
R-11	System displays previously added items of the same type
R-12	User selects which items to remove
R-13	User requests to remove item from order
R-14	System removes item from order
R-15	System confirms item has been removed from order
End 1	User declines
End 2	System returns user to previous activity

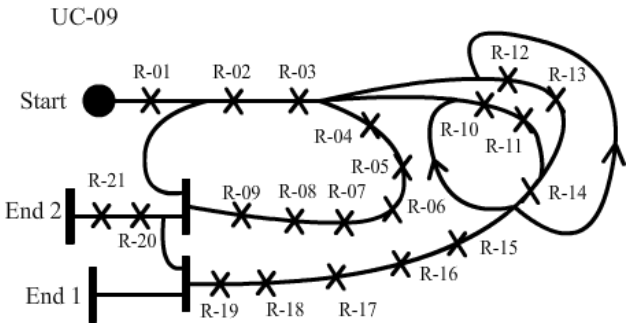
UCM-08 - Submit Order



Start 1	
Start 2	
R-01	User selects Order
R-02	System retrieves order information
R-03	System displays order information
R-04	User selects menu items
R-05	User selects Submit
R-06	User flips device
R-07	System submits order request to server
R-08	System receives order response from server
R-09	System displays order confirmation
R-10	System updates order info
R-11	System displays updated order info
R-12	User selects item to remove from pending order
R-13	User selects to cancel pending order
R-14	User selects to add selected items to pending order

R-15	User selects to submit order with other devices
R-16	System retrieves names of devices on the same table Wi-Fi network
R-17	System displays names of devices on the same table Wi-Fi network
R-18	User selects devices to submit order with
R-19	User selects Submit
R-20	System updates order info
R-21	System send order update to other devices that are to share order
R-22	System receives order update from another device
R-23	System updates order info
R-24	User selects to submit shared order
R-25	System checks if user is one to submit shared order
End 1	An order has been submitted
End 2	Order information has been updated

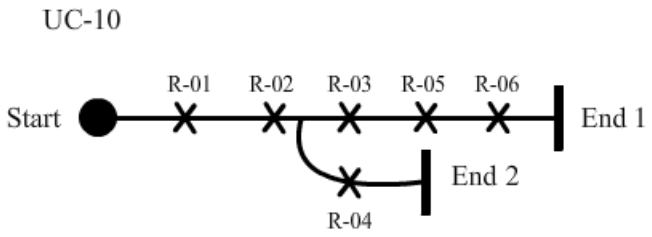
UCM-09 - Pay for Order



Start 1	
R-01	User selects Pay
R-02	System retrieves payment information
R-03	System displays payment information
R-04	User selects to split payment of a menu item
R-05	System asks user for number of times to split item
R-06	User enters number of times to split item
R-07	User selects Split
R-08	System updates payment information
R-09	System sends payment change notification to other devices on the same table Wi-Fi network
R-10	User selects menu item to add to bill
R-11	System adds menu item to bill
R-12	User unselects menu item to remove from bill
R-13	System removes menu item from bill
R-14	System updates payment info
R-15	User selects payment method
R-16	User requests to pay
R-17	System sends payment request
R-18	System displays payment request confirmation

R-19	System sends payment change notification to other devices on the same table Wi-Fi network
R-20	System receives payment change notification from another device
R-21	System updates payment information
End 1	Payment for menu items has been requested
End 2	Payment information has been updated

UCM-10 - Call for Assistance

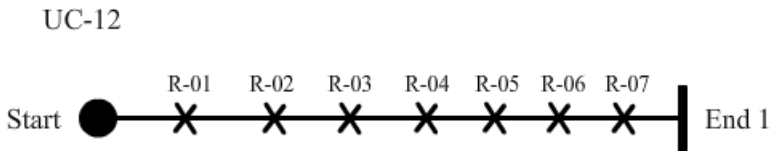


Start 1	
R-01	User selects Call
R-02	System displays call confirmation
R-03	User confirms
R-04	User cancels
R-05	System send call request to Restaurant Server App
R-06	System displays call request confirmation
End 1	A call request has been sent out to the Restaurant Server App
End 2	Call request has been cancelled

UCM-11 - Social Media

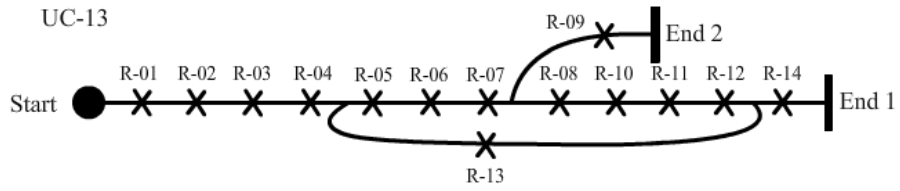
Start 1	
R-01	User selects social media share
R-02	System displays social media specific share information
R-03	User enters social media share information
R-04	User submits social media share request
R-05	System submits request using social media platform specific API
R-06	System displays call request confirmation
End 1	User has shared experience on their social media account

UCM-12 - Configure



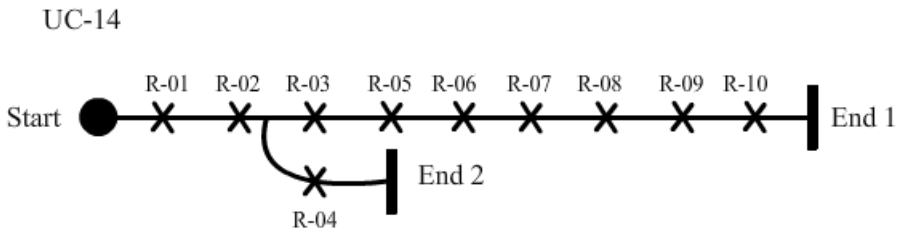
Start 1	
R-01	User selects Options
R-02	System displays available options
R-03	User select option they wish to configure
R-04	System displays appropriate option configuration properties
R-05	User configures property
R-06	User closes options
R-07	System displays previous activity
End 1	User has configured options

UCM-13 - Add to Menu



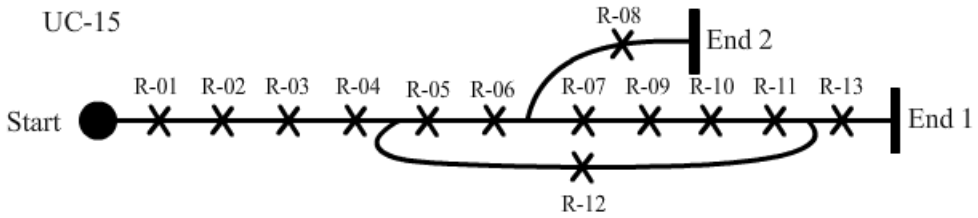
Start 1	
R-01	User selects Add Menu Element
R-02	System displays element types
R-03	User selects element type
R-04	System displays new element template
R-05	User enters new element information
R-06	User request to add new menu element
R-07	System displays add menu element confirmation
R-08	User confirms
R-09	User cancels
R-10	System sends add element request to server
R-11	System receives add element response from server
R-12	System displays response message
R-13	User acknowledges response message
R-14	System returns user to previous activity
End 1	A new menu element has been added to the menu
End 2	Add menu element request has been cancelled

UCM-14 - Remove from Menu



Start 1	
R-01	User selects menu element to remove
R-02	System displays removal confirmation
R-03	User confirms
R-04	User cancels
R-05	System sends menu element removal request to server
R-06	System receives menu element response from server
R-07	System displays response message
R-08	System receives add element response from server
R-09	System displays response message
R-10	System returns user to previous activity
End 1	Menu element has been removed from the menu
End 2	Menu element removal has been cancelled

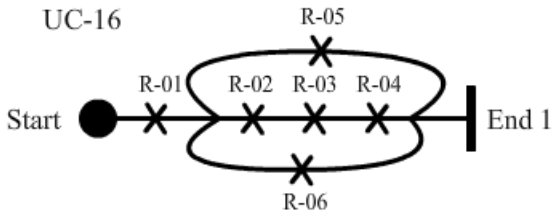
UCM-15 - Modify Menu



Start 1	
R-01	User selects menu element to modify
R-02	System retrieves menu element information
R-03	System displays modify menu element template
R-04	User makes menu element modifications
R-05	User requests menu element modify
R-06	System displays menu element modification confirmation
R-07	User confirms
R-08	User cancels
R-09	System sends menu element modification request to server
R-10	System receives menu element modification response from server
R-11	System displays response message
R-12	User acknowledges response message
R-13	System returns user to previous activity

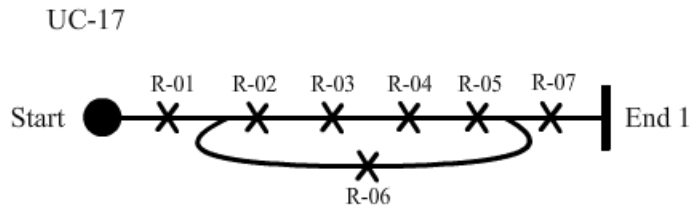
End 1	Menu element has been modified
End 2	Menu element modification request has been cancelled

UCM-16 - Restaurant Statistics



Start 1	
R-01	User selects Restaurant Statistics
R-02	System sends statistics request to server
R-03	System receives statistics response from server
R-04	System displays statistics information
R-05	User selects statistics type
R-06	User selects statistics date range
End 1	Menu element has been modified

UCM-17 - Sign In



Start 1	
R-01	System displays sign in
R-02	User enters sign in credentials
R-03	System sends sign in request to server
R-04	System receives sign in response from server
R-05	System displays sign in message
R-06	User acknowledges response message
R-07	System displays menu
End 1	User is signed in