

University of Nevada, Reno

Mobile Food Ordering System (MFOS)

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in
Computer Science

by

Hrishikesh Kulkarni

Dr. Sergiu Dascalu/Thesis Advisor

May, 2009



University of Nevada, Reno
Statewide • Worldwide

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

HRISHIKESH KULKARNI

entitled

Mobile Food Ordering System (MFOS)

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Dr. Sergiu Dascalu, Advisor

Dr. Frederick Harris, Committee Member

Dr. George Danko, Graduate School Representative

Marsha H. Read, Ph. D., Associate Dean, Graduate School

May, 2009

Abstract

Mobile devices and wireless technologies are making a large impact on our lives. Especially mobile devices have been widely used in the last few years, and recent developments in wireless technologies have provided some new solutions to modes of connectivity. In recent times, there have been dramatic changes in the field of mobile devices with the goal of improving the mobile environment. Companies that develop mobile applications are in the process of continuously evolving their products for creating tools that satisfy the customers' needs even more comprehensively and better.

The Mobile Food Ordering System (MFOS) proposed in this thesis is one of the tools that intends to provide a food ordering application on mobile devices for different food stores with the option of both delivery and pickup. Using the MFOS product, the customers can choose to pick up their order from near-by stores, and they can check store locations using the map feature of MFOS. Customers can also choose the delivery option and the order will be delivered to their address. It will be helpful to MFOS customers to choose the food items from a menu rather than spending more time on a call with the operator. By using MFOS the users do not have to wait in a queue at the restaurants. Hence, this is a win-win situation for both the consumers and the restaurants.

This thesis presents the background in terms of new technologies in handheld devices, and current research in human-computer interaction for mobile applications. It also surveys several applications of handheld devices and discusses their characteristics. This thesis also explains the main functions, key usability aspects, intended users, software model, and prototype of MFOS. The thesis also briefly compares MFOS with

related applications in terms of functionality and feature offerings. Directions of future work for MFOS are also outlined in the thesis.

Keywords: Pocket PC application, food ordering system, mobile ordering system, human-computer interaction, software model, prototype.

Acknowledgements

I would like to take this opportunity to thank my advisor Dr. Sergiu Dascalu for his support, guidance, and encouragement. I would also like to thank Dr. Frederick Harris, Jr. and Dr. George Danko for serving on my committee and providing valuable feedback on my thesis. I would also like to thank both Dr. George Danko and Dr. Sergiu Dascalu for giving me opportunity to work with them.

Further, I am grateful to Mr. Jim Hunt, my supervisor at Bally Technologies, Reno, NV for his continued support and understanding.

I would also like to thank Md Moynul Haque, with whom I started working on MobileFoodOrderingSystem (MFOS) project in Software Engineering class.

I would like to thank my parents, Prakash and Sushma, and also my brother Abhijit as well as sister-in-law Swapnali. I would not be here without you and your help. I would also like to thank all my friends and other family members specially Kedar and Manjari for their love and support.

Table of Contents

1	INTRODUCTION	1
2	BACKGROUND	4
2.1	New Technologies in Handheld Devices	4
2.1.1	Handheld Devices	4
2.1.2	Recent Communication Technologies in Handheld Devices.....	6
2.1.3	Development for Handheld Devices	7
2.2	Current Research in Human-Computer Interaction for Mobile Applications.....	8
2.2.1	User Characteristics and Limitations of Mobile Devices	8
2.2.2	Designing Mobile Applications Considering Limitations of Handheld Devices	10
2.3	Handheld Device Applications	12
2.3.1	Amazon Mobile	12
2.3.2	ETRADE Mobile	13
2.3.3	CityMint.....	14
3	MFOS: AN OVERVIEW	15
3.1	Main functions of MFOS	15
3.2	Key Usability Aspects of MFOS	16
3.3	Intended Users of MFOS	17
4	MFOS: SOFTWARE MODEL.....	18
4.1	Requirement Specification.....	18
4.1.1	Functional Requirements	18
4.1.2	Non-Functional Requirements	20
4.2	Use Case Diagram.....	20
4.3	Detailed Use Cases	21
4.4	Traceability matrix.....	31
4.5	Layered Architecture	32
4.6	Database Tables of MFOS	33
4.7	Statechart of MFOS	35
5	PROTOTYPE DETAILS.....	37
5.1	MFOS Platform.....	37
5.2	MFOS Prototype	41
5.3	Usability Components of the MFOS Prototype	54
5.4	Connectivity Aspects	57
6	COMPARISON WITH SIMILAR WORK	59
7	FUTURE WORK.....	64

8 CONCLUSIONS..... 66

REFERENCES 68

List of Tables

Table 4.1 Functional Requirements	19
Table 4.2 Non-Functional Requirements	20
Table 4.3 Requirement Traceability Matrix.....	31
Table 5.1 List of confirmation and error messages of the MFOS prototype	56
Table 6.1 Comparison with related tools	62

List of Figures

Figure 4.1 Use Case Diagram of MFOS	21
Figure 4.2 View Previous Orders detailed use case.....	22
Figure 4.3 Choose Food Store detailed use case.....	23
Figure 4.4 Choose Order Method detailed use case	24
Figure 4.5 Choose Store Location detailed use case.....	25
Figure 4.6 View Store Location detailed use case	26
Figure 4.7 Call Store detailed use case	27
Figure 4.8 Choose Food Item detailed use case.....	28
Figure 4.9 Choose Specials detailed use case	29
Figure 4.10 Enter User Info detailed use case	30
Figure 4.11 Enter Payment Type detailed use case	31
Figure 4.12 Layered architecture of MFOS	34
Figure 4.13 Database tables of MFOS	35
Figure 4.14 Statechart of MFOS	36
Figure 5.1 Pocket PC 6 Professional emulator	38
Figure 5.2 Device Emulator Manager.....	39
Figure 5.3 Cellular Emulator	40
Figure 5.4 Windows Mobile Device Center	40
Figure 5.5 Welcome screen of MFOS prototype	41
Figure 5.6 Previous Orders Screen	42
Figure 5.7 Choose Food Store screen	43
Figure 5.8 Choose Order Method as well as City and State	44
Figure 5.9 Choose Store Location screen	45
Figure 5.10 Calling Store Location using Phone API.....	46
Figure 5.11 Map of the Store Location.....	47
Figure 5.12 Food Menu screen of one of the food stores	48
Figure 5.13 Special Offers screen.....	49
Figure 5.14 Cart showing user selected food items and price	50
Figure 5.15 Screenshot of the User Information screen of the checkout part.....	51
Figure 5.16 Screenshot of the Payment Type screen.....	52
Figure 5.17 Screenshot of the Order Confirmation screen	53
Figure 5.18 Order Confirmation screen	54

1 INTRODUCTION

We may not realize it fully but technology is making a large impact in the human life, it feels so strange when we hear that it is only 25 years since the first commercial cellular call was made in the United States [1], and today there are more than 262 million wireless subscribers in this country, 83% of the total United States population [1]. With the increasing number of cellular subscribers, the cellular technology is also becoming more and more advanced. It is not too long since the first PDA was launched, Internet connection for mobile devices was made available, and also GPS-enabled PDAs appeared on the market. Now these technologies can be seen very common, and many of us carry them with us.

With increasing popularity and capability of mobile devices, new applications are being launched everyday. Today, many things that we can do using computers we can also do with mobile devices. But simply copying computer applications to mobile devices does not work, as user characteristics of computers and mobiles are quite different. Even though mobility and compactness are major advantages for mobile applications, the small screen, and the small and few buttons of mobile devices are among limitations that make it necessary for mobile application developers to study human-computer interaction for mobile devices and develop their applications accordingly.

Mobile shopping or ordering is one of the most popular areas now-a-days. One can buy songs or any applications right from one's iPhone [2, 3]. One can also bid or buy items from eBay straight away using one's mobile [4]. Similarly, food ordering is also one of the major areas of mobile application development. Some companies have already

launched their food ordering products in the market [5, 6, 7, 8, 9]. But these applications haven't fully exploited the current capabilities of PDAs and missed on some of the required features which would benefit the end users. Thus, for my thesis I have decided to design and develop a Pocket PC application, MobileFoodOrderingSystem (MFOS), using human computer interaction principles for mobile devices.

The main functionality of MFOS is that it lets one order from different food stores. MFOS also gives pickup or delivery options to the end-users depending upon the food store's service offerings. One of the most useful features of MFOS is that it facilitates calling the food store directly if the user has some specific questions. In addition, MFOS also shows map locations of stores if the user needs directions. MFOS also provides some other benefits such as: (a) descriptive menus which display prices of separate food items, (b) saving the user information so he or she will not have to type it again, and (c) the ability of paying through credit card right from his or her Pocket PC.

As they say, time is money! The MFOS software will greatly benefit customers as they don't have to waste time going to the store and waiting in the queue. They will be able to place food orders using their mobile from anywhere even when they are traveling. They will get options to compare prices from different food stores and check out their weekly specials. Stores such as Pizza Hut [10] also have ordering systems like ordering from the Internet, or by phone call. However, not everyone is always in the vicinity of a computer and then it also consumes more time to call an operator and decide on the menu. By using the proposed MFOS application, users can have updated menu on their mobile and they do not need to go for the inconvenient ways of listening to the options on phone, or finding a newspaper flyer.

The remaining chapters of the thesis are structured as follows: Chapter 2 provides a background for the thesis in terms of new technologies in handheld devices, current research in human-computer interaction for mobile applications, and other applications of handheld devices and their usability. Chapter 3 describes in brief the main functions, key usability aspects, and intended users of MFOS. Chapter 4 describes the software model for MFOS. Chapter 5 demonstrates the details of the MFOS prototype. In Chapter 6 MFOS is compared with related applications in terms of functionality and feature offerings. Future work is discussed in Chapter 7, and Chapter 8 concludes the thesis.

2 BACKGROUND

This chapter explores the background areas for designing and developing the MFOS project. The first section explores new technologies in handheld devices such as mobile phones and PDA. In the second section current research in human computer interaction for mobile applications is briefly described. The final section describes few applications of handheld devices on the basis of their usability.

2.1 New Technologies in Handheld Devices

This section gives the background information of new technologies available in handheld devices and recent communication technologies. It also explains new development tools available for building applications for handheld devices.

2.1.1 Handheld Devices

Before developing a mobile application, a thorough research needs to be done on which mobile device(s) one wants to target. As there are so many different mobile devices currently available in the market, each device may differ in screen size, operating system, and/or supported programming language. So studying the currently available handheld devices and their technologies is a must for a developer for figuring out his or her target audience and selecting the appropriate development tool(s).

According to the research in [11], it has been observed that Personal Digital Assistant (PDA) or Smartphone devices are currently owned by only a relatively small percentage of mobile phone users. However, with growing sales, the potential market for PDAs is increasing. The modern mobile phone market offers different devices for a wide

variety of customer tastes and lifestyles. Some phones are small and sleek, and are popular for their ease of carrying, while some are chosen for their appearance so they can become a form of fashion symbols. Some phones are available with alternative covers which allow their appearance to be changed to match the owner's outfit, some phones just offer basic functionality while some devices such as PDAs or Smartphones provide a wide range of business and leisure services to their users. Mobile manufactures are still developing new devices to attract different age groups and cultural groups. These devices will not only look different but also possess capabilities to attract specific groups of people, for example, phone manufactures are trying to portray their phone as a game console to attract the youngsters. To market their products, manufacturers sometimes do not call them mobile, they often use descriptions such as 'game deck', 'communicator' or 'mobile multimedia machine' to better attract their target group of users [11]. Examples of such phones include Nokia N-Gage QD, which is primarily used as a game deck but can also be used as a phone, and BlackBerry, which focuses mainly on business users by providing e-mail communication and productivity applications.

The Smartphones or PDAs attract both business and general users. They attract business users by providing productivity tools such as Word, Excel, E-mail, Organizer and they also include a virtual pop-up QWERTY keyboard for easy typing. They also attract general mobile users with their still camera, video camera, music player, radio, voice recording, games and internet browsing features. iPhone is a very good example of PDA which contains all the above features and attracts basically all groups of people. Hence, considering the promising future of the PDA and Smartphone market, it is beneficial to write applications for PDAs that target large groups of people.

2.1.2 Recent Communication Technologies in Handheld Devices

The GSM Association estimates that GSM (Global System for Mobile communications) has 82% of the global mobile market share among cellular networks [12]. GSM networks operate in four different frequency ranges. Most GSM networks operate in the 900 MHz or 1800 MHz bands. Some countries, including the United States, use the 850 MHz and 1900 MHz bands because the 900 and 1800 MHz frequency bands were already allocated [12]. The key advantage of GSM systems to consumers has been better voice quality and low-cost alternative to making calls, such as the Short Message Service (SMS). It also enables subscribers to use their phones in most parts of the world.

Newer versions of standards for the original GSM phones were also launched, for example the General Packet Radio Service (GPRS) and the Enhanced Data rates for GSM Evolution (EDGE) for higher speed data transmission [13]. GPRS is mainly used for services such as Wireless Application Protocol (WAP) access, Short Message Service (SMS), Multimedia Messaging Service (MMS), and for Internet communication services such as email and World Wide Web access. EDGE can also be used for any packet switching application, such as an Internet connection. But it is mainly used for high-speed data applications such as video services and other multimedia services. Recently 3G services have also been launched for faster data connection. We can see that there is a fast evolution in data connection technology which has greatly benefited the end-users. These fast speed communication technologies have enabled mobile application companies to launch internet applications such as Google Mobile [14], which consumes more network bandwidth and hence needs faster connection.

Other than these fundamental cellular network technologies, mobile phones now come with new communication mediums such as Bluetooth, which makes it possible to transmit signals over short distances between telephones, computers and other devices and thereby simplify communication and synchronization between devices [15]. Using Wi-Fi (802.11) the users can connect to home or office wireless networks for fast internet browsing. The GPS (Global Positioning System) is also another advanced communication technology available in mobile phones that can be used for navigation as well as location specific applications.

2.1.3 Development for Handheld Devices

This section lists the known relative differences between the most popular mobile platform development options for handheld devices such as PDAs and mobile phones.

Java 2 Platform, Micro Edition (J2ME) is a specification of a subset of Java platform aimed at providing a certified collection of Java APIs for the development of software for small, resource-constrained devices such as cell phones and PDAs. Sun provides the Sun Java Wireless Toolkit with which developers can test their applications on an emulator before deploying them on real devices. The application developed for the J2ME platform is generally able to run on most of the java enabled mobile devices.

The Microsoft .NET Compact Framework (.NET CF) is a version of the .NET Framework that is designed to run on Windows-based mobile devices such as PDAs and Smartphones. The .Net Compact Framework applications can be developed on Visual Studio .Net 2003, 2005 or 2008 in Visual Basic or C# languages. As Visual Studio .Net provides a lot of pre-defined controls which makes it a first choice for many for developing mobile applications. One of the limitations of the .NET Compact Framework

applications is that they are restricted to run only on Windows Mobile or Windows CE operating systems.

2.2 Current Research in Human-Computer Interaction for Mobile Applications

The first sub-section describes mobile user characteristics as well as limitations of mobile devices compared to personal computers. Then second sub-section describes how to design mobile applications considering their limitations.

2.2.1 User Characteristics and Limitations of Mobile Devices

Simply transferring a full-sized computer application to the mobile environment almost always results in a suboptimal mobile experience [16]. Hence before starting the development of applications for handheld devices, a developer should consider the mobile user's characteristics as well as the limitations of mobile interfaces.

Some of the Mobile user characteristics discussed by B. Ballard [16] can be outlined as follows:

- Mobile users are mobile, as they users are not sitting attentively at a desk, like desktop or laptop users do.
- Mobile users are interruptible and easily distracted. Mobile users use mobile devices while they are mobile hence they easily can get distracted by the physical world. Sometimes mobile applications can be self distracted by the mobile device itself (e.g. phone) for instance when receiving a call or message.
- Mobile users are always available. Unlike the desktop or laptop computer, the mobile phone is always with the user and always ON (sometimes users can set it

on silent mode, but still most of the time the user has it ready to ring loudly).

- Mobile phones are identifiable. Unlike the desktop computer, the mobile phone is a personal device, with unique identification features.

Similarly, B. Ballard [16] also discusses some of the limitations of the mobile application development which can be briefly described as follows:

1) Interface limitations:

- One cannot take something built for a 17-inch monitor and repurpose it for a Pocket PC device as mobile devices have very small screen resolution (e.g. 352 x 416 for Nokia E60), whereas 17-inch monitors generally have 1280 x 1024 screen resolution.
- Mobile users do not have the facility of mouse for navigation through screens. They have to navigate by using arrow keys, which can be slow and tedious.
- Basic mobile devices have 12-key numeric keypads, whereas Pocket PCs or Smartphones have QWERTY keyboards, which is much more comfortable for the user. In mobile devices, the size of the key is small, the distances between keys are small, and also the device itself is not usually placed on firm surface.

2) Connection limitations:

- As mobile phones are not connected by any wire and mobile phones can be carried to various locations, the quality and strength of the signal cannot be guaranteed.
- The data passed through mobile phones can be easily hacked.
- The cost of receiving and sending data is much higher than for desktop applications.

- The speed of the data transmission is much slower than that of wired connections.

3) Hardware limitations:

- The processor speed and the physical/primary memory of mobile devices are very limited.
- Like the physical/primary memory, the secondary memory of mobile devices is also limited.
- Unlike desktop or laptop computers, mobile phones are not always connected to the power supply. The more the mobile phone functions, the higher is the battery consumption.

2.2.2 Designing Mobile Applications Considering Limitations of Handheld Devices

Many books have been published on techniques for designing mobile applications considering the limitations of handheld devices [16, 17, 18]. The following are some of the techniques discussed by these books [16, 17, 18].

1) Techniques for overcoming interface limitations:

- Store user preferences: as typing on mobile phones is much harder and time consuming than on desktop computers, if user preferences can be stored (e.g., home address) the text entry can be reduced.
- Use input sources from within the mobile instead of asking for manual feeding: specifically GPS for city and state names (location), camera for bar codes, date and time from mobile's clock, address book or calendars. Also, use auto completion, image recognition or speech recognition to reduce text entry.
- One screen at a time: as mobile devices have very small screens, using multiple

windows will be more confusing to the users.

- Give quick response: mobile phones users are impatient hence the application should generate quick responses to the users.
- Small screens also prevent the users from smoothly reading large chunks of text, hence mobile content must be carefully designed for the small screen and must also take into consideration lack of user focus.
- Although mobile devices can certainly be used with two hands, they will frequently be used with one hand. Therefore, create mobile applications that can be used with one hand only.

2) Techniques for connection limitations:

- Ensure that the data is available when the user asks for it – whether the network is available or not. A solution for this is pre-fetching data.
- Application support to run in offline mode.
- Encrypt data before sending it on network for security purposes.

3) Techniques for hardware limitations:

- Choose the best libraries that support required application features as the minimum hardware cost is essential.
- Optimize the packaging process. Even after carefully choosing the best lightweight library, one may still find that the application utilizes only part of the library. In the packaging process, one should include only the classes one actually uses.
- Partition the application. Since the MIDP runtime loads classes only as needed, one can partition the application into separate components to reduce the runtime

- footprint.
- Carefully examine design patterns in the early stages of the development cycle.
 - Concisely reuse existing objects at the implementation level.
 - Close network connections and file handlers and store the Record Management System (RMS) record quickly after use.
 - Battery consumption should be reduced through shorter connection time, not waking the display when unnecessary, and decreasing processor demands.

2.3 Handheld Device Applications

This section describes several existing mobile ordering applications and provides some observations on their design criteria to fit into the mobile application category. To begin with, Amazon Mobile, a mobile application for purchasing items from amazon.com website is described, then ETRADE Mobile, a mobile application for purchasing stocks is briefly outlined, and lastly, CityMint, a food ordering system application, is explained.

2.3.1 Amazon Mobile

If one wants to search the price of some books or if one is in a store and would like to check the price of some item on amazon.com, personal computers will not always be available for one's use, but one's mobile will. To support customers in such situations, amazon.com launched Amazon Mobile, a mobile application for iPhone or iPod Touch [19].

It can be observed that the Amazon has certainly considered mobile user characteristics while developing Amazon Mobile. Mobile 1-Click [19] is one of the most useful features of Amazon Mobile. Typing on a mobile device is not as easy as on a

personal computer. Once one decides to buy an item from Amazon Mobile typing in all user and payment information can become a tedious task. So using Mobile 1-Click, the user can configure all his or her payment details for one time and then using just 1-Click he or she can checkout the purchased item(s). Similarly, Amazon Remembers is also another nice feature considering mobile usability. If one likes some item and wishes to purchase it later, it becomes difficult to remember its details. However with the Amazon Remembers one can just take a picture of that item and the program will store it so one can refer to it anytime later.

2.3.2 ETRADE Mobile

Buying or selling stocks at a certain price can be very important to the group of people who follow market trends. ETRADE provides ETRADE Mobile [20] to their customers to check their stocks prices or even trade them from wherever they are and whenever they want. Using ETRADE Mobile, the users can watch real-time streaming of quotes, view their accounts, create their own watch-list, and create the rules for trading their stock right away from their mobile devices.

ETRADE Mobile developers knew their target audience will be business users, so they set their target platform as Blackberry, the leading mobile device for business users. To reduce time in searching users stocks, ETRADE Mobile offers the facility to their users to create a watch-list for their stocks so they can check their stocks with few clicks. ETRADE Mobile provides a Complete Protection Guarantee to their customers for all transactions made from ETRADE Mobile so as to build confidence in mobile trading [21].

2.3.3 CityMint

CityMint [5] is a tool that has similarities with MFOS. Using CityMint users can browse different food store menus, order multiple items in single order, and check the total. CityMint lets the users pay right away from their phones using a credit card and also let their users save their favorite food items [22].

The CityMint team created their application considering their users' characteristics and needs. Considering the fact that people might want to try new food items every time they order their food, CityMint provides food menus of all participating food stores. To provide their customers with an easy way of payment, CityMint lets them pay right away from their phone. For those who would like to save their favorite food items, the software also lets them save these items as Favorites.

3 MFOS: AN OVERVIEW

This chapter describes the main functions of the proposed MFOS approach. This chapter also explains key usability aspects considering human-computer interaction principles for mobile devices and at the end it indicates the intended users of MFOS.

3.1 Main functions of MFOS

MFOS is intended to give Pocket PC users a rich and efficient GUI experience for ordering food. This section explains how this tool is useful to Pocket PC users, and describes some important functions of MFOS that makes this tool unique.

Generally, if a customer has to order food for delivery, he or she calls the food store and asks about the options he or she can have or search for paper flyers in the mail, decides what he or she wants, and then places the order. This is a long process and not as efficient as well because by asking the store person he or she may not get the idea about the entire menu available. Alternatively, the customer needs to store all store flyers which might have outdated specials. MFOS users do not have to call anybody for asking about the menu or specials and it is even not required to store the paper flyers for the stores anymore because MFOS shows detailed menu displaying the price as well as the current specials of the food stores.

MFOS is also useful for order pickups. If the user wants to take something for pickup, he or she would have to go to the food store and wait for his or her turn to place the order and then wait until the food gets prepared. If the user orders it using MFOS before leaving home then until he or she reaches the food store, the food is ready for

pick-up. This not only saves user time but also achieves user satisfaction which is of prime importance for any food store.

MFOS will result in a really useful tool when users are outside the home and want to order food for pickup or delivery, where they neither have the food store's phone number nor their menu. By using MFOS on their mobile phones, which they always carry, the users check the menu and order their favorite food from practically anywhere.

MFOS gives the users the option to browse the menu and the specials of all different cuisines so that they can try new things and/or get good deals. MFOS also shows the map if the user is unaware of the store location from where he or she wants to pickup his or her order. The user can also call the store if he or she has any questions like until what time store remains open or what is the expected time for food to be ready.

Using MFOS the users can also pay for food right from their phones using a credit or a debit card. The option to pay by cash at the time of delivery or pickup is also available.

MFOS also lets the users browse their last order(s) in case they would like to place the same order or check the total and other details of previous orders.

3.2 Key Usability Aspects of MFOS

MFOS has been designed and developed considering human-computer interaction techniques for mobile phone applications, as discussed in Section 2.2.2. It has been decided that MFOS should have minimum long forms, so the users will not have to use scrollbars. On the screens of MFOS precise, concise, and readable data should be present so it will be easier for the user to read and choose amongst them. MFOS should also give quick and clear responses so the users get the feedback of what's happening in the

application. The user interface of the MFOS should be very simple and attractive in appearance and should be intuitive to the user. MFOS should have clear instructions so any novice user can use it efficiently.

3.3 Intended Users of MFOS

MFOS is developed for Pocket PC 6 owners who would like to order food frequently from food stores. In the future, this application will be available to the users of other mobile devices. As the user interface of this application is simple, prior training of this application is not required. Basically, anyone who can use a PDA can use this application. Currently this application only supports the English language, but in the future the application can be extended and made available for other languages. Also, at this time, this application is intended to work for a few selected cities in US.

4 MFOS: SOFTWARE MODEL

This chapter describes the main components of the UML software model of MFOS. This includes the requirement specification for the system (both functional and non-functional requirements), the system's UML use case diagram, use cases, layered architecture, database table diagram, and state-chart diagram [23, 24, 25]. In addition to the classic UML specifications, a non-standard UML software modeling construct which is essentially a traceability matrix is also included.

4.1 Requirement Specification

This section describes functional and non-functional requirements of MFOS. Functional requirements define functions of a software system, i.e. how a software system should take a set of inputs, software system's behavior, and a set of outputs that a software system should produce.

Non-functional requirements represent various types of constraints placed on the software. Such constraints include technological constraints (e.g. target operating system of mobile device, development technology) and developer constraints such as process lifecycle used, implementation language, etc ^[26].

4.1.1 Functional Requirements

Table 4.1 describes the functional requirements of the MFOS prototype. These are just the requirements for the MFOS prototype. After evaluating the MFOS prototype, the functional requirements for next release of the MFOS application will be added to the list.

Table 4.1 Functional Requirements

Ref. #	Functional Requirement
FR1	The MFOS shall provide multiple food stores for a user's selection.
FR2	The MFOS shall provide option of pickup or delivery for a user's order.
FR3	The MFOS shall provide a list of cities where MFOS service is available.
FR4	The MFOS shall display multiple available store locations of selected food store for user's selection, when the selected order method is pick up.
FR5	The MFOS shall show the map of the store location.
FR6	Using MFOS the user shall be able to call the food store location.
FR7	The MFOS shall display the prices of different food items.
FR8	Using MFOS the user shall be able to add multiple food items to the cart in the same order.
FR9	Using MFOS the user shall be able to zoom in and zoom out the map.
FR10	Using MFOS the user shall be able to add the same food item more than once.
FR11	The MFOS shall display the weekly specials of the selected food store.
FR12	Using MFOS the user shall be able to see the cart anytime from both the food item and the specials screen.
FR13	The MFOS shall give confirmation message for addition or removal of the food item from cart.
FR14	The MFOS shall provide the option of saving the user's info as well as option to use the already saved user info.
FR15	The MFOS shall give payment options.
FR16	The MFOS shall be able to view previously placed orders.

4.1.2 Non-Functional Requirements

Table 4.2 provides the non-functional requirements of MFOS. After thorough background study and research the following non-functional requirements were defined.

Table 4.2 Non-Functional Requirements

Ref. #	Non-Functional Requirement
NFR1	The MFOS shall be able to work on Pocket PC 6 Professional.
NFR2	The MFOS shall be developed in the C# language using Visual Studio 2008 IDE.
NFR3	The MFOS shall store important information of software using the SQL Server CE database engine.
NFR4	The SQL Server CE database of MFOS shall sync to SQL Server, before the application starts as well as after the order is placed.
NFR5	The MFOS shall use the Google Static Maps Web Service for showing maps in the application.
NFR6	The MFOS shall use Phone API for using host phone functions like calling a number.
NFR7	The MFOS interface shall be simple and easy to learn.

4.2 Use Case Diagram

This section describes the use case diagram of MFOS. A use case defines the relationships between the user and the system. The MFOS system is represented by a bounded box labeled with the name of the system. The MFOS user is represented by an external actor, who interacts with the use cases inside the system boundary. Figure 4.1

shows the use case diagram of MFOS with one actor, i.e. the MFOS user, and 10 use cases. These use cases represent the core functionality of MFOS.

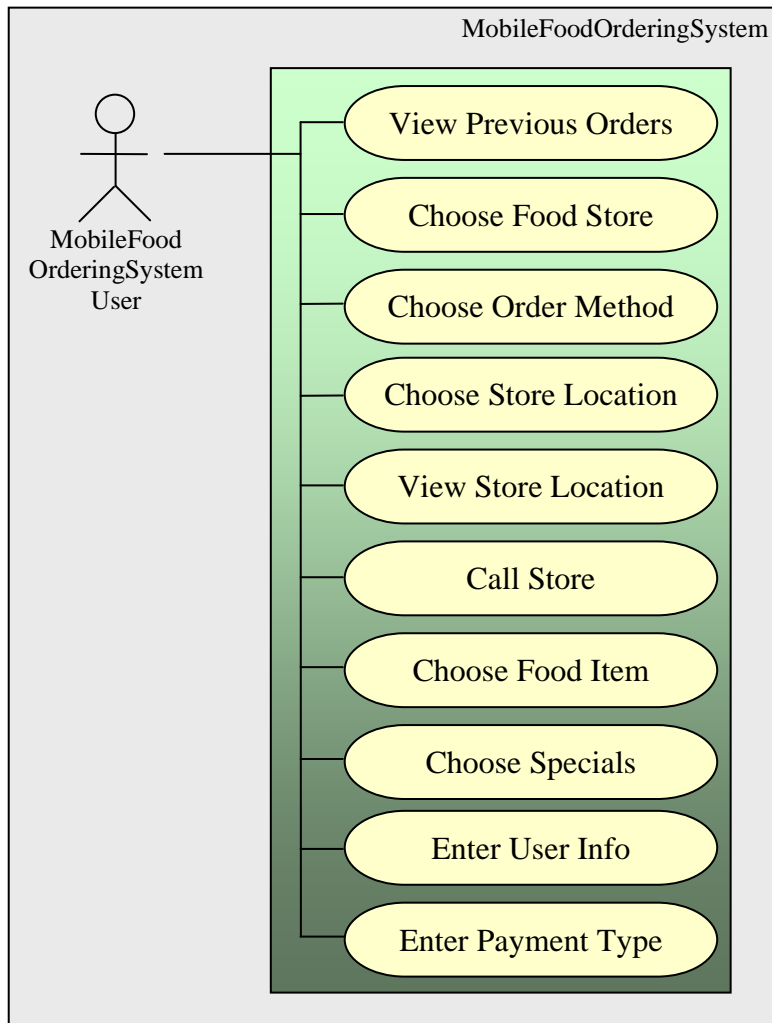


Figure 4.1 Use Case Diagram of MFOS

4.3 Detailed Use Cases

This section describes in detail all use cases shown in Figure 4.1. While explaining each use case, actors involved in it, preconditions, flow of events, post conditions, alternative flows and their post conditions are also described.

The first detailed use case is “View Previous Orders”, which is explained in Figure 4.2. This use case describes detailed information of previously placed orders by user.

Use Case: View Previous Orders
Actors: MFOS User
Preconditions: MFOS is running and the Welcome screen is displayed.
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. The use case starts when the user taps on the Previous Orders button to view his or her previously placed orders. 2. The system displays all orders placed by the user starting with the latest order. 3. In the description, it shows when did the user place the order, from which food store he or she placed the order, what did he or she order, what was the total price of the order, was it delivery or pickup order and how did he or she pay for the order. 4. When user taps on the Back button, the Welcome screen will be displayed.
Post conditions: The system takes the user to the welcome screen if he or she wants to place the new order.

Figure 4.2 View Previous Orders detailed use case

The second detailed use case is “Choose Food Store”, shown in Figure 4.3. In this use case the user makes his or her selection of food store to order the food.

Use Case: Choose Food Store
Actors: MFOS User
Preconditions: MFOS is running and displays the Welcome screen.
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. The use case starts when the user taps on the Start button to start placing new order. 2. The system fetches participated food stores from the database and displays in the list. 3. When the user selects a food store, the logo of the food store will be displayed at the bottom of the screen. 4. When the user taps on the Next button, the Delivery/Pickup screen will be displayed.
Post conditions: The system takes the user to the Delivery/Pickup screen where he or she has to choose the order method.
Alternative flow 1: At any time the user can go back to the Welcome screen by tapping on the Back button.
Post conditions: The system takes the user to the Welcome screen.

Figure 4.3 Choose Food Store detailed use case

The third detailed use case is “Choose Order Method”, displayed in Figure 4.4. In this use case, the user makes his or her selection of order method of either delivery or pickup, as well as his or her city and state.

Use Case: Choose Order Method
Actors: MFOS User
Preconditions: User selected the food store to order the food.
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. The use case starts when the user selects the food store to order the food. 2. The system gives option of delivery or pickup for his or her order. 3. On the same screen the user has to select his or her city and state. 4. When the user taps on the Next button according to his or her selection of delivery or pickup, the next screen will be displayed.
Post conditions: The system takes the user to the Choose Store Location screen if he or she selected pickup but if he or she selected delivery as his or her order method, the Food Menu screen will be displayed.
Alternative flow 1: At any time the user can go back to the previous screen by tapping on the Back button.
Post conditions: The system takes the user to the Choose Food Store screen.

Figure 4.4 Choose Order Method detailed use case

“Choose Store Location” is the fourth detailed use case, as displayed in Figure 4.5. This use case is only applicable for order pickups. Here the user has to make the selection of the store location which will be convenient for him or her to pickup the order.

Use Case: Choose Store Location
Actors: MFOS User
Preconditions: User selects pickup as his or her order method on the Delivery/Pickup screen.
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. The use case only starts if the user selects the pickup as his or her order method from the Delivery/Pickup screen. 2. The system shows all locations of selected food store in the user's city. 3. The user can see the location of the map to get better understanding of the location of the food store for pickup or the user can also call the store to ask questions, if any, before placing the order from the system. 4. When the user taps on the Next button the Food Menu of the selected food store will be displayed.
Post conditions: The system takes the user to the Choose Food Menu screen.
Alternative flow 1: At any time the user can go back to the previous screen by tapping on the Back button.
Post conditions: The system takes the user to the Choose Food Store screen.
Alternative flow 2: If the user taps on the Show Map button, the map of the store location will be displayed.
Post conditions: The system takes the user to the Store Location Map screen with the default zoom and red pin pointing to the location of the store.
Alternative flow 3: If the user taps on the Call button the system will call the store location number.
Post conditions: Using the Phone API, the system calls the store location. The calling screen of the phone will be displayed.

Figure 4.5 Choose Store Location detailed use case

“View Store Location” is the fifth use case, as shown in detail in Figure. 4.6. This use case displays the map of the selected store.

Use Case: View Store Location
Actors: MFOS User
Preconditions: The user selects the store and taps on the Show Map button.
Flow of Events: <ol style="list-style-type: none"> 1. The use case starts when the user selects the store location and to see the map of the store taps on the Show Map button. 2. Using Google Static Maps API, the map of the store is displayed to the user. 3. The store location on the map is indicated by a red pin. 4. The user can zoom in and zoom out to get a better picture of the store location. 5. When the user wants to go back to the Store Selection screen the user can tap on the Back button.
Post conditions: The system takes the user back to the Store Selection screen.

Figure 4.6 View Store Location detailed use case

“Call Store” is the sixth use case, which is detailed in Figure 4.7. By using this use case, the user can make a call to the store if he or she has some specific question about his or her order.

Use Case: Call Store
Actors: MFOS User
Preconditions: User selects the store and taps on the Call button.
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. The use case starts when the user selects the store location and to ask some specific question/s he or she taps on the Call button. 2. Using Phone API basic function of phones (Calling, SMS, etc.) can be accessed. 3. Using functions of this Phone API and fetched phone number of the selected store, the system calls that number. 4. When the user finishes his or her call he or she can go back to the Store Selection screen without losing his or her state of the application.
Post conditions: When the user exits from the phone mode he or she will automatically be forwarded to the Store Selection screen where he or she left the application before the Call Store screen.

Figure 4.7 Call Store detailed use case

The seventh and eighth use cases “Choose Food Item” and “Choose Specials” are displayed in detail in Figure 4.8 and Figure 4.9, respectively. The “Choose Food Item” use case will let the user add food items from the selected food store for placing the order. In this use case the user will see detailed description of the food items, including their individual prices. Similarly, using the “Choose Specials” use case, the user can see detailed description of weekly specials of the selected food store with their individual prices. This use case also gives a confirmation message when the user adds the food item to the cart.

Use Case: Choose Food Item
Actors: MFOS User
Preconditions: The user selects delivery as his or her order method on the Delivery/Pickup screen. This screen also shows when the user selects the store location for the pickup order method.
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. The use case can be triggered in two cases, either when the user selects the delivery as his or her order method from the Delivery/Pickup screen or if he or she selects the store location for his or her pickup order method. 2. The system shows all the food items from the selected food store. 3. Food items in this list contain the prices of each individual item. 4. The user can add the food item to the cart, or can even see the specials by tapping respective buttons. 5. When the user taps on the View Cart button, the selected food item will be displayed in the cart.
Post conditions: The system takes the user to the View Cart screen.
Alternative flow 1: At any time the user can go back to the previous screen by tapping on the Back button.
Post conditions: The system takes the user to the previous screen.
Alternative flow 2: If the user taps on the Add to Cart button, the selected item will be added to the cart.
Post conditions: The same screen, Select Food Item, will be shown.
Alternative flow 3: If the user taps on the View Specials button, the weekly specials of the selected food store will be displayed.
Post conditions: The Specials screen will be displayed.

Figure 4.8 Choose Food Item detailed use case

Use Case: Choose Specials
Actors: MFOS User
Preconditions: When the user taps on the View Specials button from the Choose Food Item screen, this use case will be triggered.
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. While selecting food item if the user taps on the View Specials button this use case will start. 2. The system shows the user all the weekly specials of the selected food store. 3. Specials in this list also contain the price of each item. 4. The user can add the food item to the cart by tapping on the 'Add to Cart' button. 5. When user taps on the 'View Cart' button the food items added to the cart will be displayed.
Post conditions: The system takes the user to the View Cart screen.
Alternative flow 1: At any time the user can go back to the previous screen by tapping on the Back button.
Post conditions: The system takes the user to the previous screen.
Alternative flow 2: If the user taps on the Add to Cart button, the selected item will be added to the cart.
Post conditions: The same screen, Select Specials, will be shown.

Figure 4.9 Choose Specials detailed use case

“Enter User Info” is the ninth use case of MFOS, displayed in Figure 4.10. In this use case the user adds his or her personal information for checkout. He or she can also use the previously saved user information.

Use Case: Enter User Info
Actors: MFOS User
Preconditions: This will start when the user decides to go for checkout.
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. The use case starts when the user selects the Checkout option of MFOS. 2. This use case is the first part of the checkout process. 3. The user needs to enter his or her name, phone number and address; the user can also use his or her already saved user info by selecting it from a combo box. 4. When the user taps on the Next button, the Payment Type screen will be displayed.
Post conditions: The system takes the user to the Payment Type screen, which is the beginning of the second part of the checkout process.
Alternative flow 1: At any time the user can go back to the previous screen by tapping on the Back button.
Post conditions: The system takes the user to the View Cart screen.

Figure 4.10 Enter User Info detailed use case

The last use case of MFOS is “Enter Payment Type”, described in detail in Figure 4.11. This use case handles the different payment options the user can select for his or her order.

Use Case: Enter Payment Type
Actors: MFOS User
Preconditions: This use case will come in action after the “Enter User Info” use case in the checkout process.
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. The use case starts after the user entered his or her information in the User Info screen. 2. In this use case the user has to select the payment type: either cash, credit, or debit card. 3. If the user selected credit or debit card as payment type, he or she needs to input his or her credit/debit card number, expiry date, and security code. 4. This is the last step in configuring the user order; after this the user just has to confirm the order.
Post conditions: The system takes the user to the Order Confirm screen where he or she needs to confirm his or her order details.
Alternative flow 1: At any time the user can go back to the previous screen by tapping on the Back button.
Post conditions: The system takes the user to the Enter User info screen.

Figure 4.11 Enter Payment Type detailed use case

4.4 Traceability matrix

Traceability matrices are not a standard part of UML, but they are useful in the management of functional requirements [24]. Table 4.3 shows which requirements are fulfilled, and by which use cases.

Table 4.3 Requirement Traceability Matrix

R/UC	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10
FR1		✓								
FR2			✓							
FR3			✓							
FR4				✓						
FR5					✓					
FR6						✓				
FR7							✓			
FR8							✓			
FR9					✓					
FR10							✓			
FR11								✓		
FR12							✓	✓		
FR13							✓	✓		
FR14									✓	
FR15										✓
FR16	✓									

4.5 Layered Architecture

Figure 4.12 shows the layered architecture of MFOS. It depicts the MFOS user interacting with the C# .NET Compact Framework forms on the mobile device. As displayed in Figure 4.12 this architecture has three layers: the Presentation Layer, the Business Layer, and the Data Access Layer. The Presentation Layer consists of C# .NET Compact Framework forms. The Business Layer consists of a set of objects that contain the business and application logic [27]. The bottom layer consists of three parts: the Phone API, the Service Gateways, and the Data Access Layer. Phone API consists of functions with which native phone functions can be called, such as SMS, calling a number, etc. Using the Service Gateways, web services such as Google Static Maps API

can be used. The Data Access Layer consists of the data access objects and the database itself [27]. MFOS uses a local database, managed by the SQL Server CE to temporarily store data on a mobile device. In the future, it will communicate with a larger SQL Server database on a web server through the Internet or on a PC through active synchronization.

4.6 Database Tables of MFOS

MFOS has five database tables, based on which the required information can be queried, the previous information can be updated, and the new information can be saved. Figure 4.13 shows the database tables in MFOS. The description of these database tables is as follows:

- **FoodStore:** FoodStoreID is the primary key of this database table. It stores the food store names.
- **StoreLocation:** StoreLocationID is the primary key of this database table. It stores the details of the store locations of all food stores. FoodStoreID is the foreign key.
- **FoodMenu:** FoodMenuID is the primary key of this database table. It stores the information about all food items and their details. FoodStoreID is the foreign key.
- **UserOrder:** UserOrderID is the primary key of this database table. It stores all the details of the order(s) placed by the user.
- **UserInfo:** UserInfoID is the primary key of this database table. It stores the user information saved by the user.

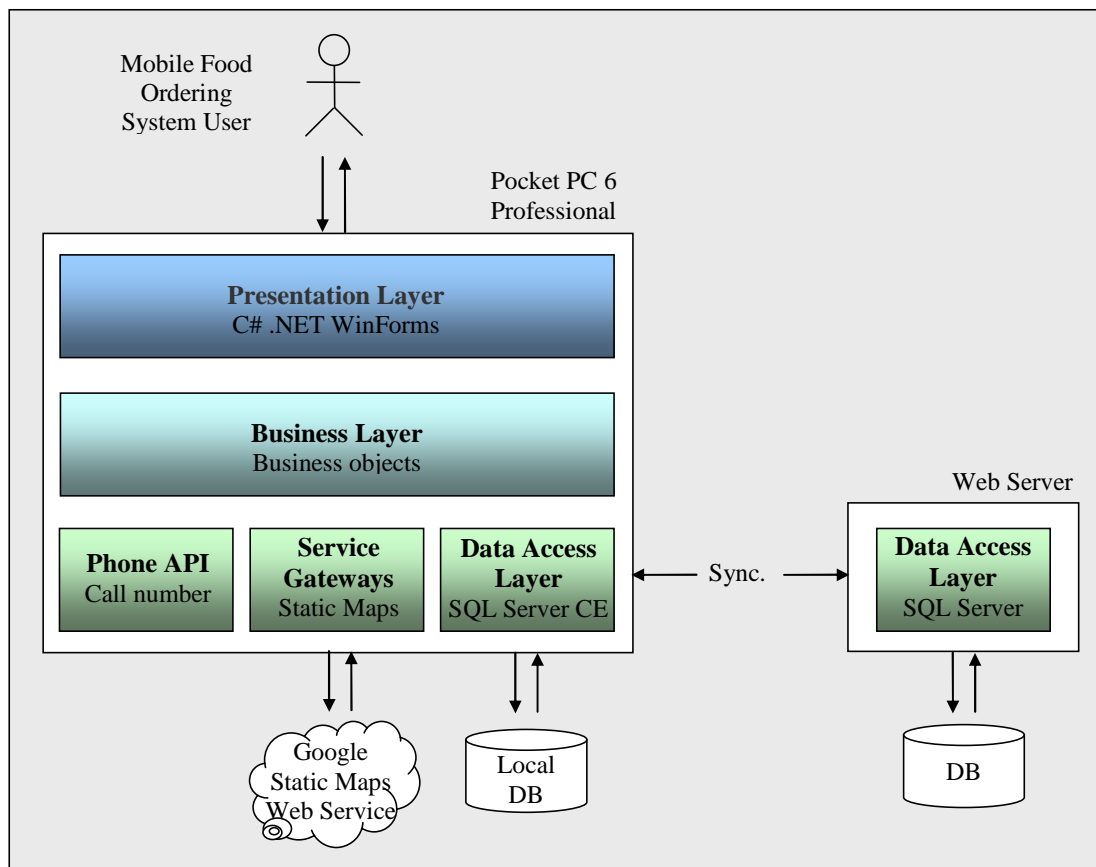


Figure 4.12 Layered architecture of MFOS

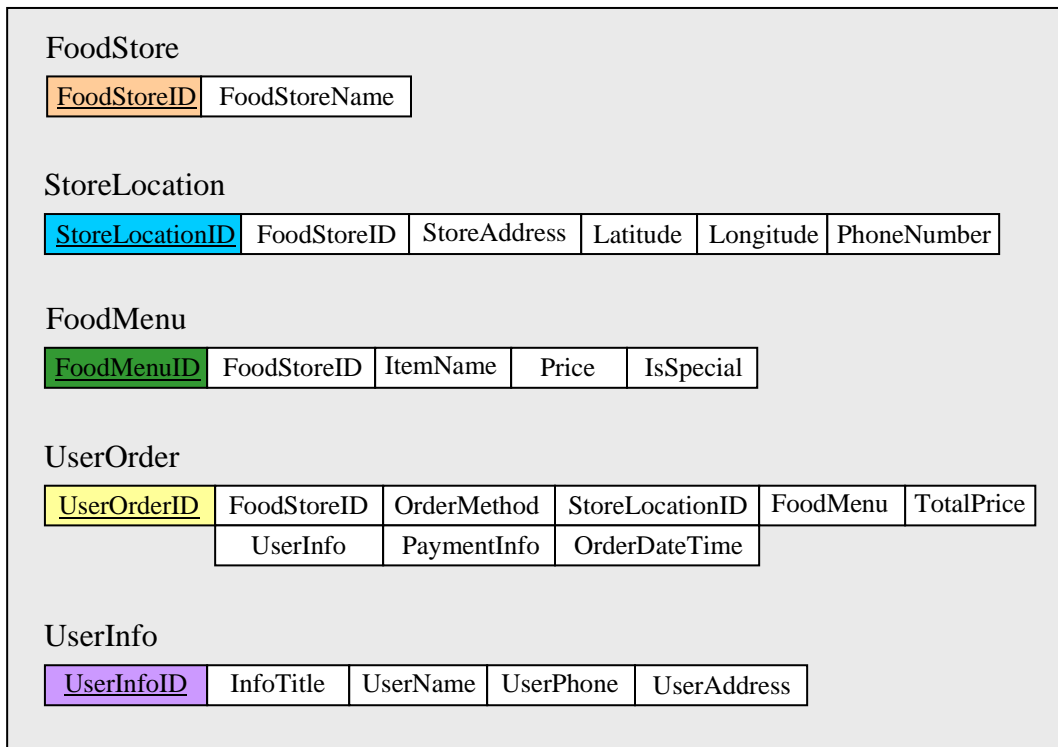


Figure 4.13 Database tables of MFOS

4.7 Statechart of MFOS

Figure 4.14 depicts the state transitions during MFOS execution. It describes the basic behavior of the system in response to external stimuli.

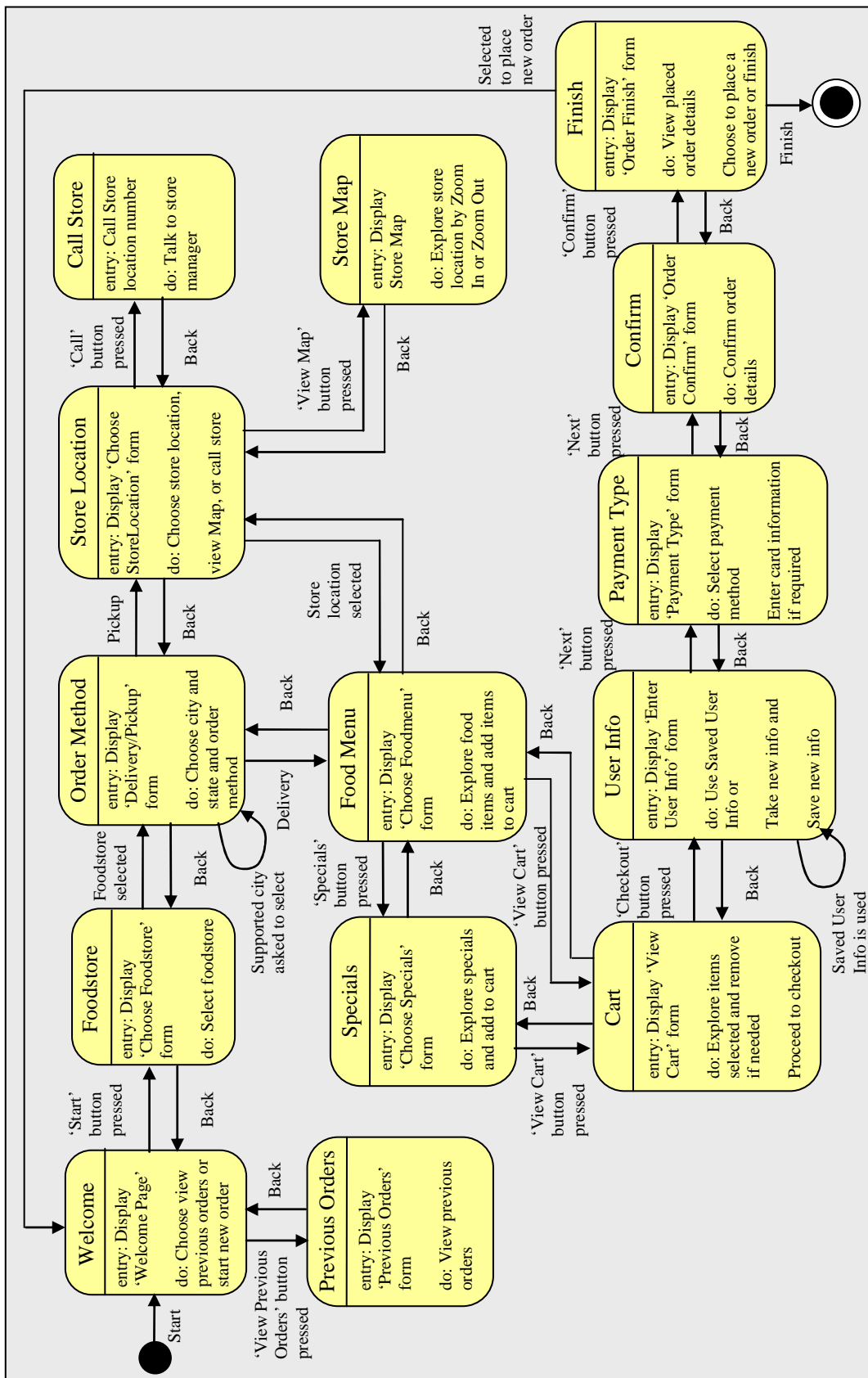


Figure 4.14 Statechart of MFOS

5 PROTOTYPE DETAILS

This chapter provides details of the MFOS prototype. To explain the technology, functionality and usability of MFOS, this chapter is divided into three sections. The first section describes the components used to generate the platform for deploying the MFOS prototype. The second section explains the workflow of the project with a brief explanation of the screens of the MFOS prototype. Finally, the third section discusses the usability of components used by the MFOS prototype.

5.1 MFOS Platform

This section explains the different components used to generate the platform for deploying the MFOS prototype. The MFOS prototype was developed using the Visual Studio 2008 IDE. The C# development language was used to write the prototype's code. The target platform of the developed prototype is Pocket PC running Windows Mobile 6 Professional or higher. However, with small changes in the code MFOS can be used in Pocket PC 5 as well as Pocket PC 2003 as MFOS does not use any special libraries which are not available in Pocket PC 5 or Pocket PC 2003.

For development and testing purposes, the Pocket PC 6 Professional emulator that comes with the Windows Mobile 6 Professional SDK [28] was used. Figure 5.1 shows the screenshot of the Pocket PC 6 Professional emulator which can be connected through the Device Emulator Manager shown in Figure. 5.2 (the latter being part of the Visual Studio 2008). The Device Emulator Manager can be used to connect to all the Pocket PC and Smartphone emulators whose SDKs are installed on the computer. The Device Emulator Manager can also be used for different variants of the same Windows Mobile

OS, as seen in Figure 5.2, which depicts five different Windows Mobile 6 Professional emulators with different hardware capabilities.



Figure 5.1 Pocket PC 6 Professional emulator



Figure 5.2 Device Emulator Manager

The emulator that can be seen in Figure 5.1 is sufficient for testing applications which do not use phone or internet connections. MFOS uses the PhoneAPI to provide the option to call the store directly from the application, as well as the Internet connection to call the map web service from the application. The Cellular Emulator, shown in Figure 5.3, facilitates a fake radio interface layer for Windows Mobile 6 Professional Emulator. The Windows Mobile Device Center, shown in Figure 5.4, brings in the data connection to Windows Mobile 6 Professional Emulator. The Cellular Emulator simulates a Radio Interface Layer to test applications that use Phone API for providing SMS capabilities and calling a number [29]. Similarly the Windows Mobile Device Center simulates the data connection in the emulator by using the computer internet connection to test applications that use internet to access web services [30].

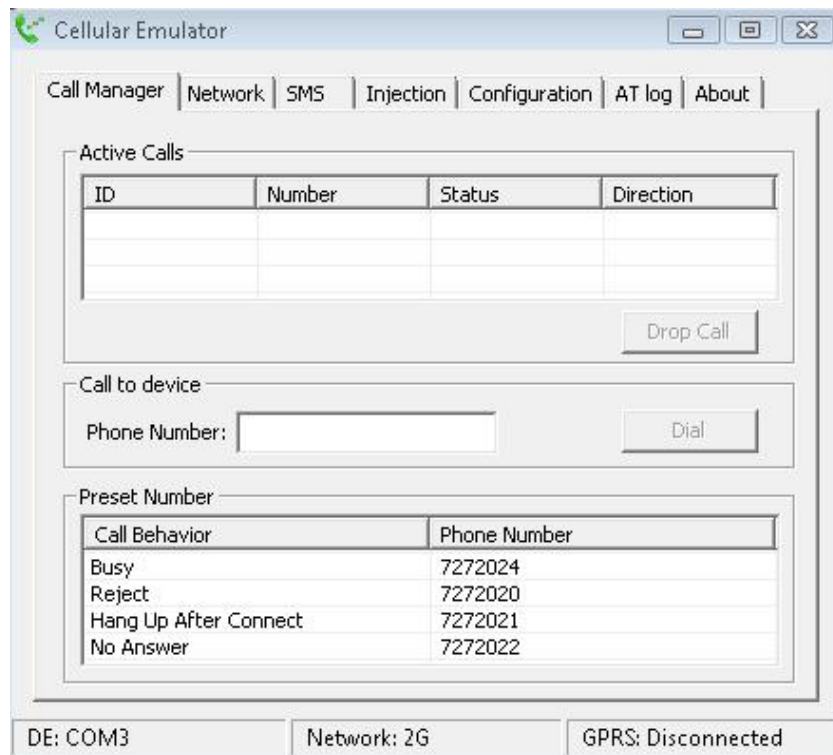


Figure 5.3 Cellular Emulator



Figure 5.4 Windows Mobile Device Center

5.2 MFOS Prototype

This section describes the MFOS prototype functionality with the help of screenshots and brief explanations. Figure 5.5 presents the Welcome screen of the MFOS prototype where it loads resources such as the connection to the MFOS database and checks internet connection.



Figure 5.5 Welcome screen of MFOS prototype

On the welcome screen if one taps on the Previous Orders button MFOS will show one all one's previous order details starting with the most recent order at the top, including: date and time when one placed the order, food store for which one placed the

order, what did one order, what was the total price of the order, was it delivery or pickup order, and how did one pay for the order.

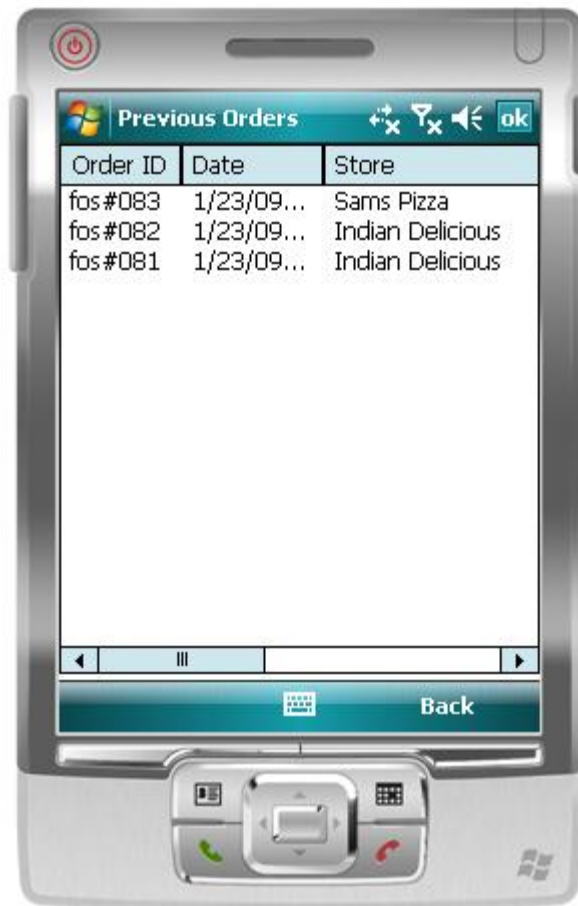


Figure 5.6 Previous Orders Screen

If the Start button is tapped on the Welcome screen, the order placing process begins, where first the user is directed to choose the food store. Figure 5.7 shows the Choose Food Store screen where the user will see all the food stores available in the system to choose from. When he or she selects any food store, the store logo is displayed to improve the look and feel of the prototype as well as for the ease of the user to visually confirm what he or she selected. For example, as shown in Figure 5.7, when the user tapped on the Indian Delicious food store, this store's logo is displayed.



Figure 5.7 Choose Food Store screen

Figure 5.8 shows the Choose Order Method screen where the user will be asked to choose the order method, either pickup or delivery. According to the user's selection, the next options will be shown, e.g. if the user chooses pickup he or she will be asked to choose the store location from where he or she wants to pick up his or her order, whereas the user will not receive this screen if he or she chooses delivery. The user will also be directed to choose the city and state on the same screen. The city and state information narrows the store locations according to the user's choice.



Figure 5.8 Choose Order Method as well as City and State

If the user chooses the Carryout option, he or she will be asked to choose the store location from where he or she wants to pickup the order. All available store locations in the city and state selected for the selected food store will be shown in the list view. If the user wants to call the store for more information such as store hours or food preparation time, or if the user wants to see the location of the store on the map, he or she can click on the Call or Show Map buttons, respectively. The Choose Store Location screen is shown in Figure 5.9.



Figure 5.9 Choose Store Location screen

If the user wants to call the store from the Choose Store Location screen for over the phone further queries, the application will look up for the phone number of the store from database and by using the Phone API application it will make a call to the store. After the call finishes the user will automatically get back to the page where he or she left so he or she will not lose any of his or her work. Figure 5.10 shows the calling screen from the Phone API.



Figure 5.10 Calling Store Location using Phone API

The user can also see the map of the store by using the Show Map button on the Choose Store Location screen. As the application is already connected to the internet, it will get the map using the Google map web service, which indicates the store location on the map with a red pin. The user can also zoom-in and zoom-out to get a better idea of the store location. Figure 5.11 shows one of the store locations on the map.

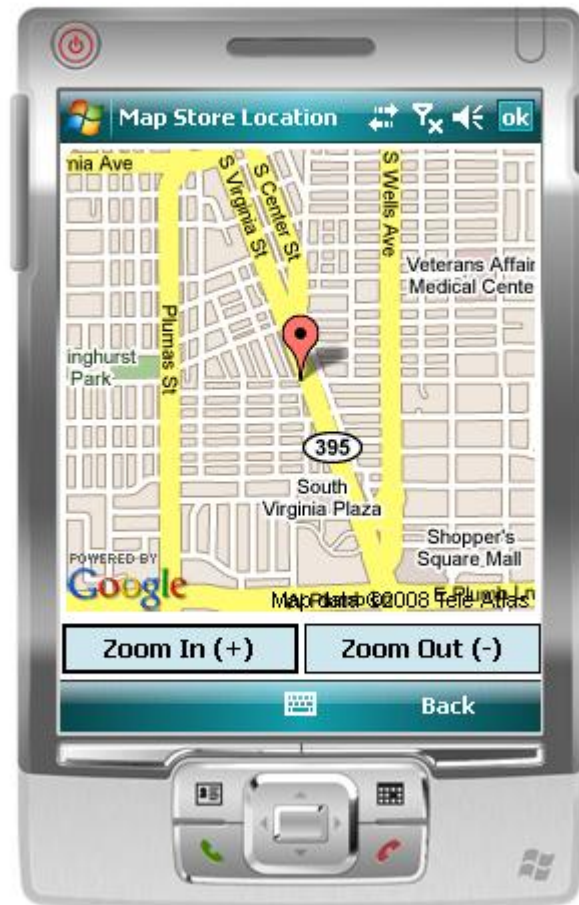


Figure 5.11 Map of the Store Location

After selecting the store location for pickup from the Choose Store Location screen or directly after choosing the Delivery option, the food menu will be shown to let the user select his or her favorite food for order. In the Food Menu screen the user will see all the food items that the food store offers. Next to the name of the item the user will also see the price of the food item. To add a food item to the cart, the user will just click on the food item and then taps on the Add to Cart button. A confirmation message that “the item is added to the cart” will also be shown. From the same screen the user can go to the cart for checkout or he or she can check food store specials. Figure 5.12 shows some of the food items in the food menu screen of the Indian food store.

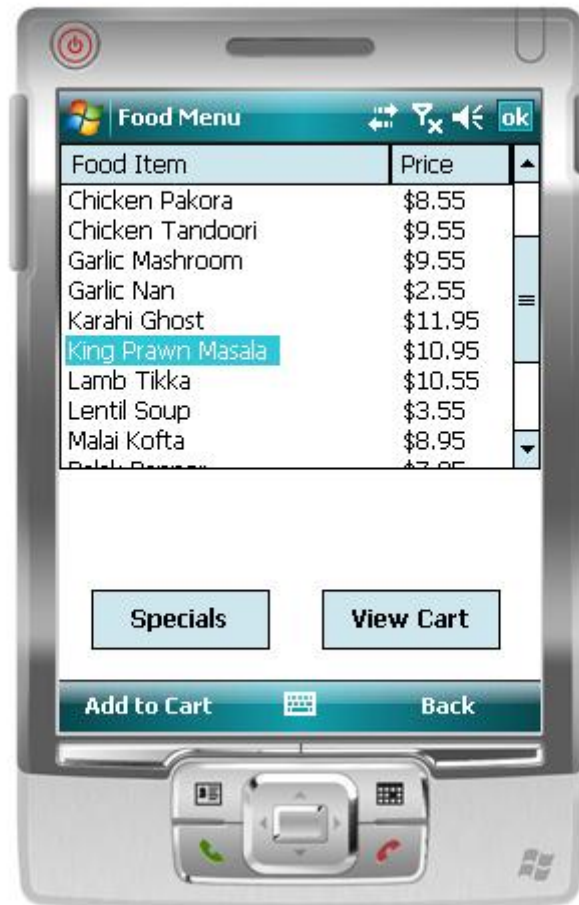


Figure 5.12 Food Menu screen of one of the food stores

Figure 5.13 shows the screenshot of the special offers offered by the food store. The user can check these offers by tapping the Specials button on the Food Menu screen. Specials are generally combo pack savings to attract the customers. Like the Food Menu screen, the user can see the price of the specials and also add the items to the cart by tapping on the Add to Cart button. For going to the cart for checkout the user does not have to go to the Food Menu screen again, he or she can directly go to the cart by tapping the View Cart button.



Figure 5.13 Special Offers screen

When the user finished adding food items as well as specials to the cart he or she can go to the checkout by tapping the View Cart button. Here the user will see his or her selected order and their individual prices as well as the cumulative price. The user can also remove food items by tapping on the Remove Item button. After removing the item and passing the confirmation message, the user can see the modified cart. The user can also go back and add food items by tapping on the Back button. Once the user is sure about the food items in his or her order and wants to proceed for checkout he or she can click on the Checkout button. Figure 5.14 shows the screenshot of the example cart.



Figure 5.14 Cart showing user selected food items and price

The checkout process has two steps, in the first step the user will be asked to fill his or her personal information viz. name, phone number, and address. In this process the user can use previously saved information as generally the mobile device is personal and personal information does not generally change. This will ease the user from typing the same information every time he or she places the order. If he or she wants to change some information such as address, he or she can change it and save it under the same user profile or he or she can save it as a new entry. Figure 5.15 shows an example of this screen.



Figure 5.15 Screenshot of the User Information screen of the checkout part

After entering the user information, now the user has to choose the payment method as the second part of the checkout process. In the payment type, the user has three options. He or she can pay by credit card, debit card, or cash at the time of delivery or pickup. If the user chooses credit card or debit card as the payment type he or she has to enter the credit card number, expiry date, as well as the security code. We want to secure the credit or debit card information of the user and ensure it is not misused, so the user is always prompted to enter his or her credit/debit card info. Figure 5.16 shows the screenshot of the Payment Type screen.



Figure 5.16 Screenshot of the Payment Type screen

After entering the order and the user information, the user will finally be asked to confirm the name, the address, and the food items at the Order Confirmation screen. Once the user confirmed the order, he or she will be charged on his or her card if he or she chooses the credit or debit card option for payment. If the user wants to change anything here he or she can navigate back to the screen where he or she wants to change something or else he or she can tap on the Confirm button to place the order. Figure 5.17 shows the screenshot of the Order Confirmation screen.

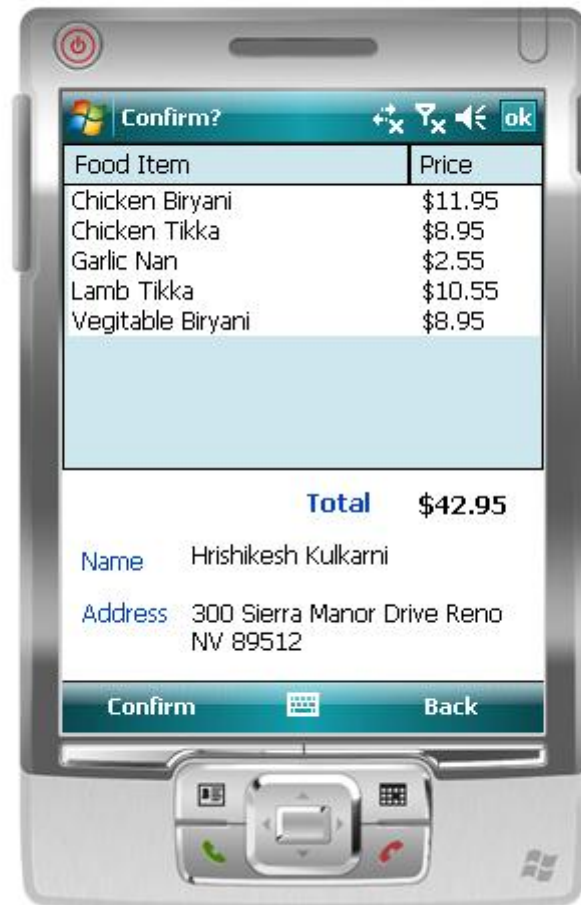


Figure 5.17 Screenshot of the Order Confirmation screen

After the order is placed the user will be directed to the “Order Confirmation” screen showing that the order has been placed with the date and time of the order and the order number for reference. If the data connection is not available, the order will be saved on the pocket PC and once the internet connection has been detected the order will be placed. At the end the user has the option to place another order, and if the user selects Yes, the application will begin from the Welcome screen. Figure 5.18 shows the Order Confirmation screen.



Figure 5.18 Order Confirmation screen

5.3 Usability Components of the MFOS Prototype

The major HCI techniques explained in Section 2.2.2 to overcome the limitations of the mobile device applications are implemented in the MFOS prototype. Some of the techniques used are as follows [16]:

- Using too many menus or long hierarchical menus in mobile applications can be frustrating to the user. The MFOS prototype has only at most 4 selection buttons on any screen without any hierarchical menu structure, which will help the user to place his or her order faster (with fewer clicks).

- As small screens prevent the user from smoothly reading large chunks of text, the MFOS prototype has only few user selection criteria on any screen, with just the required minimum yet sufficient text on each single screen.
- Having a long page on one screen with scrollbars is inconvenient to mobile users as they have to scroll every time they want to see or change the information. The MFOS prototype has very few scrollbars throughout the application as I instead opted for multiple screens when required.
- A mobile application should close network connections and database connections immediately after their use to reduce the battery consumption as well as the connection charges. The MFOS prototype closes the network connection as well as the database connection right after its use.
- Typing on mobile devices is not as easy as on a desktop computer, so the application should store user preferences and profiles wherever it can. The MFOS prototype gives the option to store the user's personal information. As a mobile device is usually personal, this information should mostly be same, so the user can use it without having to re-type it.
- Mobile users expect very quick and clear responses to avoid confusion. The MFOS prototype gives response and feedback to every major action it does, e.g. when the user adds a food item to the cart it displays the response that the food item has been added.

The MFOS prototype provides clear error messages so the user will know what is missing. It also forces the user to enter all the mandatory information and does not allow the user to navigate to the next screen until he or she has entered it. Table 5.1 shows some

of the confirmation and error messages displayed by the MFOS prototype and their descriptions.

Table 5.1 List of confirmation and error messages of the MFOS prototype

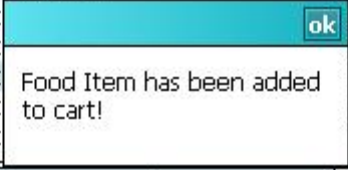

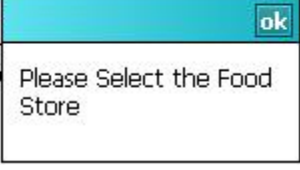
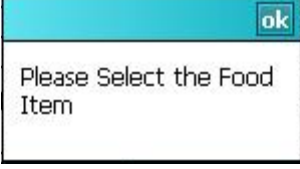
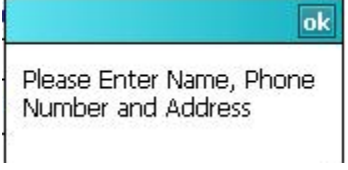

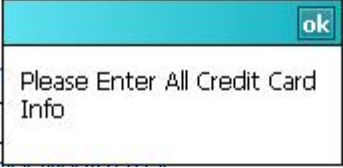


Error/Confirmation Window	Details
 <p>Food Item has been added to cart!</p>	<p>Either in the Food Menu screen or in the Specials screen whenever the user adds a food item to the cart he or she gets this confirmation message for it.</p>
 <p>Food Item has been removed from cart!</p>	<p>Whenever the user removes a food menu item from the cart in the View Cart screen he or she gets this confirmation message.</p>
 <p>Please Select the Food Store</p>	<p>In the Choose Food Store screen if the user wants to go ahead without selecting the food store, MFOS displays this error message.</p>
 <p>Please Select the Food Item</p>	<p>When the user taps on the Add to Cart button without selecting a food item in the Food Menu or Specials, MFOS displays this error message.</p>
 <p>Please Enter Name, Phone Number and Address</p>	<p>If the user does not select the already saved user info and also doesn't type in the user info, this message will be shown.</p>
 <p>Cart is Empty!</p>	<p>If the cart is empty and the user wants to go to the Checkout section, this error will be displayed suggesting to add food items to the cart.</p>

Table 5.1 List of confirmation and error messages of the MFOS prototype [continued]

Error/Confirmation Window	Details
	<p>If the user selects Payment Type as credit or debit card and tries to go ahead without entering the card info, this message will be shown.</p>
	<p>Currently not all cities are supported by this application; this message will be displayed if unsupported cities are selected.</p>
	<p>If the user forgot to select the Order Method and tries to go ahead without specifying it, this message will be displayed.</p>

5.4 Connectivity Aspects

As discussed in Section 2.1.2, with the latest connection technologies such as 3G and GPRS, the mobile Internet is becoming faster and cheaper everyday. However the mobile Internet is still very costly and slower than home broadband connections. Thus, mobile application developers need to keep this in mind and develop their software in a way that would require less Internet connection.

Considering costs for data connection on mobile, MFOS uses Internet only when it is required and closes the connection quickly after its use. MFOS uses the Internet to:

- 1) get the map for store location using the Google Maps web service;
- 2) update the food store information for uploading food menus, and
- 3) send the user's order to the store.

As shown in Figure 4.12, MFOS uses two databases, one at the mobile, and another at the web server. The mobile database contains the food menu for food stores, so when the

user starts the application, the local database of MFOS syncs with the web server database and only updates those food menus which are changed.

6 COMPARISON WITH SIMILAR WORK

As mobile phones and the mobile technology are becoming more advanced and popular, many mobile food ordering products are being launched to attract the customers. This chapter summarizes the differences between some of these food ordering products in the market and MFOS. The MFOS project developed for this thesis is essentially a prototype developed in the academic environment and the commercial products are much more steady or powerful. Thus, the brief comparison on the basis of features offered by different products presented in this chapter is for information purposes only, and should not be taken as an indicator of the products' strengths. For more precise details, the reader is asked to visit the references provided for the products mentioned in this chapter.

The following are some of the products that I surveyed to relate to MFOS:

- CityMint
- Food To Go
- GetQuik
- GoCelly
- GoMobo

CityMint [5] is a food ordering product for iPhone devices to order food from local food stores in New York City. CityMint lets one browse the menus of local places and add different food items into a single order. The user can choose either delivery or pickup depending upon store offerings; the application also has an option of paying right from one's phone. After placing the order it will send a user confirmation via e-mail with one's estimated delivery time, and details of the order [22]. MFOS also is designed to

offer these features. Additionally, MFOS allows users to select food items from store specials, shows the store location on the map, facilitates calling the store from within the application, and as MFOS aims to work with fast food and restaurant chains this application could in the future incorporate many cities. MFOS also differs with CityMint in terms of technology, as CityMint works on iPhones whereas MFOS works on the Windows Mobile Pocket PC platform.

Food To Go [6] is a PDA application where the user can enter the quantity of the food items from the food store menu. The user also has the option of placing special cooking instructions. Then the user just have to give his or her name, address, phone number, and estimated pickup time. Although MFOS works on the same technology (Pocket PC), it has some differences in functionality. Specifically, the current version of MFOS does not take special cooking instructions and does not allow the user to select the estimated pickup time. However, MFOS is designed to offer some other useful features such as delivery option, store location map, and calling the store.

GetQuik [7] is a web as well as a mobile-based application. Using the GetQuik mobile application one can select the food store, store location, choose food items or choose from favorites, and confirm the order. GetQuik also works on iPhones. MFOS offers similar features except the option of selecting food items from favorites. MFOS also does not have iPhone and WAP-based phone compatibility.

GoCelly [8] is an SMS-based food ordering application for pickup. The user first needs to configure different orders on their website and save them with a nickname. For example, the user chooses the food store, then chooses the food items, customizes the order and saves it with a nickname such as MyOrder1. Next, whenever the user wants to

place this order he or she can SMS the nickname to GoCelly service number. But this is restricted in terms of changing orders on the fly and also requires the user to remember all the nicknames corresponding to their orders. MFOS displays the food menu every time one wants to order so one does not have to remember all the nicknames for the orders. Also, one can try new food items without the necessity to add them through the web. In addition, MFOS provides a rich user interface and other options for users.

GoMobo [9] is also an SMS-based food ordering application for pickup, similar to GoCelly. GoMobo also lets one select one's favorites from the website, and while ordering the user has to SMS the favorite item to GoMobo service number. GoMobo also offers a "time of delivery" option in the SMS. As MFOS uses an application interface for menu, the user does not have to remember the syntax of the SMS or a favorite's code. Also MFOS provides a map with the store location of the food store, credit card payment option, and some other features.

The tools mentioned above are powerful commercial applications. They provide some features that MFOS does not provide and are currently operating on the market. However, none of the above applications includes currently a store location map, or the option of calling the store from within the application (which MFOS includes in its design). Table 6.1 contains a chart comparing briefly MFOS with the several other products mentioned in this chapter. The criteria selected for comparison are the following:

- Lets one choose the delivery or pickup option (A)
- Works in different cities (B)
- Shows and Calls the food store's phone number (C)

- Shows the map of the store location (D)
- Shows the food menu (E)
- Lets one enter special instructions for order (F)
- Lets one select pickup time (G)
- Lets one store favorite food items (H)
- Lets one store user information (I)
- Provides option for credit card payment (J)
- Lets one check the already placed orders (K)

Table 6.1 Comparison with related tools

	A	B	C	D	E	F	G	H	I	J	K
MFOS	✓	✓	✓	✓	✓				✓	✓	✓
CityMint	✓				✓			✓		✓	
Food To Go					✓	✓	✓				
GetQuick					✓			✓			
GoCelly		✓						✓			
GoMobo		✓					✓	✓			

As previously mentioned in this chapter, the comparison shown in Table 6.1 should not be taken as an indicator of the products' strengths. It is included here to help place MFOS in the context of food ordering application for mobile devices. While MFOS is a project developed in academia, the other applications are commercial products developed professionally and currently available to their users. Hence they are more powerful and fully operational. Furthermore Table 6.1 includes only a specific set of

criteria, which was not meant to be comprehensive but rather only help identify some of useful features for food ordering applications for mobile devices.

7 FUTURE WORK

As discussed in Chapter 5 and Chapter 6, it can be observed that MFOS prototype offers several very useful features but still it can be improved in many directions such as functionality, usability, and so forth. This chapter discusses some of these potential areas where further work can be done to improve the MFOS application.

As we have shown in Table 6.1, the MFOS prototype offers a set of valuable functions but still several useful functions which are offered by other products are missing in the MFOS prototype. Following are some of the features required to be added in the MFOS prototype to make it a more complete and better application.

- Includes the ability of saving favorite items so that the user can skip some steps of placing an order and save time.
- The user should be able to give special instructions for his or her order.
- The user should be able to select a custom pickup time according to his or her preference.
- The user should be able to repeat the order from his or her last orders.
- The Menu should be more detailed in terms of images and descriptions, such that the user will know better about the food items he or she can choose from.
- The user should be able to add multiple items into the cart at a time.
- Provide a map with driving directions to the store.
- Provide improved security in credit card transactions.

For increasing the customer base:

- Currently MFOS only works on Pocket PC with Windows Mobile 6 Professional. To reach a wider audience it is necessary to make sure that MFOS can work on other platforms also. As development language I used C# so with relatively few modifications this application could work on other Windows Mobile OS.
- For other handheld devices such as iPhone and Blackberry this application has to be re-written in programming languages compatible with their OS.
- Another idea for reaching more customers would be for users who do not have internet connections on their devices, they should be able to send SMS messages for ordering.

As discussed in Section 5.3, MFOS has been developed considering the usability of mobile application as well as considering human-computer interaction challenges of mobile devices. I am planning to test this product with real users and according to their experiences I would like to further improve MFOS' usability.

MFOS is still a prototype, and to become a more complete and fully operational product it has to be tested thoroughly, particularly on real devices (and not only on an emulator). MFOS also has to be tested on an increased network traffic.

Finally, I also intend to work on launching MFOS as a commercial product. Marketing this product to end users, investors, as well as food stores is going to be a key challenge in MFOS' commercialization.

8 CONCLUSIONS

The goal of this thesis was to explore the human computer interaction challenges in designing commercial applications for mobile devices through the development of MFOS. MFOS was designed and developed to provide Pocket PC users with a single tool with which they could order their favorite food from different cuisines without having to wait in the queue and, importantly, be able to order food from anywhere. While developing the interface of MFOS, researching human computer interaction challenges as well as designing a system which can deal with them and provide more useful features to the user were significant objectives of this thesis.

MFOS aims to offer some “cool” and handy features to the users, which other products in the market do not provide in the same way as MFOS. The location of the food store on the map is one of the very useful features of MFOS, which uses the latest functions of the Pocket PC. Similarly, the ability to call the food store, view the menu, pay the order with a credit card, save user information, and view previously placed orders are also several other valuable features of the system.

In summary, the main contributions of this thesis are as follows:

- Exploration of current technologies for developing software applications for mobile devices.
- Exploration of the challenges of developing commercial applications on mobile devices.
- Development of a well-structured UML-based software model for the MFOS application considering HCI challenges in mobile applications.

- User interface design and implementation of the MFOS prototype using the latest development technology (Visual Studio 2008).
- Identification of several directions of future work intended to expand the features, capabilities, and scope of MFOS.

MFOS still has to evolve in terms of features and reliability. Chapter 7 explains in detail the features and scope that need to be improved to make MFOS a more complete and better application. MFOS still needs to be tested thoroughly for reliability and robustness of the application on different actual devices before possibly launching it commercially.

REFERENCES

- [1] CTIA-The Wireless Association Celebrates 25 Years of Mobile Communication
<http://www.ctia.org/media/press/body.cfm/prid/1781>
Accessed: 12/28/08
- [2] Apple – iPhone - Features
<http://www.apple.com/iphone/features/itunes.html>
Accessed: 12/26/08
- [3] Apple – iPhone - Features
<http://www.apple.com/iphone/features/appstore.html>
Accessed: 12/26/08
- [4] eBay Mobile
<http://pages.ebay.com/mobile/>
Accessed: 12/26/08
- [5] CityMint Food on the Go
<http://www.citymint.com/>
Accessed: 12/27/08
- [6] Mobile Food To Go
<http://www.mobilefoodtogo.com/>
Accessed: 12/27/08
- [7] GetQuik – Restaurant Orders Made Easy
<http://www.getquik.com/Pages/Common/Demo.aspx>
Accessed: 12/27/08
- [8] gocelly – Food On the Go VIP Style
<http://www.gocelly.com/>
Accessed: 12/27/08
- [9] GoMobo: Order Food Ahead, Skip the Line!
<http://gomobo.com/>
Accessed: 12/27/08
- [10] Pizza Hut!
<http://www.pizzahut.com/index.html>
Accessed: 12/26/08
- [11] Attewell, J., *Mobile technologies and learning: A technology update and mlearning*. Project summary, London: Learning and Skills Development Agency, 2005.

- [12] Wikipedia: GSM (Global System for Mobile Communications)
http://en.wikipedia.org/wiki/Global_System_for_Mobile_Communications
Accessed: 12/26/08
- [13] Ivanov K., Ball C.F., and Trembl F., *GPRS/EDGE performance on reserved and shared packet data channels*, IEEE 58th Vehicular Technology Conference, Volume 2, 2003. pp. 912- 916.
- [14] Google Mobile
<http://www.google.com/mobile/default/mail.html>
Accessed: 12/26/08
- [15] Haartsen J., Naghshineh M., Inouye J., Joeresson O., and Allen W., *Bluetooth: Vision, Goals, and Architecture*, ACM Mobile Computing and Communications Review, 1998. pp. 38–45.
- [16] Ballard B., *Designing the Mobile User Experience*, Wiley, 2007.
- [17] Yuan M.J., *Enterprise J2ME: Developing Mobile Java Applications*, Prentice Hall PTR, 2004.
- [18] Wigley A., Moth D., and Foot P., *Microsoft Mobile Development Handbook*, Microsoft Press, 2007.
- [19] Amazon Mobile
<http://www.amazon.com/gp/help/customer/display.html?ie=UTF8&nodeId=200287200&>
Accessed: 12/26/08
- [20] ETRADE Mobile Pro
https://us.etrade.com/e/t/mobile_pro?SC=BR12345&WT.mc_id=BR12345
Accessed: 12/26/08
- [21] ETRADE Mobile Pro: Quick Start Guide
https://content.etrade.com/etrade/estation/pdf/ETRADE_Mobile_Pro_QSG.pdf
Accessed: 12/26/08
- [22] CNET: Order food on your phone with CityMint
http://news.cnet.com/8301-17939_109-9950302-2.html
Accessed: 12/27/08
- [23] Booch, G., Rumbaugh, J., and Jacobson, I. *The Unified Modeling Language User Guide*. Second edition, Addison-Wesley, 2005.
- [24] Arlow, J. and Neustadt, I., *UML and the Unified Process: Practical Object-Oriented Analysis and Design*. Second edition, Addison-Wesley, 2005.

- [25] Object Management Group - UML
<http://www.uml.org/>
Accessed: 12/26/08

- [26] Sommerville, I., *Software Engineering*, 8th edition, Addison-Wesley, 2006.

- [27] Lee, V., Schneider, H., and Schell, R., *Mobile Applications – Architecture, Design, and Development*. Prentice Hall, 2004.

- [28] Windows Mobile 6 Professional and Standard Software Development Kits Refresh
<http://www.microsoft.com/downloads/details.aspx?familyid=06111A3A-A651-4745-88EF-3D48091A390B&displaylang=en>
Accessed: 12/26/08

- [29] WM 6 SDK and Cellular Emulator
<http://blogs.msdn.com/fzandona/archive/2007/04/11/wm-6-sdk-and-cellular-emulator-can-you-hear-me-now.aspx>
Accessed: 12/26/08

- [30] Windows Mobile Device Center 6.1 for Windows Vista
<http://www.microsoft.com/windowsmobile/en-us/help/synchronize/device-center.msp#>
Accessed: 12/26/08