2009

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

iTour:
A System for Self-Guided Virtual Campus Tours of UNCW

Camilo Alvarez

A Capstone Project Submitted to the
University North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems and Operations Management

University of North Carolina Wilmington

2009

Approved by

Advisory Committee

Dr. Bryan Reinicke	Dr. Douglas Kline

Dr. Ron Vetter, Chair

Abstract

This paper describes the systems analysis and implementation activities of an Apple iPhone application that provides self-guided virtual campus tours of UNCW. These tours enable visitors to physically navigate the university campus while at the same time receiving useful information about its buildings and facilities. The systems analysis presents an overall system description, actor diagrams, major systems components, use case analysis, data model, and mobile device user interface design documents and considerations. A detailed description of the technical challenges faced during system implementation and the solutions adopted are also discussed. This includes the implementation activities of the iPhone application and its companion content management website. Finally, the results of a small usability study are presented.

List of Tables

List of Figures

Table of Contents

Chapter 1: Introduction

For many U. S. universities, campus tours are essential in the effective recruitment of new students. Currently, universities offer a number of different options – from traditional campus visits to the more modern web-based virtual tour. One of the major influences on a high school graduate's decision to attend one college over another is based on the information and experiences they encounter while exploring a university [2]. This is a fundamental decision that marks the start of a potential four year (or more) student-college relationship.

For this reason, campus tours are critical to keeping the interest levels of new students high, and as a result help universities accept the best students and convince those students to join them. A 2004 poll of 472 high school graduates, directed by the Arts & Science Group, a marketing consulting firm for higher education, concluded that 65 percent of students base their college decision on campus visits. This was followed by 26 percent from their websites, and 25 percent from printed materials [2]. In other words, for future college freshmen, a campus visit is the most influential source of information in the selection of their new campus.

However, many universities have been shifting admissions practices by engaging in other recruitment activities and neglecting the importance of the campus tour. Resources are being allocated to college fairs, advertisements, website enhancements, college booklets, and other media that ends up being forgotten or thrown away [14]. This is a less than desirable situation for all parties. Students investing time in a campus visit expect to receive accurate information and a good impression of a college, but they might not find it when they visit the campus; and universities who are spending substantial resources in alternative media advertisement, risk losing their investment and the interest of their visitors.

Similar to other universities, UNCW has put a plan in place for visitors that consists of scheduled guided group tours, and a brochure that summarizes how to take a self-guided campus tour. However, this is not adequate for the numerous visitors who cannot attend scheduled, in-person guided tours, yet still expect to receive the best information available. Therefore, it is necessary for the university to reinvest, rethink, and refocus efforts to provide prospective students with experiences that are similar to the traditional campus tour. One alternative, and an objective of this project, is to utilize newly available technology provided by Apple with their GPS-capable and programmable iPhone mobile phone as a solution for providing campus tours when in-person tours are not available. The purpose of this project is to show how effective campus tours can be accomplished by an innovative iPhone application, called iTour, for the UNCW campus.

1.1. Purpose of The Project

There are several objectives the iTour project is aiming to achieve. First, is to become a valuable addition to UNCW's Office of Admissions marketing strategy. The availability of mobile device tours will allow visitors, who cannot attend in-person guided tours of the university, the opportunity to go on virtual tours anytime they want. Parents and students, who would like to visit and receive information about the campus but are unable to participate in the guided tours provided by the university, will be able to receive a helpful orientation of the UNCW campus using the iTour application. These users will be able to navigate the campus first-hand, knowing where they physically are located at all times, and receiving real-time information about their surroundings. Thus, UNCW will be able to provide prospective students and their parents with campus tours that fit their schedules.

Another objective of the iTour application is the standardization of campus tour content. By standardizing the content of tours, the admissions office will have more control over tour content and thus the visitor experience. In addition, by ameliorating the burden of providing in-person guided tours, which imply constantly allocating and coordinating resources to this activity, the university should be able to save time and money.

Finally, a third objective of this application is to encourage prospective students to get involved in the Computer Science and Information Systems (CSIS) program. The fact that the iTour application was developed in-house by a CSIS student should add to the university's appeal for prospective students who are interested in computer science, software engineering, and information systems related fields.

1.2. Significance of the Problem

Currently universities offer visitors a number of options to experience and receive information about their campuses. UNCW offers "student ambassador" guided tours in which current students direct a group of registered visitors through the university. The tours follow predefined routes in which ambassadors visit several campus buildings (or points of interest) while reciting information prepared by the admissions office. Another popular option offered by several universities, including UNCW, is online tours. Most of these tours can be found on the admissions website of each university and are varied in their content and presentation. Among the tools provided for visitors on these websites is a mix of maps, documents, pictures, videos, etc [7, 16, and 17]. A third, less known, but emerging option used by a small number of universities consists of a combination between the physical and web-based campus tours using a

mobile device. The following subsections discuss the advantages and disadvantages of these existing methods as sources of information and recruitment tools.

1.2.1. Discussion of Alternative Solutions

Face-to-face guided tours

Face-to-face guided campus tours give prospective college students the most influential source of information for making a decision to attend a college over another [2]. High School seniors are advised by their institutions to look beyond paper advertisement they receive in the mail, or the sleek web pages they visited while gathering information about a college of interest. "While a college may look fantastic on paper, it may not feel right when you physically set foot on campus" [4].

The guided campus tour has many advantages, especially when it is provided by a person that is eloquent and knowledgeable about the university. Prospective students can ask the guide specific questions that relate to their own situation or, if possible, they might even ask other students around the campus. This first-hand experience can give visitors a better sense for the campus's dynamics and social environment. In these face-to-face tours, the guide plays a pivotal role. Both, the university and the visitors depend on him or her for being a pleasant, accurate, and reliable source of information.

However, there are some disadvantages with this solution for campus tours. Starting from the fact that not all universities offer guided tours of their facilities; to a lack of training for those who administer these important tours. Fortunately, UNCW has an active and very successful guided campus tour program. Every day of the school year (this excludes Saturdays and Sundays) guided campus tours are provided for visitors two times a day. But even so, the visitor

must register for a tour at least three days in advance for a tour that starts at a specific hour of the day. Out-of-State visitors often find it difficult, inflexible, and impractical to have to coordinate so many variables to make it into a scheduled campus tour. These visitors might be interested in attending the scheduled tour, but their own time constraints may not allow them. For those fortunate enough to attend scheduled campus tours, another issue might arise. The campus guide might not offer the experience or information they need or expect. Although UNCW has numerous well-informed and agreeable "student ambassadors" who volunteer to take groups of visitors around campus, there is always the occasional mishap. It is not unusual that an ambassador cannot answer a question, or the response to a question is not phrased in an appropriate manner [2]. The primary goal of the person directing the tour is to provide accurate, gaffe-free information to the interested parties. Because tours last as much as two hours and information recited by the student ambassadors sometimes is not transmitted or not delivered efficiently, there is always a concern that the university message might be getting lost.

Although face-to-face guided campus tours are the preferred and most influential source of information for prospective students visiting a campus, it is undeniable that several of the issues that arise from using this option require another alternative to complement, or even replace the in-person tour. This new alternative should try to keep the advantages of offering a firsthand experience, while also realizing that the delivery of information in a precise and clear manner is very important.

Technology-Enhanced Tours

Many universities, including UNCW, have opted for offering students a web-based virtual campus tour of their facilities. This type of tour tends to serve as a point of introduction

for high school seniors trying to gather as much information as possible about a university before visiting it. Virtual tours go "…beyond providing better access to information that might be secured the old-fashioned way through catalogs, brochures, directories, and guidebooks" and "… rapidly growing numbers of higher-education institutions are taking advantage of the interactive, multimedia capabilities of the Web to offer "virtual tours" that give prospective applicants a real-world flavor of their campuses" [5].

There are several dedicated websites that offer users links to a variety of college and university's sites and their web-based campus tours. The following are examples of some of the sites that provide these services for universities and students:

- CampusTours.com - http://www.campustours.com
- CollegeNET - http://www.collegenet.com
- CollegeView Virtual Tours - http://www.collegeview.com/vtours
- Kaplan Test Prep - http://www.kaptest.com
- Link Camp Tours - http://www.linkmag.com/Links/College_Campuses
- Virtual College Day - http://www.criterioninfo.net/vcd

The advantages that these web-based campus tours offer can be significantly appealing to students. In particular, "…students can visit several schools without spending hardly any money or missing days of school" [5]. From their own homes, prospective students and their parents can comfortably and inexpensively obtain valuable information in an easy manner. This type of tour includes several tools ranging from a picture and description of a building, to the more sophisticated 360-degree panoramas with audio and video comments. In this Internet era, the sole fact of not having a website featuring some content about the university for visitors can be a severe disadvantage for new student's recruitment.

Although the virtual tour option offers visitors the opportunity to access varied and precise information about the university, the approach lacks some benefits and places a certain burden on the visitor. Most online solutions require the visitor to have a medium to high level of understanding about a myriad of web technologies. In some instances, too many tools are used concurrently, which can cause confusion and frustration for visitors trying to find simple and specific information. Some web technologies used include RealPlayer (http://www.real.com) and Microsoft Windows Media Player (http://www.microsoft.com/windows/mediaplayer) for audio and video; VivoActive PowerPlayer (http://www.vivo.com), Quicktime and Quicktime VR (http://www.apple.com/quicktime) for movies and virtual reality. Also popular are: Live Picture (http://www.livepicture.com) and iPIX (http://www.ipix.com) for 360-degree panorama views; and Java (http://java.sun.com) and Shockwave (http://www.macromedia.com) for animations [5]. More disadvantageous still is the fact that many virtual tours are static and don't provide a physical experience. Universities may post in their websites the most appealing information about their campuses, but may fail to provide details that prospective students are interested in. Most students know this, and therefore use the website campus tour solely as a reference to make a decision further down the road. It has been noted by admissions officials that, "While online virtual tours can be a great source of information, they still cannot replace in-person visits to college campuses and taking the official tour, meeting students, attending a lecture and even staying in a dorm on an overnight prospective student visit" [8].

Another technologically-enhanced alternative introduced recently, which could be used for campus tours, is the mobile device virtual campus map. Currently, around fourteen universities worldwide have iPhone applications specifically designed for their own use. This number is quickly increasing as more and more students and institutions adopt the new

technology. Two of the first universities to develop iPhone applications include: Stanford University which has its iStanford application and Duke University with the Duke Mobile application. Both applications were developed by TerriblyClever Design, a company based in San Francisco California (this company has been acquired by Blackboard in June 2009 for 4 million dollars). The same framework is used for both universities' applications. The applications feature several functions that provide relevant information to the public. For more information, refer to [1].

- Athletics – To browse news, schedules and scores for athletics teams. When a game is in progress, scores are updated in real time.
- Courses – Provides the school's entire course listings. Detailed views of a single course allow the user to simply touch the professor's name to access the directory application, or touch the course location to access the maps application.
- Directory – To browse the institution student/staff directory to find a contact's email address, school id, phone number, title and other information made available by that user.
- Events – To access the school's event's calendar to search for any events happening on or around campus.
- Images – To view through the school's photo archive or search for any specific photo by keyword. Users can send photos to a friend or download as wall paper for the phone.
- News – To browse all news articles for the institution, organized in categories similarly to the school's traditional website. Utilizing push notifications, users are alerted of the most recent and significant articles as they are published.
- Videos – Serve to stream iTunes U or YouTube or custom video content directly from within the application.

- Maps – Can be used pin point the user's exact location on the campus using GPS, and search for any building by name and address. When made available, users can also see a photo of the building they search for.

Although these applications provide a map function, the difficulty with this solution for campus tours is that the map has no interactive content and no additional information such as what departments reside in a certain building, their history, or links to their websites. Thus, these applications are of little use to visitors who want to become familiar with a new campus and provide limited new information for existing students.

1.3. Research Questions

In comparing solutions for campus tours offered by different universities one can see the similarities in approaches and the opportunity that exists to develop a new alternative. That is the primary aim of this project – to develop a new alternative for the traditional campus tour. Three research questions are explored in the design, development and implementation of this project. These questions arise from the need of a new alternative for campus tours and the apparently seamless match with the solution offered by the emerging iPhone hardware and software technologies. The research questions are:

a) Is the iPhone a suitable mobile device to conduct campus tours?

b) Can an iPhone virtual campus tour, by unifying data with a companion website, serve as an appropriate source of information and support different and incompatible web technologies?

c) Are the technologies offered by the iPhone device appropriate for implementing virtual campus tours? In particular, are iPhone's built-in GPS capabilities and its Wi-Fi and cellular network connections adequate for supporting the virtual campus tour application?

The remainder of this document is organized as follows. Chapter 2 provides some important background information on the capabilities and features of the iPhone and the applications development process. The underlying programmable components of the iPhone play an important role in understanding the capabilities and limits of the technology. In chapter 3, a review of alternate software engineering design methodologies is provided, along with the analysis and design activities undertaken for this project. This includes use cases, data modeling, user interface design, and overall system architecture. Chapter 4 describes development choices, implementation issues, and usability results. Finally, chapter 5 provides a summary of lessons learned and future work.

Chapter 2: Background

The Apple iPhone is one of the most popular mobile phones on the market today. Since the original Apple iPhone was released in June 2007, later the iPhone 3G in 2008, and then iPhone 3GS in 2009, tens of millions of phones have been sold. The Apple software developer program allows users from all over the world to develop applications for the iPhone and post them online in the Apple iTunes App Store. The App Store had 1.5 billion application downloads in its first year. Apple claims over 100,000 paid and free apps are now available for download in 77 countries.

In spring 2009, UNCW joined Apple's Developer University Program, a program designed for higher education institutions looking to introduce curriculum for developing iPhone or iPod touch applications. The University Program provides a wealth of development resources, sophisticated tools for testing and debugging, and the ability to share applications within the same development team. Institutions can also submit applications for distribution in the App Store.

Because the iPhone platform is fairly new, the development of mobile software applications, such as iTour, require a solid understanding the Objective-C programming language and the inherent capabilities of the device. This chapter provides some needed background and information on how to develop and deploy iPhone applications.

2.1 iPhone Development

Developing applications for iPhone devices is very different from developing applications for desktop computers. The smaller size of the iPhone implies that the information displayed in the application's user interface should be well organized and always focused on

what the user needs most [10]. Additionally, iPhone development requires a specific set of tools and a unique knowledge set for successfully creating and distributing an application.

Although the iPhone mobile device must account for size constraints, it also brings innovative ways for users to interact with its applications. The iPhone operating system (OS) and its multi-touch interface enable the collection, management, and processing of complex user's input actions and data seamlessly. Additionally, the device is equipped with hardware that can identify its position on the globe using GPS satellites, and an accelerometer that responds to the device's motion which can be used to change the content and orientation of an application. The iPhone is constantly being upgraded. Since its inception to the market, three different hardware versions (iPhone, iPhone 3G, and iPhone 3GS) of the phone have been introduced, each one improving on the features and capabilities that the previous version possessed. However it has been the main features mentioned above (GPS, accelerometer, and touch interface) that have placed the iPhone a step above other devices in the mobile phone landscape.

There are a number of tools required to develop applications for the iPhone. The only desktop OS's that support iPhone application development are the Mac OS 10.5 Leopard and its later version Mac OS 10.6 Snow Leopard. This means that having an Intel-based Mac computer is obligatory, since Mac OS 10.5 and later now only run on Intel-based Mac machines. The next requisite is having the latest iPhone SDK (System Development Kit). This is a free set of software tools which can be downloaded from the Apple developer website (http://developer.apple.com/iphone/). iPhone applications are written in the Objective-C programming language. At a minimum, it is essential to have a basic understanding on how the object oriented language Objective-C works. A final recommended tool for effective application development is a physical iPhone. A physical iPhone becomes necessary to test the developed

applications extensively. Although the SDK has an iPhone simulator in which applications can be tested, run, and monitored, available memory differences between a physical iPhone and the simulator can be significant. Additionally, the iPhone GPS and accelerometer features can only be tested using a real iPhone.

2.1.1 iPhone OS Overview

The iPhone OS is the operating system that runs and controls the internal functionality of the iPhone device. "This operating system manages the device hardware and also provides the basic technologies needed to implement native applications on the phone" [10]. The OS is composed of four technology layers (figure 1). The "Core OS and Core Services" layers deal with the fundamental interfaces for accessing files, network sockets, and low-level data types among others. The next layer, the "Media" layer, help support more advanced technologies such as handling and rendering 3D graphics, audio, and video. Finally the "Cocoa Touch" layer provides the basic frameworks used by iPhone applications to access the device's features and capabilities such as obtaining the user's coordinates through GPS, accessing a page on the Internet, or playing and recording video [10].

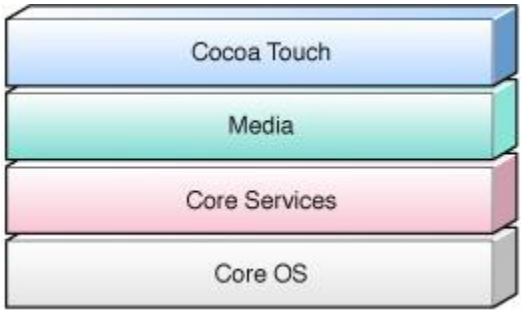


Figure 1 - iPhone OS Technology Layers [10]

2.1.2 Tools For iPhone Development

The iPhone SDK includes all the necessary programmatic tools for developing iPhone applications. Within the SDK there are special packages called frameworks which provide access to the OS features and functions that can be used by applications. In order to use a framework it is necessary to link them to the application's project. Besides the provided frameworks, the iPhone SDK features a set of applications that support code editing, building executables, debugging, performance tuning, and interface creation among others. These are known as Xcode tools.

The main Xcode application is an Integrated Development Environment (IDE) that allows the user to create the source files of an application's project, compile and build the code into an executable, and run and debug the code into an iPhone Simulator (figure 2) or on a physical iPhone device attached to the computer [10].

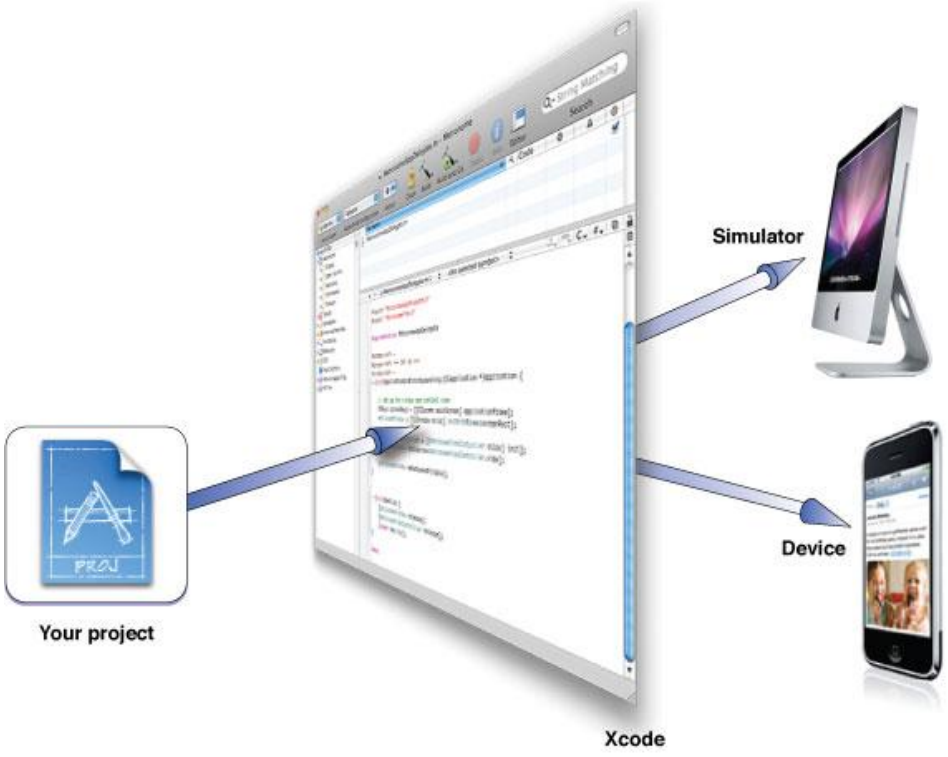


Figure 2 - Running a Project from Xcode [10]

Another essential part of the Xcode application suite is the IB (Interface Builder). IB can be used to assist in the design and implementation of the applications interfaces. IB comes with several prebuilt components which can be dragged and dropped onto a window that resembles the typical iPhone window's dimensions (figure 3). After an interface is built, IB connects source files created in Xcode to the respective interface objects set in IB. Interface Builder is a visual editor that eliminates the need for custom code to create, configure, and position the objects that make up an interface [10].

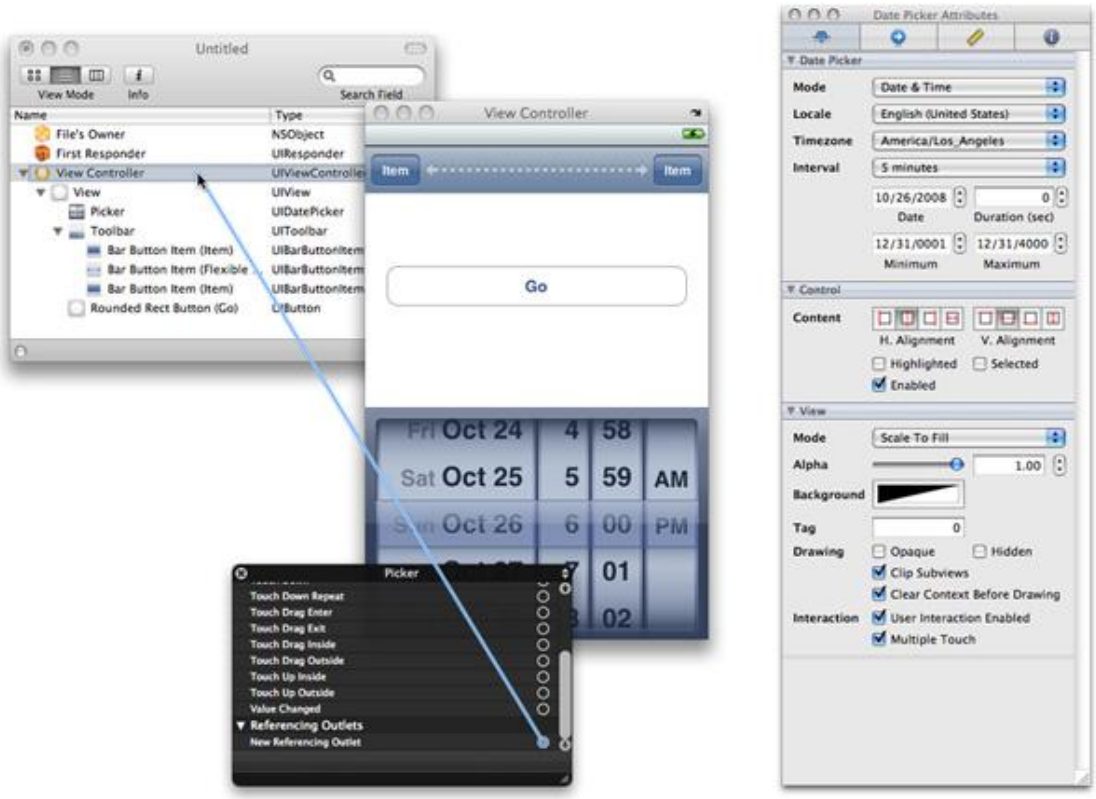


Figure 3 - Interface Design in Interface Builder [10]

The last component of Xcode is the "Instruments" application (figure 4). This application's purpose is to help fine tune developed iPhone applications. The instruments panel collects data of running applications. This helps the developer monitor and analyze memory

usage, disk activity, network activity, and graphics performance. Although not entirely clear, Apple has placed certain limits on the resources a certain application can consume while running. Memory usage, for example, is "recommended," that it doesn't exceed 20 MB of the total memory RAM of the device. Applications that exceed the threshold can be rejected from the iTunes Application Store. For more information on Instruments, see [10].

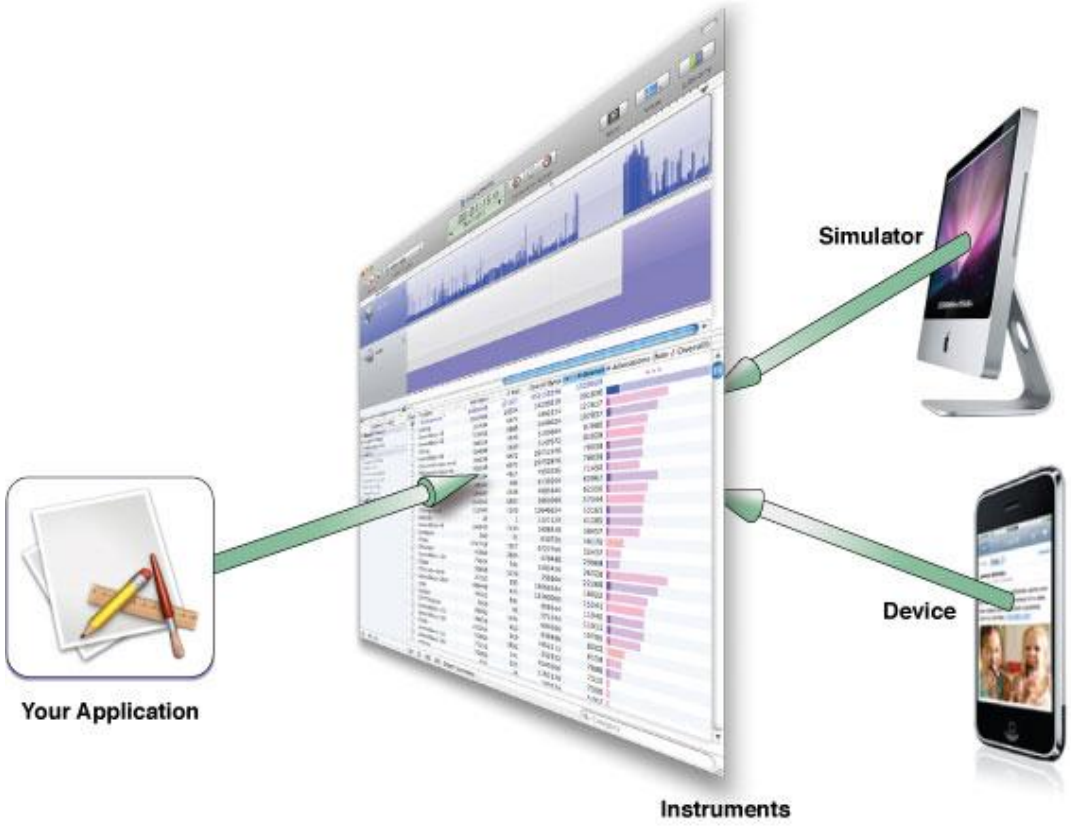


Figure 4 - Application Fine-tuning in Instruments [10]

2.1.3 The Objective-C Programming Language

Objective-C is the programming language used to write iPhone applications. This language is "a superset of the ANSI version of the C programming language and supports the same basic syntax as C" [10]. In Objective-C classes are defined in two parts: header and

implementation files. Header and implementation files are denoted by the extensions ".h" and ".m" respectively. Headers present the public declarations of the classes while the implementations contain all the code necessary for their methods.

In addition to classes, Objective-C features other basic programming constructs such as methods, properties, protocols, and delegates. As in other object oriented programming languages, classes are the blueprints from which objects can be created and contain all the data and actions that can be performed on that data. These actions are known as methods. Methods in Objective-C can be of two types: instance and class methods. These are declared by a type identifier, a return type, signature keywords, parameter name(s), and the name(s) information. A number of special "hidden" methods in Objective-C are known as "declared properties." These are declarations that simplify a class by creating accessor methods (get and set) without having to write their code explicitly. Another construct that deals with method's organization and accessibility are the "protocols." Protocols are interfaces that present a set of methods without implementing them. Other classes "conform to a protocol" by implementing and overriding the methods specified in them. Objective-C also provides another type of constructs known as delegates. Delegates are helper objects in a program which other objects act in conjunction with, or transfer tasks to, in order to accomplish some function for the main object.

2.2 Building an iPhone Application

The building of an iPhone application process is divided in three main steps: creating an application project; creating the application's interface, writing the code, building, running and debugging, and finally tuning for best performance.

To start building an iPhone application in Xcode, a new application project must be created. Most of the coding while developing an application is done through the application's project window (figure 5). This window includes five major sections through which the application can be edited and configured. The toolbar which has the main Xcode commands, the favorites bar in which frequently used file locations can be stored for quick access, the groups and files list which provides an overview of all the files and folders that make up the full project, the detail view which displays a chosen file details at a time, and the status bar which shows the application's state while built or ran.

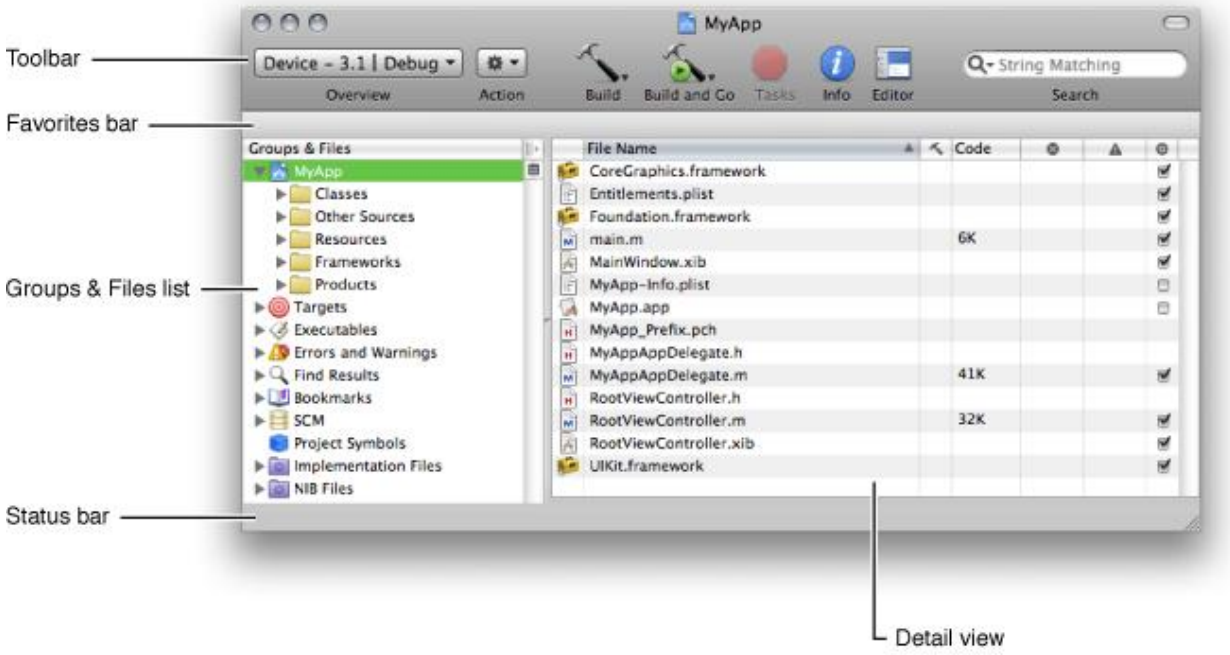


Figure 5 - Xcode Project Window [10]

To build an application, Xcode compiles the application's source code files and then places the binaries in the iPhone simulator application or in a physical device attached to the computer. The iPhone simulator (or device) then attempts running the application. If it fails, or if

it fails while testing the application's functions, failure messages will be shown in the compiler's window.

Some common error messages for Objective-C beginner developers are related to object's memory allocation and de-allocation. To find and improve the memory performance of a running application's resource (among others) Instruments is used. Using Instruments allows an application's performance analysis through which the developer can make code changes to make the application's resources as efficient as possible (figure 6). For more information see [10].

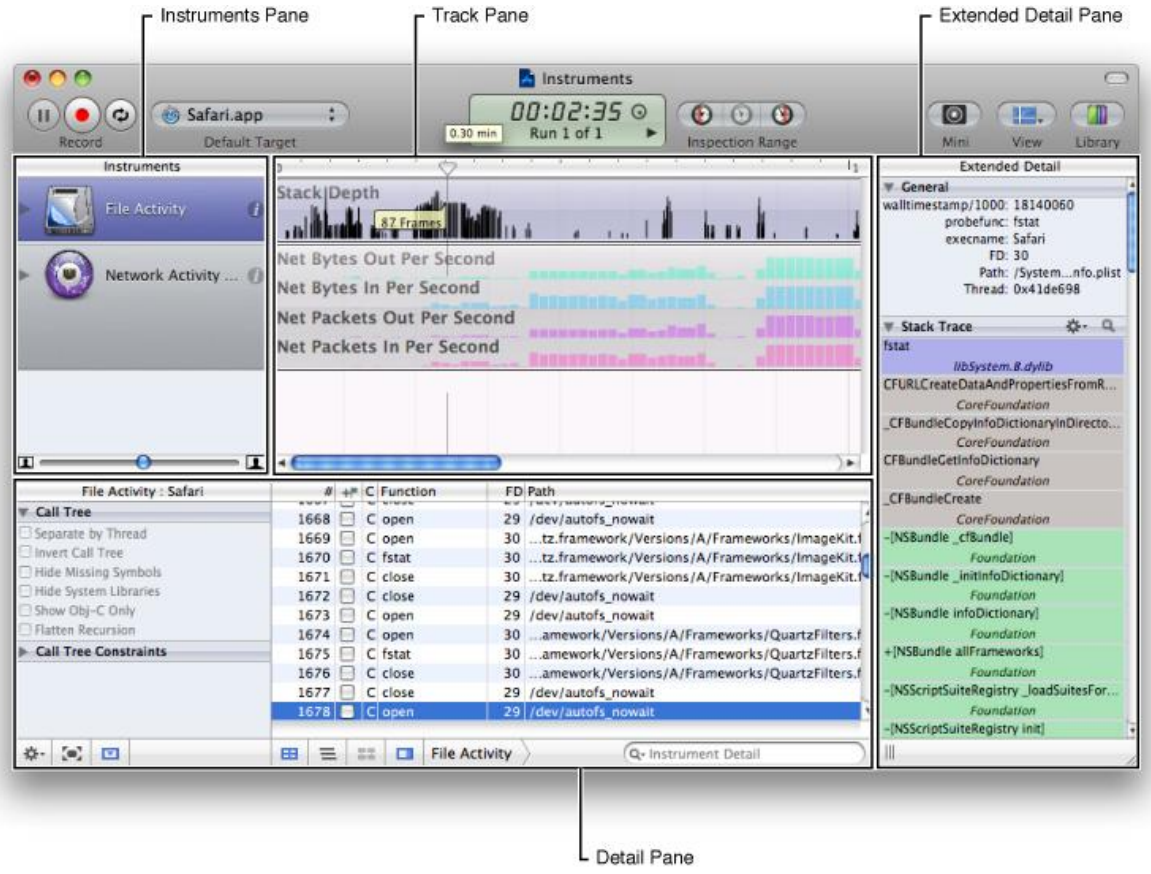


Figure 6 - Instruments Window [10]

2.3 Deploying iPhone Applications

There are three ways to deploy iPhone applications: in an ad-hoc manner, for enterprise distribution, and through Apple's iPhone application store (AppStore). Ad-hoc and enterprise deployment share a very similar distribution procedure that differs only by the number of devices (100 for the former, 500 or more for the latter) allowed to install the application [12].

Ad-hoc distribution is mostly used while testing an application among users that belong to one organization, while enterprise is used for deploying in-house proprietary applications to authorized users in a company. These methods of distribution require pre-registration of devices to the organization's developer program. In order to be registered, the devices' unique identifier (UDID) must be known, and a provisioning profile must be created. A provisioning profile is a security certificate that authorizes a unique device to run the application being tested [9]. The entire process of enabling a device to install an application can be found at the iPhone developer website (http://developer.apple.com/iphone/).

Uploading an application to the AppStore enables iPhone users (with iTunes accounts) the ability to download the application through the Internet. The process is divided in three steps: (i) register for individual or enterprise development with Apple, (ii)sign the application using the individual or enterprise's developer certificate, and finally (iii) upload the application's information and executable binaries to iTunes Connect (only Apple's authorized agent for an organization can perform the last step). Once the application is reviewed and approved by Apple, it should be downloadable through the AppStore or Apple's iTunes to any iPhone or desktop computer. For more information, see [10].

2.4 Summary

Apple's iPhone is one of the most advanced mobile phones on the market. Its intrinsic hardware and software capabilities are a generation ahead of many of its competitors. Developing and deploying iPhone applications requires a good understanding of the Objective-C programming language and the XCode integrated development environment. Also, software developers must join Apple's iPhone Developer Program, to be able to develop and distribute applications for the iPhone and iPod touch. Prior to uploading an application to the iTunes App Store, developers must be approved as authorized agents by Apple. Applications submitted to the iTunes store are reviewed by Apple and are generally approved or declined within two or three weeks. The entire process, from inception to testing to approval and finally appearing in the app store, is quite involved and complicated. The iPhone Developer Program and the SDK, including XCode tools such as Interface Builder, the simulator, and debugger, help to streamline the process and make it more manageable.

Chapter 3: Analysis & Design

There are many methodologies that could have been chosen for the software development process of this project, including: the waterfall model, extreme programming paradigm, spiral model, and the unified process model. We begin this chapter by reviewing these potential methodologies and then discuss the particular methodology we choose for this project. Next, we discuss the desired capabilities of the iTour system and provide both preliminary and completed analysis and design documentation. An overall system description, actor diagrams, major system components, use case analysis, entity-relationship data model, and mobile device user interface design constraints are highlighted.

3.1 Review of Potential Methodologies

3.1.1 Waterfall Model

By dedicating time up front to the collection of requirements and design of the system, the waterfall model offers the possibility of finding some bugs and correcting them early in development cheaper in terms of time and effort. However, this model is inflexible in that it does not allow revisiting early phases for corrections or improvements. In reality (for non-trivial projects) it is almost an impossible idea to perfect each phase of the software development lifecycle (SDLC) before moving to the next one [19]. Requirements may change due to client's input or technologies used during development may evolve. Another potential threat of this model to the completion of a project is that unawareness of implementation issues may occur. By investing too much time in the design phase, difficulties in implementation later in the process may result as more development effort will be required.

3.1.2 Extreme Programming Paradigm

In response to the waterfall model, the extreme programming (XP) model offers an almost complete opposite development methodology. Its benefit lies in the absence of unnecessary documentation, basing development in the constant exchange of ideas between client and developer. The development is carried out through several coding, testing, and design cycles, where coding and functionality implementation drive the advancement of the project. Problems with XP are significant in that by having a client frequently involved in the project's development without predefined requirements can lead to scope creep and costly rework [6]. Also because extreme programming is code-centered and not design-centered, the reusability of a medium size project (as the iTour) could be limited. Lack of documentation makes it unnecessarily harder to reuse a non-trivial project for other purposes without the knowledge imprinted in past documentation.

3.1.3 Spiral Model

Similarly to the waterfall model, the spiral model requires predefining and establishing requirements in the first phases of the SDLC as explicitly as possible. This gives the advantage of an early focus to producing a well-documented reusable product. Also, by taking an iterative approach, which evaluates risks of implementation; presents different alternatives; and allocates time for analysis and feedback, the spiral model is flexible for addressing changing requirements and implementation issues [20]. A disadvantage of this model, however, is that it demands manager's or client's approval in all development iterations. Since the technologies used for the iTour application are new, little is known about their capabilities and limitations. This makes reporting to clients unfeasible and distracting.

3.1.4 Unified Process Model

Although the unified process model shares many characteristics of the spiral model and benefits from a structure similar to the waterfall model, it is not limited by their rigidity. In the unified process (UP) model developers have the freedom of determining when the project is ready to move to the next phase of development without requiring approval from third parties. Also, in contrast to the waterfall structure, past phases can be reviewed and improved according to the project's status later during development.

3.2 Selected Methodology

The methodology chosen for this project adheres to the principles of the Unified Process systems development lifecycle (figure 7). Such approach was assumed would provide enough flexibility so that, through each system development activity, there would be the capability to plan ahead specific assignments in manageable periods of time. These included gathering requirements, designing, implementation, construction of the software, and finally preparation for transitioning to the next iteration. As the iTour project was centered on the development of an iPhone application and a tours web content management system, the UP disciplines aided in guiding the project based on the necessary use cases for campus self-guided tours and the administration of the tours content by the admissions office. Although the UP was a good fit for development, it was found that since the technologies used for implementing the systems were relatively new, widely unknown, and difficult to manipulate, the elaboration phase of the SDLC dominated beyond expectations the lifetime of the project. In chapter 4 (development and implementation) the major assignments that were carried out during the development of the

project will be described. In chapter 5 (conclusions and lessons learned) there will be a brief analysis on the appropriateness of the selected methodology for this project.

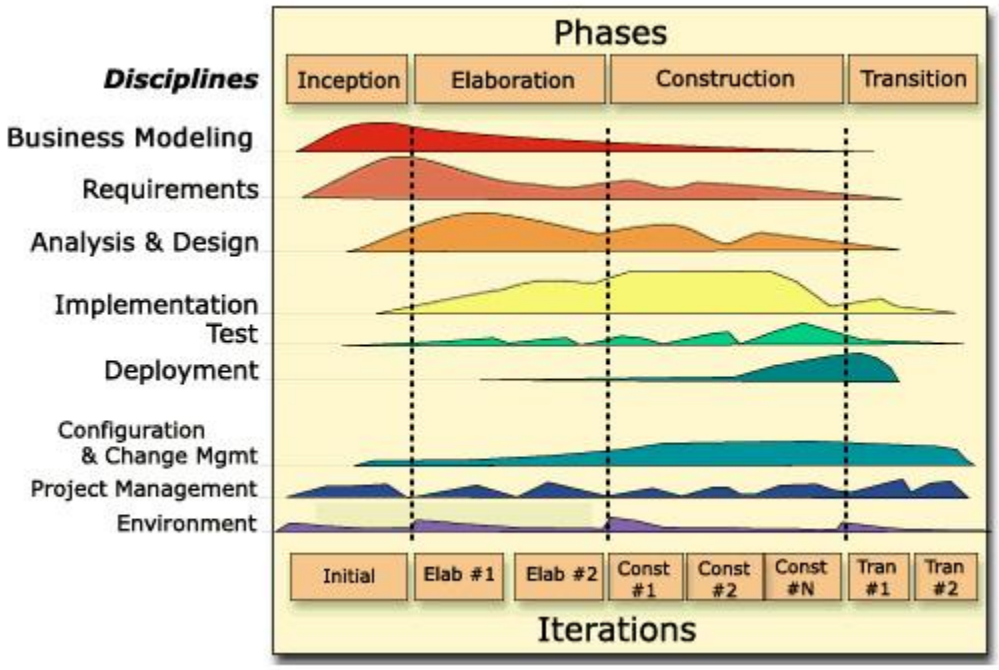


Figure 7 - The Unified Process Systems Development Lifecycle

3.3 Desired Capabilities of Proposed Solution

Early in the project it was determined that the proposed application solution should have the following functions and capabilities:

1. Find the user's location and pinpoint it on a map shown as an interface in the application itself. The purpose of this capability is to enable tourists to quickly recognize their location from an aerial view of the university and hence have an immediate grasp of their position with respect to the university's campus.

2. Display all UNCW points of interest (POIs) on the map that the office of admissions deems necessary. This should allow visitors to identify where the points of interest are located with respect to their own location.
3. Show a list of the tour's point of interest and display the user's current location in the map at a time. This would enable users to visualize their own location with respect to one specific building or quickly select a particular POI for further actions.
4. Provide basic information about each point of interest. This information should encompass a brief description of the landmark, pictures, voice commentaries, and internet links to additional sources in different media formats. All of this information should be located and accessible from one centralized spot.
5. Allow media (audio/video) to be accessed and played from within the application.
6. Allow the tour's content to be editable from a web content management system external to the application itself. The purpose of this capability is to permit the office of admissions the opportunity update the university's tours information though a user friendly, easy to use interface, without having to recompile or update the iPhone application's native code.

3.4 System Description

The iTour system accomplishes finding information and touring the university, a simple activity for users. However, the infrastructure, technologies, and interacting components working behind this system are significantly complex. Users are shielded from this complexity by allowing their interaction only with those interfaces that provide the information they need. In this section a description of the system as a set of networking pieces is described at a high level.

The purpose of this description is facilitating a better comprehension of its working elements, visible and hidden, and the relationships with one another.

Three major activities can be used to describe how the elements of the iTour system communicate with each other: (1) making the iPhone application available for download (which only takes place once), (2) touring the university, and (3) changing the tour's content. The figure below is used to illustrate how the communication between these different activities occurs in the system.

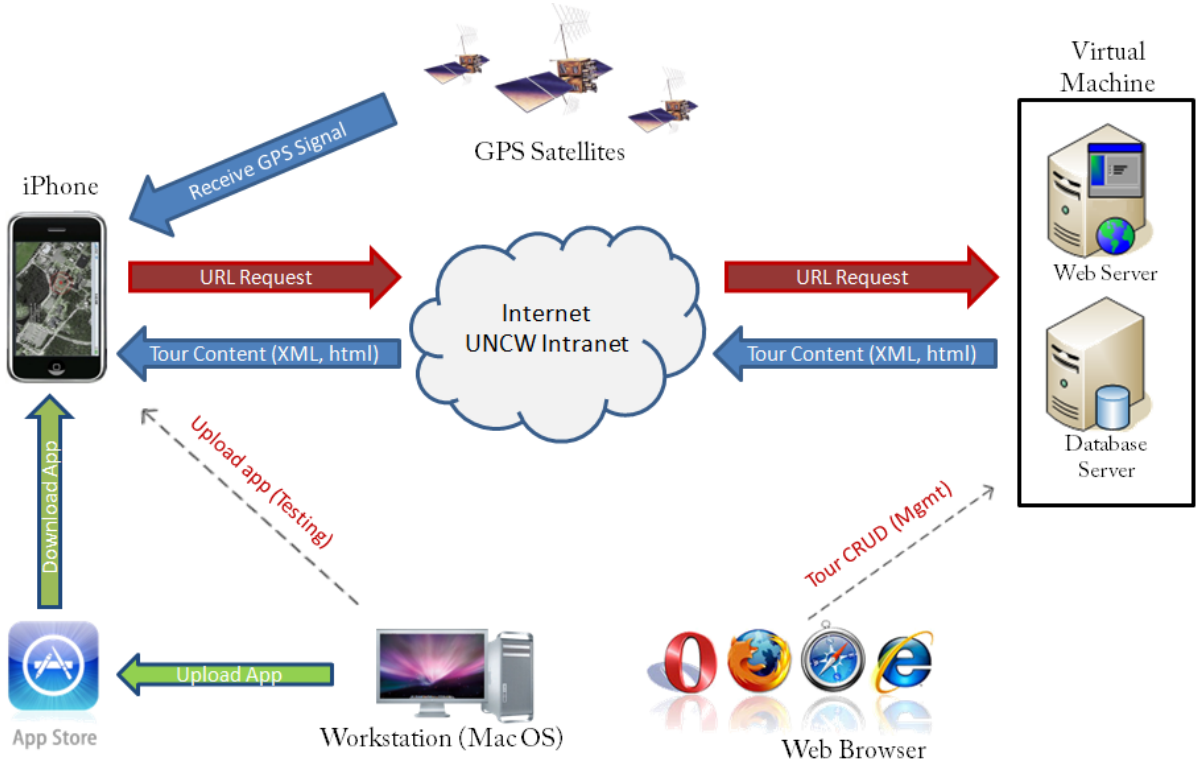


Figure 8 - System Diagram

(1) The green arrow lines and the dotted line in the bottom left part of figure 8 indicate development and deployment of application activities:

A. A Mac OS X workstation is used to create the iPhone application and to upload the iPhone application into an actual iPhone device for testing purposes (debugging, performance, durability, etc).

B. Once a final version of the iPhone application is finished on the Mac workstation, it can be uploaded to the App Store for distribution to other iPhone devices.

C. Any iPhone with an iTunes account download a working version of the iTour application directly from the online Apple App Store.

(2) The iPhone connected to the blue arrows (starting from top left of figure 8) encompasses the elements involved during the touring activities of an iPhone client using the iTour application.

A. The iPhone application requests XML pages from the iTour web server for available tours and points of interest information. The web server communicates with the database server to obtain this data. The database server responds sending back the data. Then the web server posts the XML pages featuring the requested data. This is then retrieved by the application.

B. The iPhone receives a signal from GPS satellites which triangulate its location and return its coordinates back to the device (if GPS satellites are not reachable it then switches to a Wi-Fi hotspot in range, if this fails then it contacts the carrier's cellular towers to get the most accurate location). This functionality is given by the device, its software, and the service provider. The application developed in this capstone project only calls functions within the device's OS that give access to this information.

C. The application can request web pages (via http) based on content retrieved for all POIs and display them on the device.

(3) Finally, the dotted line on the bottom right part of the model depicts that, using any web browser, administrators can access the iTour website (currently hosted on a CSIS virtual machine) that will allow them to create, read, update, and delete (CRUD) POI information on the database server.

3.5 Actor Diagram and System Components

The actor diagram (figure 9) is used to "define needs and intentional relationships of the business actors or users of a system" [18]. The actor diagram presents a system using the system components (placed in the middle of the diagram) and actors (placed on the outside borders of the diagram). Actors "are users of a software product" [18]. They can be human or other computer systems. Components are the depicted parts of a system with which actors interact.

The iTour system is architected as the composition of a number different components including, the system administration, system reports, tour reports, tour management, and touring. The scope of this project however, is only concerned with two of these fundamental components, those required to make the system function for visitors: tour administration and touring. The other components can be then integrated into the system in the future. A brief description of each component follows.

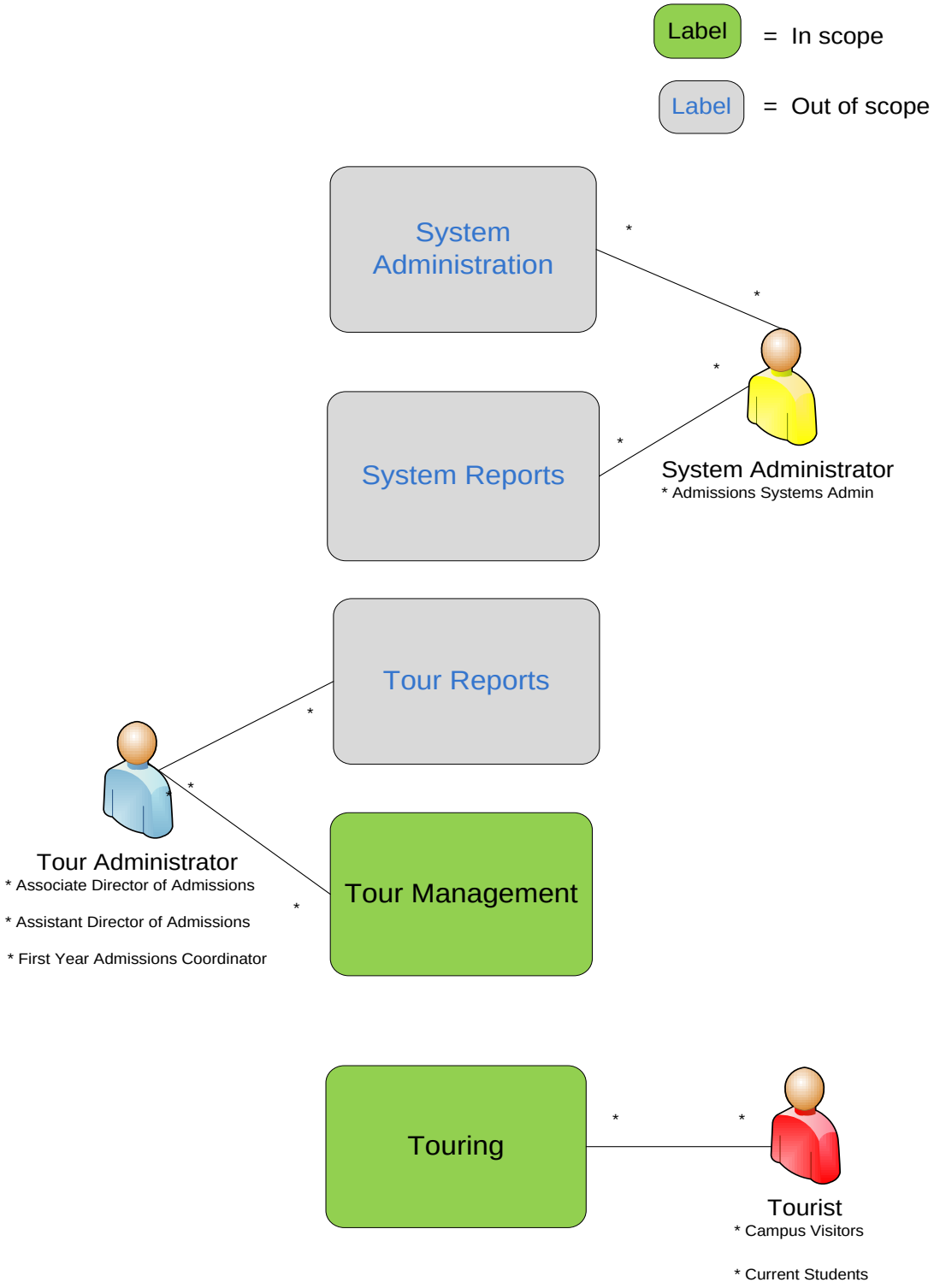


Figure 9 - Actor Diagram

3.5.1 Touring

The focus of this project is on allowing visitors to tour the university campus using an iPhone device. The touring component of the system extracts UNCW tours data in the form of XML files from the iTour web server and makes it available to visitors online. This data is then transformed into useful information through the iTour application. This is accomplished by generating interactive interfaces on the iPhone which display the data in the form of tables, maps, or web pages. Each format, in which this information is displayed, depends on the user's needs and actions.

3.5.2 Tour Management

The tour management component employs as its user interface a set of ASP.NET web pages which are connected to a SQL Server as its database management system. The interfaces are designed to be as user friendly as possible. Tour administrators should be able to use this component to enter and/or modify data about UNCW tours in a simple, quick, and efficient manner. These web pages use standard .Net data output and input elements with data entry validation and SQL server stored procedures on the back end which make the task of creating or updating tours safe and easy.

3.5.3 Tour Reports

The tour reports component allows tour administrators to collect data on users taking campus tours. The iTour SQL Server database model was designed and is currently built so that this tours usage data could be extracted from it. Data such as number of taken virtual tours, most frequently visited points of interest, and the number of devices that have been used to run the

tours can be gathered. Such information could help the office of admissions to measure the effectiveness of the iPhone device virtual tour of campus and make more informed decisions in the future. One example could be the decision of expanding the system to include different mobile device platforms to provide the tours.

3.5.4 System Administration and Reports

The system administration and reports component deals with the usage of the iTour system as a whole. This component would enable a UNCW office of admissions systems administrator to monitor the system primarily and take actions to modify or tweak it when necessary. The envisioned interface for this component would exhibit a dashboard to immediately inform the administrator of the status of the system (load and capacity). From this interface the administrator could have direct access, control, or at a minimum information of those systems elements that consume its resources.

3.6 Use Case Analysis

A use case identifies how a system is employed by its clients which are involved in a single usage event. It can be formulated as a narration of how a user accomplishes a task in a particular scenario [19]. Each use case is divided into the steps taken by the user interacting with the system, and the system's responses to those steps. For the iTour system, three use cases were studied (see table 1).

USE CASE	DESCRIPTION
Touring	Describes how a user would tour the university campus using the iPhone application.
Edit Tour	Describes how a user updates a particular tour's data through the web interface.
Edit Point of Interest	Describes how a user updates a particular POI's data through the web interface.

Table 1 - Use Cases Studied

Each use case has the following entries:

- **Use Case Name** – title of the use case.
- **Scenario** – activity that is to be accomplished.
- **Brief Description** – a short walk-through of the steps a user takes.
- **Actors** – the users (humans or other subsystems) that are involved in the use case.
- **Related Use Cases** – references to other use cases that are relevant to the case described.
- **Preconditions** – prerequisites for the execution of the use case.
- **Post-Conditions** – the system's circumstances that have changed as a result of the use case completion.
- **Flow of Events** – describes in detail the interaction with the system by the actor to accomplish his/her task.
- **Exception Conditions** – error handling conditions due to invalid external or internal circumstances.
- **Alternative Paths** – different activity threads for the use case due to valid variations of action.

The following table shows the use case "touring." This use case describes how a regular iTour user would find information about a point of interest by navigating the application until it accesses the POI's web page. For a list of all use cases refer to Appendix I.

USE CASE NAME	TOURING	
Scenario	Tourist looks for information of a Point of Interest (POI) in the iPhone application.	
Brief Description	The tourist has either obtained an iPhone device with the iTour application pre-installed or has installed the application. The actor starts the application chooses an available tour of the university, and in the map he/she taps a landmark that is of interest.	
Actors	Tourist	
Related Use Cases	None	
Preconditions	• iTour application has been installed on the device • A tour for the respective location exists (e.g UNCW) • A POI for the respective tour exists (e.g CIS building) • Actor is sees a table of available tours.	
Post-conditions	None	
Flow of Events	**Actor**	**System**
	1. Selects a tour from the available tours table	2. Launches the POI view featuring a map of the tour's points of interest (POIs) and the user's location
	3. Taps a POI in the map	4. Displays a description message giving the name of the POI tapped and displays a disclosure button for more information
	5. Taps more info button	6. Launches embedded web browser and display "POI web view" featuring additional info about the chosen POI. Browser has a back, and "BACK TO TOUR MAP" buttons for application navigation
Exception Conditions	1. Database Server unavailable 2. Web Server unavailable 3. User loses internet connectivity 4. iPhone device battery is drained 5. Actor presses the "home" button	
Alternate Paths	1. User receives and takes a phone call 2. User receives and reads a text message SMS	

Table 2 - Use Case "Touring"

3.7 Data Model

Since the iTour system relies on manipulating data that belongs to different entities such as tours, points of interest, administrators, etc, a database system for storage and manipulation of this data becomes necessary. The database should enable adaptation to several interfaces from which different types of users can interact with. For the iTour system, a Relational Database was chosen due to its capabilities and the nature of the data to be manipulated. In this section a brief description of how the data model for the system was designed is put forward.

Data modeling is the process of producing a detailed database design. This activity is crucial to implement a relational database and avoid costly refactoring during the realization of the system the database supports. Although the iTour system only handles a relatively small set of data, some thought was put into its design and implementation. A short process of identifying the tables (or relations) and their columns (or attributes) was executed. In addition, the database was aimed to be normalized to the 3N form. Normalizing a database to 3N form, in short terms, means its information entropy is reduced enough for flexible data manipulation.

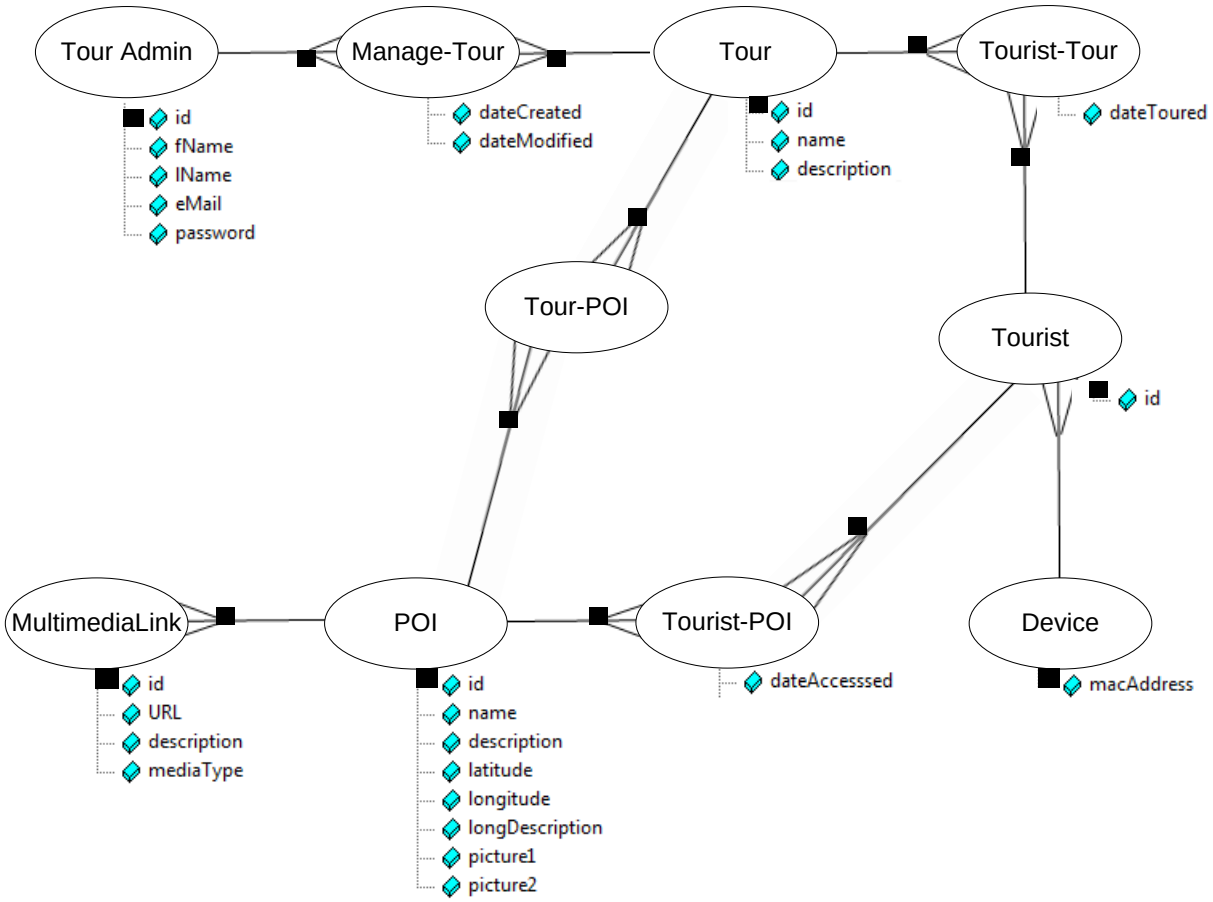


Figure 10 - Data Model for iTour Application

The figure above, presents an initial entity-relationship diagram which specifies each table as an oval, with connecting lines that indicate relationships, and words connected to vertical lines that depict the table's attributes. Each table denotes its primary key with a solid square in front of the first attribute or attributes' names. Solid squares that are connected to relationship lines indicate that the primary key of the further connecting table is a foreign key on the most adjacent table. Also, although system components such as tour reports or system administration are out of scope, they were taken into account for the development of this database model.

3.8 User Interface Design

As mentioned earlier, developing an application for a mobile device is very different than developing for a desktop computer. The developer must consider “the environmental constraints of mobile devices (such as limited memory and processing power) which don't just affect the functional aspects of mobile applications, but also the user interface” [15]. As advanced and capable as the iPhone is; it does not escape the same physical constraints most mobile devices share. Screen size, available memory, CPU, and bandwidth affect UI (user interface) design at different levels.

Screen size is the most notorious hardware concern that impacts user interface design of mobile applications. The iPhone screen measures 320 by 480 pixels of workable UI area. Due to limited screen real estate, the UI should present only relevant information in a simple and well organized manner. Similarly, memory and CPU processing power for the iPhone are limited. Managing memory well in the application is essential. Memory is of concern “because the iPhone OS virtual memory model does not include disk swap space, the developer must take care to avoid allocating more memory than is available on the device” [10]. Another issue occurs in managing processing power. Apple encourages iPhone developers to stick to the window paradigm of “only one view at a time.” What this means is that although iPhone users can access several screens in an iPhone app, they do it sequentially and never concurrently. In conjunction with the “only one application at a time,” (which allows only one application running at a time) these paradigms reduce the amount of processing power consumed by the device at a certain moment. One last hardware consideration while designing the user interface of the iPhone application is bandwidth and/or connectivity. In the Human Interface Guidelines (HIG: a document Apple provides iPhone developers as rules to follow when programming applications)

it is specified that the user must be informed, in a graphical way (using an animated UI object known as the activity indicator) that the application is retrieving data from the internet. It is also advised that anytime an application loses network connectivity or switches between network interfaces that the application take appropriate measures to avoid overusing bandwidth depending on which network is used (Edge, Wi-Fi, or 3G). For more information, see [10].

From these hardware considerations and software capabilities of the iPhone, several design best practices can be followed. The UI design of the iTour application, strived to comply with as many of these considerations, while keeping the application's requirements in mind. In essence, when designing interfaces for iPhone applications, one must dedicate each UI to address a specific need in the simplest way possible. The design should aim to make the application extremely easy to use. In Figures 11 and 12 two of the developed user interfaces for the iTour application are shown. The former is used for displaying the available tours of campus and the latter for showing POIs (belonging to a specific tour) on a map. The following tips and suggestions summarize important points to make a simple yet effective iPhone application interface:

- Have a well organized workflow where users are cascaded from general information to specific information.
- Make the layouts for each screen clean and uncluttered.
- Provide finger size targets (44 pt square) easy to tap buttons.
- Minimize zooming and panning.
- Minimize text entry and instead creating lists for selection.
- Ensure consistency across screen (buttons, colors, sliders, fonts, etc).

Figure 11 - Available Tours Interface View

Figure 12 - POI Map Interface View

Chapter 4: Development & Implementation

In this chapter the implementation aspects of the iTour software system are examined. This includes a presentation of the resulting documentation from user requirements (system sequence diagrams, actor diagrams, system components architecture) and a description of the activities followed for the development of the software. Next, the steps taken during the development of the iPhone application are discussed, and an analysis of the structure and role of this component in the system is presented. A similar analysis is made about the web server program component that manages campus tour content (e.g., images, audio files, video clips, etc). This component was loaded with data for each point of interest defined by the UNCW Office of Admissions (see table 3), and includes a tour administration website to update and manage the content. Finally, specific implementation decisions are discussed.

Point of Interest	**Buildings and Landmarks**
Historic Quad	James Hall, Hoggard Hall, Alderman Hall
Front of Campus	Kenan Hall, Kenan Auditorium, Deloach Hall, Westside Hall
Randall & Campus Commons	Campus Commons, Bear Hall, Randall Library, King Hall, Morton Hall, Leutze Hall
CIS Building	CIS Building
Chancellor's Walk	Social & Behavioral Sciences Building, Cameron Hall, Dobo Hall
Cahill Drive	Friday Hall, Friday Annex, Cultural Arts Building, New School of Nursing Building
Education Building	Education Building
Price Drive & Rec Center	Wagoner Dining Hall, SRC
Fisher University Union	Fisher University Union
Fisher Student Center	Burney Center, Warwick Center, Fisher Student Center

Table 3 – Points of Interest Buildings and Landmarks

4.1. Description of Software Development Activities

The methodology used for the implementation of the iTour system followed the principles of the Unified Process Systems Development Lifecycle (SDLC). The main reason for this decision was the flexibility it offered and its capacity to adapt to changing requirements and resources (see section 3.2 for explanation). A description of the activities on each phase of the project follows.

4.1.1. Inception Phase

The goal during the inception phase is to "develop and refine a vision for the new system to show how it will improve operations and solve existing problems" [19]. This phase involved creating the business case, recognizing key requirements, defining scope, and producing estimates of schedule. The first iteration (I1) comprised the inception phase for iTour.

Iteration I1. This iteration's basic activities, such as making the business case and defining the scope can be found in chapter 1 of this document. In summary, two main benefits of the iTour system were established: greater availability and flexibility for UNCW visitors to take campus tours and standardization of delivered content from the UNCW admissions office to visitors.

The monetary value and cost analysis of elaborating this system for the university was left undefined and out of scope due to time constraints and the limited data that could be gathered about the system's usage. Yet the developer and equipment expenditures of developing this project for the university were estimated to be minimal. iTour would be developed for free as part of a student capstone project. Membership to the Apple Developer University Program was free and provided by a joint initiative of the Computer Science Department and Information

Technology Systems Division of UNCW. This membership provided access to the SDK and programming tools necessary to write and deploy iPhone applications. Additional software and hardware resources for hosting the web and database servers were obtained from the Information Systems and Operations Management Department and its existing virtual machine farm infrastructure.

The scope of the project was also defined during this phase. Through interviews with UNCW admissions office personnel, and an analysis of currently provided physical tours, key requirements for iTour were identified. It was determined that each location visited during a tour, should be easy to locate and should present relevant information about the particular point of interest to the visitor. Additionally, it was determined that all content to be used by the application, would come from UNCW and be managed by the admissions office staff.

As a result two main use cases were created: "touring" and "edit POI" (Appendix I). In short, the scope of the project would entail the creation of two complementary software products. These products were: the iPhone iTour application for visitors and the iTour web content management website for admissions administrators.

To finalize this iteration, an ambitious development schedule was proposed. This schedule indicates the major activities to be carried out for the development of the project.

4.1.2 Elaboration Phase

During this phase of the project, concentration was centered on the finalization of analysis and design activities, elaboration of the core architecture of the system, as well as the initial implementation of the major components of the system. More details about the analysis

and design activities involved during this phase can be found in chapter 3 of this document. The elaboration phase consisted of six iterations described below.

Iteration E1. The first elaboration iteration focused on the analysis, design, and initial implementation of the database that would store all content and allow the manipulation of data used by iTour. Users involved in the process managing tour content, together with basic information available to visitors taking physical and online tours were identified and became part of the data model. See Appendix J for the final version of the iTour database design diagram.

It was decided that tour administrator "Tour-Admin" would have the capability of managing several tours and that a tour could be managed by several tour administrators. Supported activities would consist of creation, modification, and deletion of tours and its content. A tour itself would contain several points of interest (POIs), which among other information could have several multimedia links. A POI possesses the basic tour information accessed by tourists and updated by administrators.

Entities and relationships that would support the generation of reports for the tour administrator (out of scope) were also defined but not used. The main entity that allows this reporting is the tourist. The tourist could take several tours and visit several POIs within those tours. A tourist would employ one device to take the tour, but one device could also be used by one or more tourists.

After the database design was completed, the original user interfaces for both, iPhone application and content website were drawn (see Appendices G and H). Additionally, methods of communication between the front ends of the system and the database were studied. For the iPhone application, it was found that the easiest way to communicate would entail using a web service. This method involves a simple http call from the device to a web service which would in

turn connect to the database to return the data. The data is returned in XML form which is then parsed to create tour objects. Direct communication from a device to a database server is achievable, but not supported by Apple; hence it was discarded from consideration. For the web site application to communicate with the database, connections would be opened using the available connection pools provided by ASP.NET tools. After the design of the database was completed, user interfaces were created, and communication decisions made, the project was ready to move to the next iteration.

Iteration E2. This iteration followed the design activities performed during iteration "E1." Here setting up the virtual machine database and web application servers was completed. Initially, both database and web application servers had been planned to reside in a physical machine. However, it was later decided that for mobility and safety considerations these servers would be hosted on an independent virtual machine server. In addition, open-source project management software known as "Trac" was installed and configured on the virtual machine. The software presents a web interface, accessible from any browser, from which the developer can manage the realization of the project by generating work tasks known as "tickets" and allowing revision control of code written. For more details about "Trac" see section 4.3.

Having the running database server then allowed the realization of the iTour database (tables, relationships, and attributes specifications). With it, a first basic set of stored procedures to retrieve elemental data from the database was created and tested. For this purpose the database tables were populated with a combination of real-world and mock data.

Finally the web site's user interface for logging in was created and tested. The login page of the website helped in checking, tweaking, and reviewing the database settings and credentials so that it would be functional and accessible to the rest of the web application.

Iteration E3. During this iteration, the initial iPhone application interface views "intro" and "avToursTable" were created (see Appendix G). The application's view "intro" was built to display the application's title, the UNCW logo, and a characteristic picture of the university's campus. This view did not require any action by the iPhone user since it would act as a splash screen. The interface view would be seen for a few seconds and then it would automatically disappear giving way to the "avToursTable" interface view. The "avToursTable" consisted of an iPhone table view that would display the tours available to users. For this purpose and since the iPhone application does not communicate directly with the database server, a dedicated web page (from the iTour website) was created to serve as a bridge of communication between the two. This web page retrieves all the available tours information (tours names and descriptions) from the database and publishes it as an XML feed for the iPhone application (or any other device). The iPhone application "avTours" class was then coded to parse the published XML feed, create "avTour" objects from this data, and populate its table of available tours with this content. Once the complete list of available tours was visible on the iPhone application, the project could move to the next development iteration.

Iteration E4. This iteration referred to the completion of the POI interface view for the iPhone application. After a user taps on one of the available tours displayed in the "avToursTable" view, the "POI" view is pushed to the top of the application's navigation controller stack, rendering it visible. This view is comprised of two sub-views: "mapView" and "avPOIsTable." The reason behind having two sub-views instead of two separate main views is that the information shared between the two is the same; that is, all POIs for the selected tour. As in the previous project's iteration, a new XML feed was created to extract the information from the database for availability to the application. This web page would now feature the POIs data

for those POI's that pertain to a tour. The iPhone application parses this data to create the respective POI objects. These objects' location data is then used to create their map annotations. Each annotation is an object itself which is then displayed on the map's respective location of the "mapView" subview, which is displayed first after tapping the tour on the previous available tours table.

To access the "avPOIsTable" subview, the user taps on a "list of POIs" button on the "mapView" subview. The "avPOIsTable" takes the same list of POI objects created by parsing the POI XML feed and its table is populated with this data. Each cell shows a different POI. The purpose of this interface view is to allow users to quickly find POIs by name. If the user then taps a table's cell with the name of the chosen POI, the application would then redisplay the "mapView" subview with the single POI selected and the user's location for reference.

The creation of the "mapView" and "avPOIsTable" subviews belonging to the POI view together with testing that their appropriate content was being displayed signaled the end of iteration E4.

Iteration E5. The goals to accomplish in this iteration were to implement the iPhone application's "webInfo" interface view, finishing populating the iTour database with tour content data, and creating the web page that would display POI information.

The sole function of the "webInfo" view is to serve as a web browser embedded in the iPhone application. There was the possibility of using the native iPhone web browser "Safari" for this function, but it was discarded. This decision was taken because to access Safari would have required the user to exit the running iTour application. In addition, it was determined that the iTour application's embedded browser would be controlled to only access web pages related to the application. This would prevent the user getting distracted from the tour by visiting other

sites. The browser was created using an iPhone SDK object known as a Web View. The object's methods were adjusted to have the desired behavior and functionality as the application's browser. The browser was set to be able to receive either a POI identification string to display the iTour's web page featuring the respective POI content, or a regular URL string to navigate to the web page associated with it.

Also, during this iteration the content provided by the office of admissions to be featured by the application was organized and entered into the iTour database. Each POI was required to have a small description and pictures of buildings or landmarks. Related and multimedia links featuring additional web pages, videos, or sound clips were also included, but left optional. To finalize this iteration, the creation of the dynamic web page that would show the POI's information was completed. The presentation of this page was optimized to be viewable in the iPhone application browser. Optimizing in this instance meant including only the necessary information to be transmitted, making the size of fonts larger for readability, and adjusting the size of the page to fit the iPhone viewable screen. The page itself consists of a group of ASP.NET user controls that retrieve and display particular sets of data based on the POI identification (poiID) string passed by the application. Once a particular POI's information was viewable from the browser in the iPhone application using the created web page, the project was ready to move to the next iteration.

Iteration E6. This final iteration of the elaboration phase had as its purpose the initial development of the iTour content administration website interfaces. This iteration involved creating the skeleton of the web ASP.NET pages through which a tour administrator could edit tours and points of interest. As a skeleton, no logic, validation, or content was dealt with during this iteration. The only focus rested on the assimilation to the design established earlier during

this phase (iteration E1). For this purpose ASP.NET input controls were used and set on the created web pages. A master page that exhibits the UNCW logo, characteristic colors, and that organized the content of the pages into a particular format was employed to make the application more uniform. The completion of this website for actual content manipulation was left for the respective iteration in the construction phase.

4.1.3 Construction Phase

The construction phase activities might include detailing the system controls such as data validation, fine-tuning the user's interface design and finishing routine data maintenance functions [19]. During this phase, the iTour project went through four iterations. These iterations concentrated mainly on refining the system's user interfaces initially implemented during the elaboration phase. As Apple's new Human Interface Guidelines to create applications surged, this phase of the project was used to re-factor and ensure that written code followed these principles Apple demanded. A narrative of the activities performed during each of the iterations follows.

Iteration C1. This iteration concentrated on improving the iPhone application's startup procedures and the presentation of the introductory user interface views. The main concern addressed during this iteration was the "reachability" element of the iPhone application. "Reachability" is Apple's required application feature used to monitor the network status of the device. Since the iPhone iTour application requires internet access to obtain data from the iTour web server and to navigate the Internet itself, it also needs "reachability" checking. For this purpose the "Reachability.m" class was incorporated to the application and the respective calls to its methods were set to check for network connectivity. When there is no connectivity, the

application was modified to send an alert to the user and disable further navigation. Once a network is available, the application automatically resumes its operations.

With respect to the introductory interface views, two changes were made. First, an additional information subview "infoView" was added. This subview is used to display the developer's information (this common among iPhone applications). The second change was the elimination of the automatic switching between views that existed between the "introView" and "avToursTable" view. This was done to comply with Apple's human interface guidelines which indicate that the use of "splash screens" is discouraged.

Iteration C2. During this iteration the iPhone application user interface view "POI" and more specifically its subviews "mapView" and "avPOIsTable" were revisited. One modification was made to the "mapView" in order to minimize zooming into the visible location of the tour's points of interest. Code was introduced to do the task automatically for the user. An area comprised of the farthest apart map annotations (by coordinates), including the user location, is now calculated each time the user enters the view and zooms in the map automatically. In addition, and after examining the subviews for compliance with the latest version of Apple's user interface guidelines [10] one more change was made. All buttons shown in the "mapView" view were programmed to be disabled when not functional. This only occurs when the view is loaded for the first time and information is being retrieved from the XML feed. With respect to the "avPOIsTable" view, adjustments were not made from its earlier iteration implementation.

Ensuring that the revisited POI view complied with Apple's human interface guidelines marked the end of this iteration.

Iteration C3. Iteration C3 was used to conclude the last details of the iPhone application interface view "webInfo" (see appendices G and H for final detail of user interface). The only detail added to this view was a new "back" button. This button now enables the user to navigate to a previous web page (if available) visited using the browser. As it was done with previous interface views of the application, this view was also examined to ensure that it complied with the latest interface guidelines posted by Apple. Once the iPhone application views were completed, the project could move into checking that the web application interfaces were functional.

Iteration C4. The last iteration of the contruction phase (C4) involved finalizing the ASP.Net pages used for the iTour content administration website. Until this point the web pages had no functionality, error checking, or logic added to them. Therefore the first step taken in this iteration was to add the code necessary to manage activities like session variables control and input validation. This would ensure that the pages could withstand common potential user errors and prevent them early. An intermediary step taken was to develop a small unit test suite (see Appendix F) for the web pages. The unit tests scripts for the web pages were created using the open source product "Selenium IDE" [22]. The intention, besides ensuring that certain potential error situations were accounted for and prove that the web application could correct them, was to gain some insight in the concept of unit testing. The final step was adding the logic in the pages' code to connect with the backend SQL command stored procedures and ensure they worked as expected. These connections would then enable the pages to update and receive information from the database. Once using the website successfully allowed creating, reading, updating, and deleting tours and their content the project was ready to advance to its next phase.

4.1.4 Transition Phase

The transition phase usually counts with one final iteration involved with the final user-acceptance and beta tests, and making the system ready for operation [19]. Although deployment of the iTour system into a production environment was left out of scope, this phase was partially completed in order to acquire user acceptance information from the system's users. The iPhone application is finalized as a beta version, open to adjustments in the future. The transition phase involved one iteration (T1) described below.

Iteration T1. During this iteration a survey of ten questions was prepared to obtain users' feedback about the usability of the iPhone application (see Appendix L). The survey was directed to all users of the system to measure several acceptance factors. University admissions personnel, faculty, staff, and students were asked to use the system and provide feedback that highlights its performance and usability aspects. Among these evaluation factors for the system were included: ease of use, appearance, readability, intuitiveness, responsiveness, completeness, etc. The survey was specifically delivered via a website to several Computer Science undergraduates, ITSD staff, and Computer Science faculty. Survey results are described in detail in section 4.4.

4.2 Model-View-Controller Architecture

In order to understand how all the iTour subsystems fit together as a whole, it was recognized that the system, at a high level, adapts to the Model-View-Controller (MVC) paradigm. The MVC is a design pattern that serves as a way of separating the system's data and rules to access that data (Model) from the parts concerned with presenting information to the user (View). The part that maps the user actions performed in the view to updates in the model is

called the Controller. Advantages from following the MVC pattern are derived from the modularity that is introduced into the system's architecture. MVC allows a system to be changed or extended easily by recognizing the correct components that need additional work [13].

The MVC design pattern divides the iTour system into four components or subsystems: iPhone and browser clients as views, the web server's data (source and administration) applications as controllers, and the database server as the model (see figure 13 below).

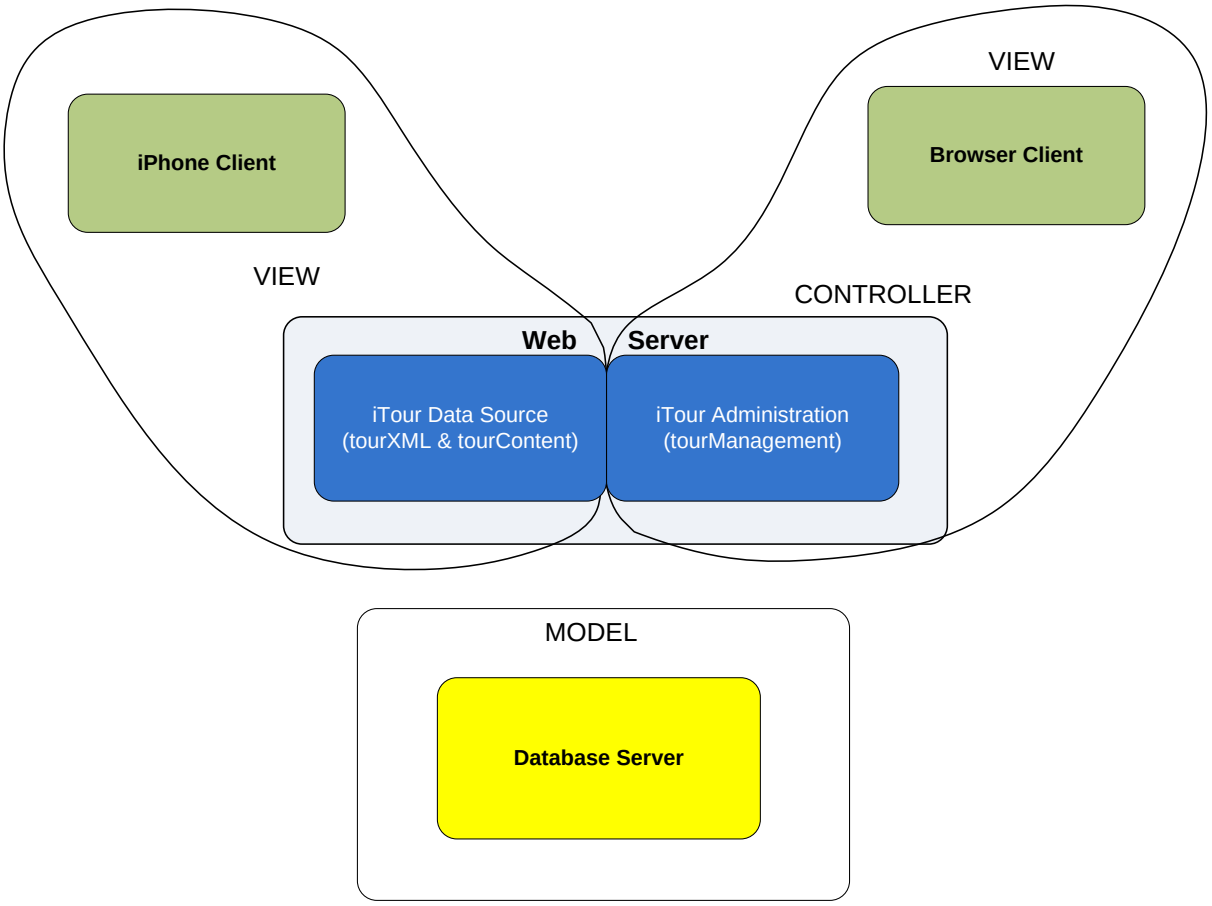


Figure 13 - iTour System MVC

The iPhone client and the browser client act independently of each other, yet they share the database server (or model) to retrieve and/or update data. The iPhone is clearly separated from the browser client. On one hand, the iPhone only accesses the model to retrieve and display

data for the user. For this it uses the “data source” part of the web server. On the other hand, the browser client uses the “administration” part of the web server as its controller, to not only retrieve, but also to send messages and modify data within the database.

Details of each of these individual subsystems and their interactions are given in the following subsections.

4.2.1 iPhone Client

The diagram of figure 14 represents the iPhone client portion of the iTour system. The diagram shows the interactions between the iPhone client, the data source part of the web server, and the database server (observed in the left side of figure 13). This follows from the implementation of the actor diagram component “touring” (see figure 9). The sequence of events starts once the iPhone iTour application launches. The application enters its first view package denoted as “Root Package.” This package makes a call to the web server requesting the “AvTours” (available tours) XML web page (see Appendix E). The web server then runs a stored procedure in the database to retrieve the respective data. The query results are sent back to the web server, which populates its “AvTours” page in XML form. The iPhone application’s root package then parses this information to populate and display it as a list in its “AvToursTable” view. Once a user selects a tour from the shown table in the iPhone app, the package sends the tour selected identification (tourID) to the next iPhone application package (POI Package). With this information, the new package requests the POIs (points of interest) XML web page from the web server. Again the web server runs a stored procedure in the database to obtain the points of interest data. Once the page is populated, the POI package of the iPhone application parses the information from the posted XML page and displays it in its two views “mapView” and

"AvPOIsTable" for the user. The user selects one POI from the map or table views. The selected POI identification (poiID) is then passed to the browser package. The browser package this time requests the POI html page from the web server. This in turn runs the respective stored procedures on the database. Once the page is posted, the "webInfo" view of the iPhone application browser package renders the html page for the user.

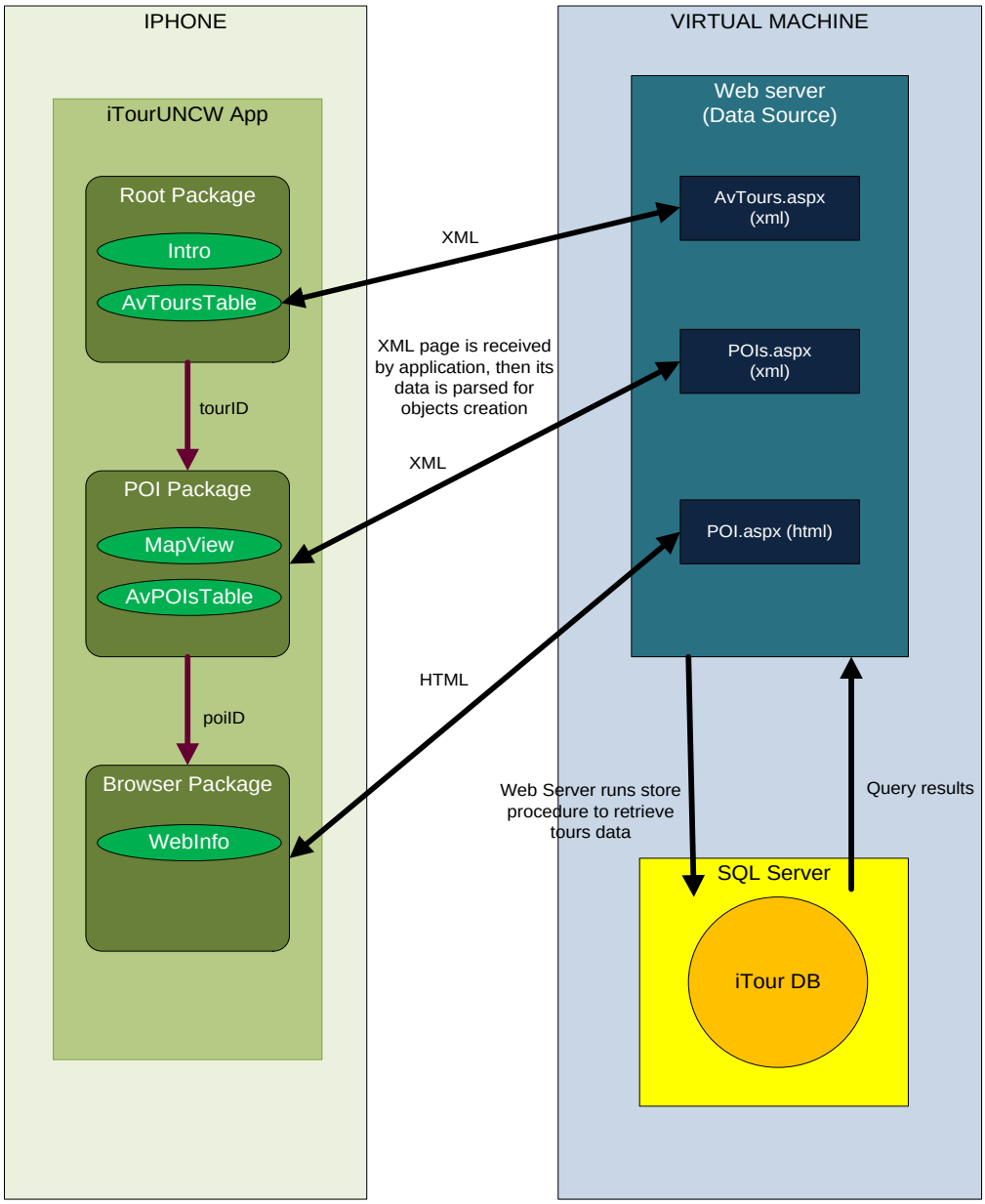


Figure 14 - iPhone Client System Sequence Diagram

To further examine the internal arrangement of the components presented in the MVC diagram, the multitier architecture of the main elements within each portion of the iTour system is presented next.

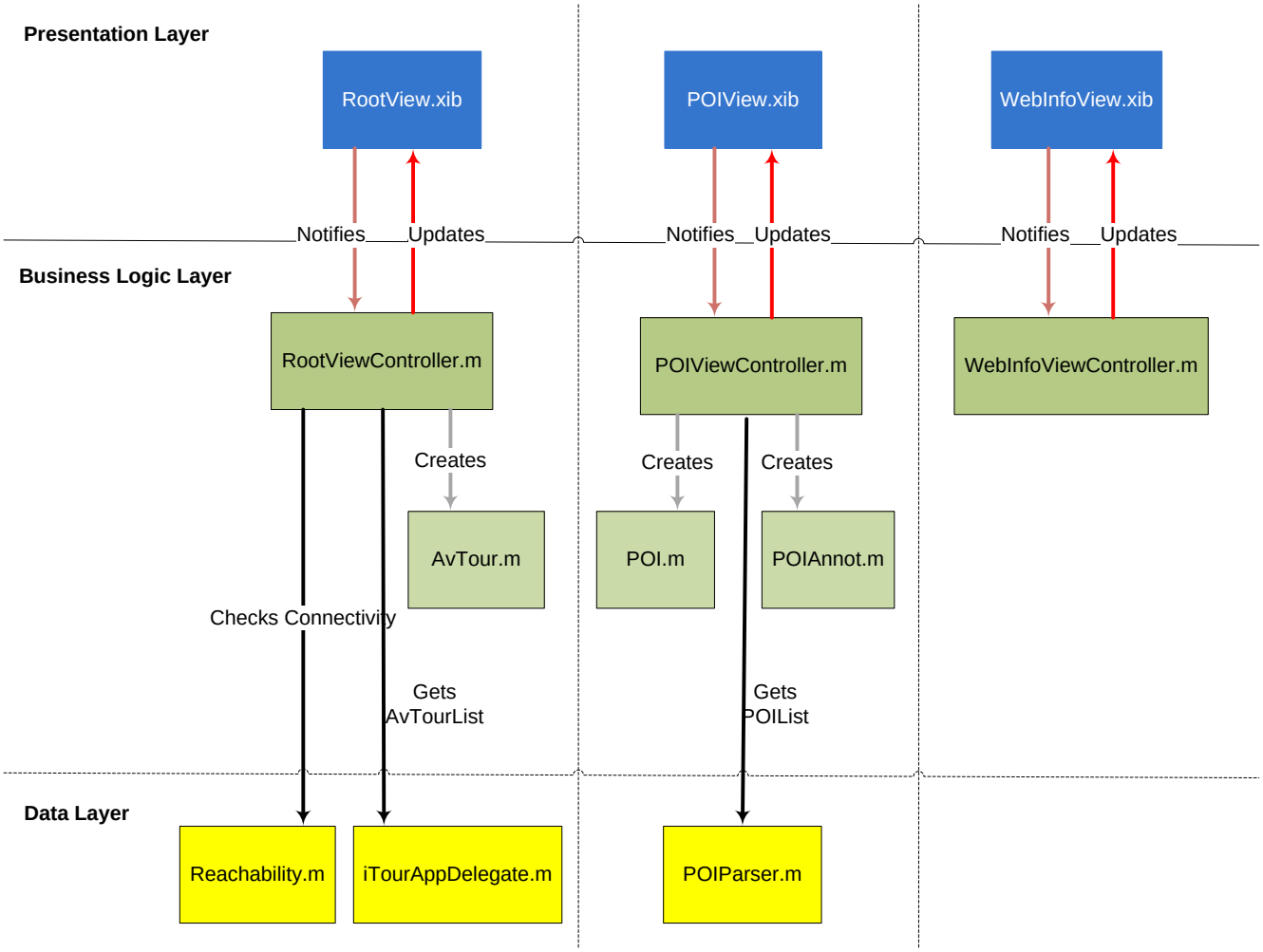


Figure 15 - iPhone Client Multitier Architecture

As shown in figure 15 above, the iPhone client was developed as a multitier application. The presentation layer contains the primary views that handle all presentation logic, multi-touch activities, and iPhone user controls on the screen. The business logic layer handles the information exchange between the data layer and the user interface. RootViewController.m,

POIViewController.m, and WebInfoViewController.m are the primary classes that control the updates sent to the user interface and receive notifications from actions in the presentation layer. Also in this layer, the necessary objects are created and used by the controllers to process the information requested. Finally, the data layer contains the network reachability code, iTourAppDelegate and the POIParser. These classes deal with getting information from external sources to the iPhone application. The reachability class checks if a connection to the internet is available, and if it is, from what type of network (Edge, Wi-Fi, or 3G) it has been acquired. The application delegate and POI parser classes mainly deal with extracting the tour's information from the iTour web server.

The data source part of the web server was also designed in a flexible multitier architecture (see figure 16). Two presentation layer classes (AvTours.aspx and POIs.apsx) are used to generate XML with the data received from the data layer. Similarly, an additional presentation layer class (POI.aspx) processes information received from the data layer, but presents it in html format.

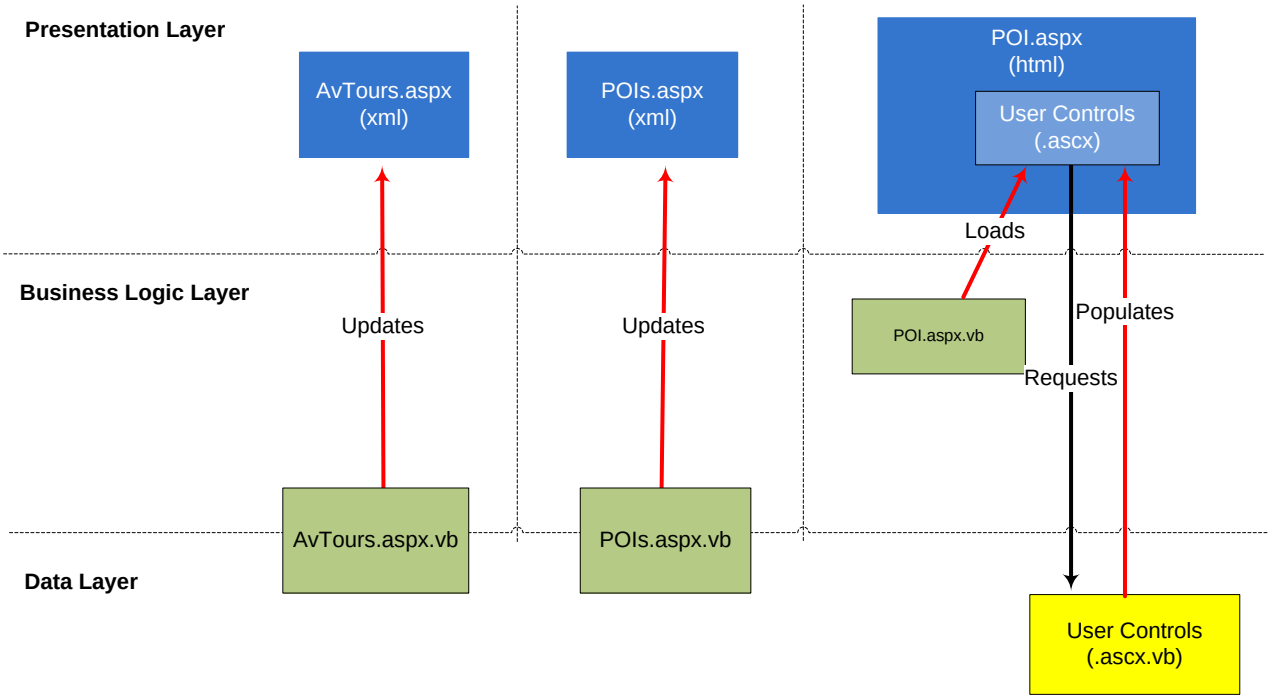


Figure 16 - Data Source Multitier Architecture

4.2.2 Web Browser Client

The diagram of figure 17 represents the browser client portion of the iTour system. The diagram shows the interactions between the browser client, the data administration part of the web server, and the database server. This follows from the implementation of the actor diagram's component "tour-management" (see figure 9). The generic sequence of events starts when the user logs in to the web server (specifically the iTour administration website) to create, read, update, or delete tours content. Once logged in, CRUD actions taken by the user through the browser's interface are then interpreted by the tour administration application. The application calls the database server stored procedures and modify its data. The updated data is also retrieved by the web server running a different set of stored procedures on the database server.

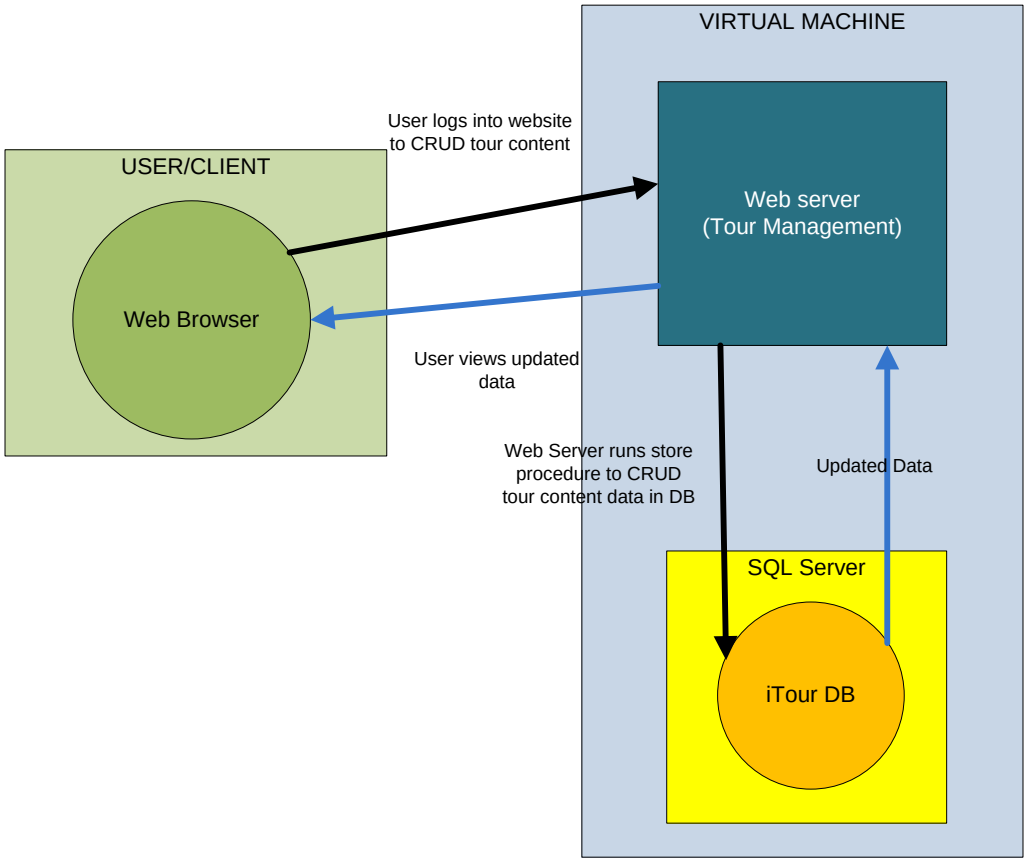


Figure 17 - Browser Client System Sequence Diagram

The "tour administration" part of the web server can be best described in terms of a multitier architecture (see figure 18 below). The presentation layer is comprised of the application's ".aspx" pages that render the html to be read by browsers. These pages solely act as the user interface for the web application. Next in the business logic layer we find the classes controlling the flow of the application, coordinating session variables, and in some cases making connections to the external database. Because of this last point, the majority of these classes also lie on the border between the data and business logic layer.

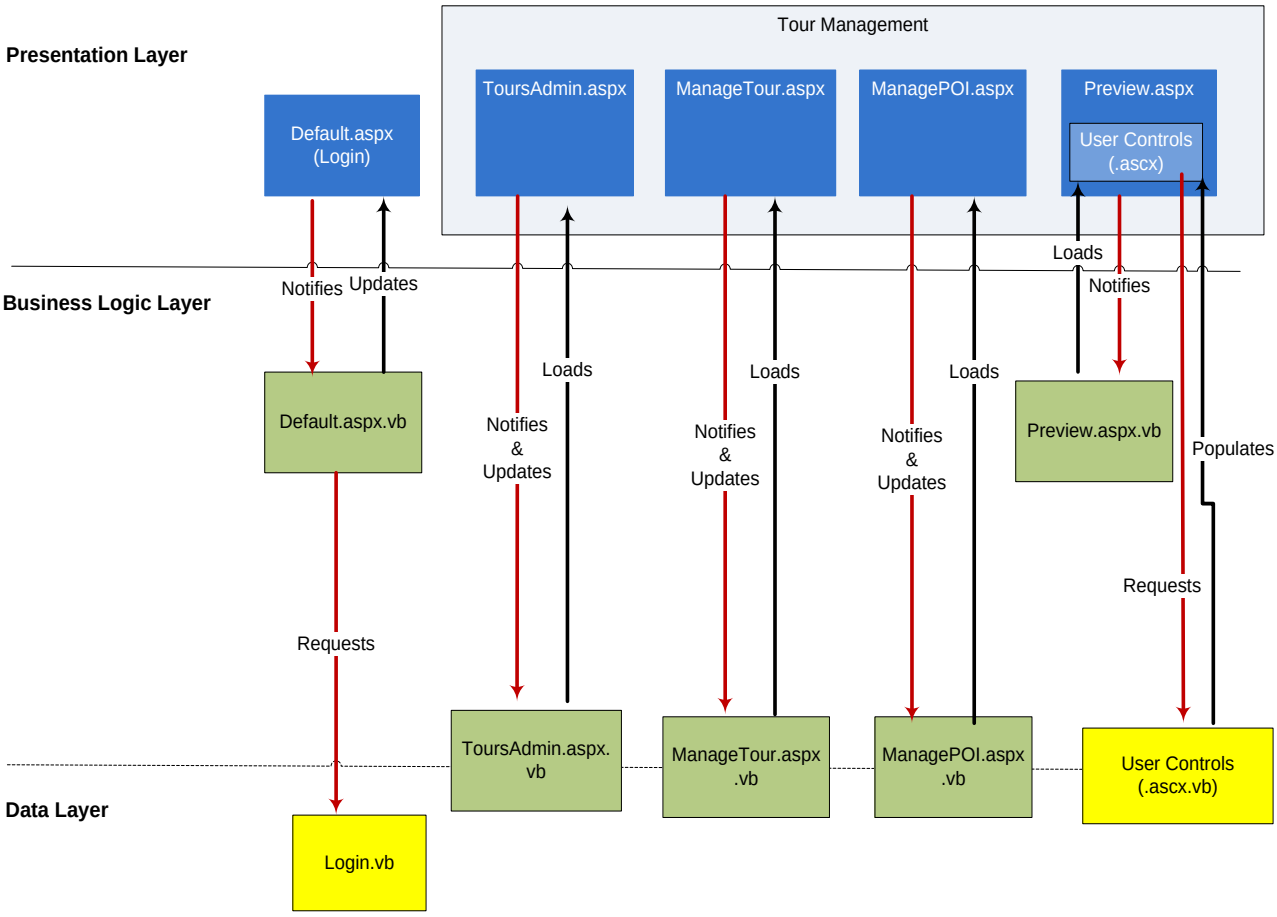


Figure 18 - Tour Administration Multitier Architecture

4.3. Discussion of Implementation Decisions

In this section I discuss a few of the implementation decisions that had a major impact in the development of the iTour system project.

- Apple advocates the use of the Model-View-Controller paradigm for iPhone programming due to the many benefits it brings. Among these it is argued that objects in programs that follow this architecture "tend to be more reusable and their interfaces better defined" [10]. Following the MVC design pattern to develop the iPhone

application, more than a conscious decision, was as a consequence of using the provided SDK tools to implement it. The Interface Builder tool, for example, forces the views to be separated from any controller or model code. Classes created with this tool are only concerned with the way data is presented to the user. Xcode also influences in the adaptation to the MVC architecture by providing pre-made skeleton classes (TableController, ViewController, NavigationController, etc) with empty methods that clearly signal the intermediary role they are supposed to play in the application. Ultimately, to the use of the MVC pattern makes the developed application potentially easier to extend in the future.

- Adding a web component to manage the tour's application content was a fundamental change that introduced flexibility and extensibility to the iTour system. This decision changed drastically the initial project's scope and previously envisioned iPhone application. The application changed from being planned to advertise static content introduced by code, to a dynamic source of information easily updatable from any web browser. The decision was taken as a reflection of the need to change iTour content over the next few years. With planned growth in the university over the next few years, it is expected that more academic departments and buildings will be established, which in turn creates the need for advertising this information to prospective students. If this choice to make the content of the application easily updatable had not been taken, it would have decreased the useful life of the application. This could have happened since once information about the campus changed, the static iTour application would have become legacy.

- The decision of using ASP.Net and SQL Server technologies to implement the website and database server originated from my experience and familiarity with these technologies. I had previously implemented a website for the university using these technologies. This was a multitier ASP.Net application that students of the Cameron School of Business use today to apply online for scholarships. Other web development and database products could potentially been learned and used, but the decision to use familiar products was taken to ameliorate the burden of getting acquainted with more new concepts and tools. As a result of this decision, I was able to dedicate more time and effort in efficiently developing in the unfamiliar iPhone environment and finish the project on time.

- In order to receive the tour's content and populate the objects used in the iPhone application, XML document feeds were employed. In the designed architecture for the iTour system it was decided that the tour's data would reside in a SQL database server. This originated the technical problem of how to extract this data from the database for the iPhone application. Two options were considered to solve this issue. One solution would have been to write Objective-C code to access the database directly from the iPhone application. However, this was rejected because no documentation, samples, or support from Apple to carry out this task were found. The second solution, and the one adopted in this project, was to establish a web service, using a XML feed that acts as an intermediary between the database server and the iPhone client. This solution was radically simpler to implement than writing unfamiliar Objective-C code, and also took advantage of the

existing architecture which already counted with a web server. Code examples for parsing XML feeds were also readily available.

- The decision to present a specific tour's point of interest information (content and format) was reached with the advice and assistance of the UNCW admissions office. The points of interest themselves, were determined by the usual stops made by visitors taking a student guided tour. At each stop the visitors are briefed about a group of buildings and/or landmarks nearby. Pictures, videos, and POI's descriptions were obtained from the admissions office and entered in the system by the developer.

- The project management aspect of this endeavor was aided by an open source, web-based project management and bug control software tool called "Trac" (http://trac.edgewall.org). The initial effort to set up and run Trac was significant. The instructions to set up the software are confusing and some of the links provided are out of date. Each installation and configuration step requires thorough attention to detail; one minimal deviation causes the whole installation effort to fail. Trac uses several other open source technologies to function. These include "Subversion" which helps in storing and keeping track of changes in the source code to be managed; Python, used to successfully employ installation scripts for Trac; and Apache, which is used by Trac to function as a standalone server. In the long run Trac proved a useful tool to manage the iTour's project progress. Although not all of its features were used to their full potential, keeping tasks in order and displaying specific changes in the source code were two greatly useful features in debugging and refactoring code.

4.4. Usability Survey & Evaluation

This section presents the results obtained by the survey prepared for UNCW admissions office personnel, students, and faculty about the usability of the iPhone iTour application (the complete survey instrument can be found in Appendix L). Unfortunately, only 13 respondents participated which is too small of a population to obtain significant product evaluation statistics. Although several requests were sent to the UNCW faculty, staff and student population via E-mail and through social networking mediums of communication, it was impossible to specifically target iPhone users, and hence obtain a larger number of willing participants. Other factors that might have influenced this outcome include, the busy time for faculty and students ending a semester; the application's deployment delayed two additional weeks from Apple's AppStore; and the short time available to gather results. With this in mind however, the results of the usability survey are presented to obtain a sense of the application's acceptance among users (see Appendix M).

Question 1 in the survey was formulated to find if the population had previous experience using the iPhone device. Users who have had more practice with the device may find easier to navigate application since usually applications share similar characteristics. The survey presented 92% of the population had used an iPhone before, hence giving a sign that their evaluation will be more focused on the application itself and not the device. Question 2 was employed to find what percentage and classification of UNCW users rated the application. The end result of this question shows that the majority (62%) of respondents were students.

In order to appropriately compare the iTour application to regular online virtual tours, question 3 was presented. Since 69% of the population responded that they had taken a virtual tour through a website, this gives more confidence of the responses to question 5. Question 5

rates the iTour application (ease of use, information availability, and completeness) in comparison to online virtual tours. The responses to this question show that iTour is, at the moment, equivalent to more traditional online tours.

Question 4 shows the evaluation of the iTour application in itself. On one hand, above of 70% of survey takers rated the application with grades of 4 and 5 (in a scale from 1 to 5) for factors such as attractive appearance, ease of use, intuitive handling, readability and responsiveness. On the other hand the application was rated poorly on the completeness of the tour (55% rated it 1 and 2).

With respect to the application meeting the expectations of the population surveyed, the overall results of question 5 are inconclusive. Seven respondents found that the application met their expectations, while four users responded that it fell short. A larger population size would provide better insight to this question.

Question 7 shows some of the comments of why the application fell short of expectations for some of the surveyed. One of the respondents made the point that visitors should be guided by buildings as points of interest as opposed to general areas of campus, which only people familiar with the university would know. Another two responses to this question complained that the application was slow and unresponsive.

When asked, in question 8, about what additional iTour features respondents would like to see included in the future, some interesting ideas were obtained (see Appendix M). Finally, question 9 results show that 61% of respondents would recommend the iTour application to others, while only 23% would not. Again as the number of respondents was small, it is difficult to assess the success of the application for users. However, these results are encouraging and show that, at least for the participants surveyed, the application achieved several of its goals.

Chapter 5: Conclusions and Lessons Learned

Using the iPhone for the development of the iTour system project lead me to learn numerous lessons about working within the emergent mobile device technology field. These lessons span not only from the iPhone device itself, but also from the previously unknown methods and tools used for application development. Although there was a clear idea of what needed to be accomplished at the beginning of the project, there was some uncertainty about how to proceed and employ the tools necessary to reach the objectives for the project. Several assumptions were made during planning which in some cases proved correct and useful but in other instances resulted in project delays. In this section, I present the lessons learned during the design and implementation of the iTour project, and the conclusions taken away from the approaches used.

5.1. Reflections on Selected Methodology

Although the Unified Process systems development lifecycle presents a detailed framework to complete a project in an orderly manner, it does not account for the resources that are needed for the investigation of new technologies. Therefore, when significant time is needed to investigate an unknown technology, additional development time must be allocated to the project.

It is extremely risky to initiate a project with a hard deadline, when the technologies to be used to complete the project are unknown. Furthermore, constantly evolving and incompatible upgrades (e.g., iPhone OS changes, acceptable guidelines for application approval, etc) make the development task even more difficult. Adapting to evolving and changing technologies is not the same as adaptation to changing requirements of a project. The UP systems development lifecycle

provides developers with the advantage of implementing a project in several iterations. However, this fact helps more in the adjusting and refining gathered requirements than with adapting to constantly changing technologies. In the iTour project, I experienced some backlash from not knowing the capabilities of the technologies being used early in the development process. These capabilities then had to be reviewed and reworked in the later iterations of the SDLC, which increased the amount of time necessary to complete each stage of development beyond initial estimates.

In the end, a clear assessment of how the SDLC methodology helped or hurt, in the iTour system design and implementation, is difficult to make. This is the first time I have developed a project of this magnitude following the principles of a systems development paradigm. To make a comparison with this or other methodologies, about its advantages and disadvantages, more experience in developing software products is necessary.

5.2. Reflections on Previously Considered Implementation Issues

One previously considered issue for the successful implementation of the project was that I (the developer) could have experienced unforeseen circumstances preventing me from working on the project as expected. These circumstances could have ranged from sick time, to a family emergency overseas. Fortunately, I was able to work full time on the project and sometimes exceeded the time previously allocated for these activities. It was fortunate that this was the case because there was little that could have been done to extend the project deadline (i.e. the end of the academic semester). Lack of monetary resources could have also adversely influenced the project's success. Happily, with the help of the university, which acquired some books and equipment for initial development, and Apple's free software development tools acquired

through the university developer program, this turned out to be a non-issue. However, in order to increase my productivity, I purchased a new Mac laptop so that I could spend time working, practicing and writing iPhone code. This laptop had a cost of US $1800 invested by me, the student developer. This was an unforeseen expense, but one that helped in taking the project to its finishing point.

Besides external factors that could have caused the project to fail, there were more internal implementation issues that had to be considered. The first had to do with learning how to use the Mac OS and also how to manipulate the COCOA Touch API for developing the iPhone application. As expected, using the new Mac operating system was not difficult, even after there was a version change during the development of the project. The latest version of the Mac OS is called Snow Leopard (10.6) which facilitated the tasks necessary for using the iPhone SDK tools. In contrast, the COCOA touch API and the Objective-C language used to manipulate it and write the iPhone code were difficult to learn and use. Besides the new syntax for me, there were other issues with Objective-C which took extensive practice and training to become proficient with them. The COCOA touch framework for the iPhone operates in a singular way which is difficult to understand at first. The numerous functions offered by the many frameworks within the COCOA touch API can be overwhelming. And even when the functions desired are found, it is not clear how to use them. On top of these issues, there was also the concern for memory management. The iPhone does not use garbage collection. What this means is that the task of allocating and de-allocating memory used by every object of the application has to be "manually" managed by the developer through special function calls.

Another issue considered was the level of proficiency with which I could use the iPhone integrated software development tools. After much work, and practice with the iPhone, they

fortunately were very useful and certainly easy to use. With the help of several blogs on the Internet that discuss how to tackle coding issues and through trial and error at first, I was able to use the tools to successfully design interfaces, write and debug code, and fine-tune the use of resources in the application.

5.3. Research Questions Revisited

Before work on the iTour project had been started, I formulated three basic research questions to be explored during its development. These questions analyze the overall effectiveness of using the iPhone mobile device to implement and conduct campus tours. They serve as concluding findings for the overall iTour project effort. In order to answer these questions, I based their answers on three sources: the results from the usability and evaluation survey of the iPhone application, personal comments from UNCW admissions office personnel, and the experience gained while developing the system. The questions are found below.

a) Is the iPhone a suitable mobile device to conduct campus tours?

The iPhone is a versatile device. Besides voice communication and internet access, with the great (and growing) number of applications developed for it, it can be used for much more. There are several easily accessible applications that allow users to listen to radio, watch TV, share contact information, play games, etc. All of the functions offered by the iPhone can be combined in different ways to develop innovative applications for other uses such as the iTour for campus tours.

The iTour evaluation survey results are inconclusive (Appendix M questions 6 and 9) with respect to the level of satisfaction with the campus tour taken using the application.

Besides having a small number of respondents, those who thought of the tour negatively compared to those who favored it was very close. Admissions office personnel expressed their satisfaction with the application, but also saw that there can be many improvements to make it a more powerful advertising tool. Despite, the small number of survey respondents, these were chosen randomly and they integrated the key demographic of iPhone users (92%) which have experienced taking college tours (through a website or in person) before applying to one of them (69%).

b) Can an iPhone virtual campus tour, by unifying data with a companion website, serve as an appropriate source of information and support different and incompatible web technologies?

As it was discussed in chapter 1, it was found that the numerous and disparate number of technologies used by virtual tours using solely websites are a detrimental factor to the overall tour's user experience and ease of use. The iTour application however, shows great promise as a university's source of information, first, by unifying data which is available and easily accessible from one place, second, by using technologies which don't require a complex installation or a high level of expertise, and third, by presenting information in an uniform, predictable manner.

From the iTour evaluation survey, we found that 69% of respondents had tried taking a university virtual tour through a website in the past. Next, in the comparison with other online virtual tours, we found that in average 30% thought it was a better alternative to online tours while 25% thought it was worse. We can conclude (at our own risk) that the experience offered by the iTour to unify data in one place for easier access than a website was slightly

better or remained the same. Some of the comments that respondents included about these results remarked that the application needed to present more complete information about the tour's points of interest and include more features to deliver this information. The responsibility of finding and providing more and better sources of information (videos, 360 degree pictures, voice narrations) for future UNCW tours lies in the hands of the UNCW admissions office. The current tour's content, although it covers all points of interest, lacks more depth and organization. Fortunately, the framework to improve the existing material is in place, and hopefully in the future, the application may fare better with respondents.

c) Are the technologies offered by the iPhone device appropriate for implementing virtual campus tours? In particular, are iPhone's built-in GPS capabilities and its Wi-Fi and cellular network connections adequate for supporting the virtual campus tour application?

Based on my own experience while developing the iTour iPhone application, I believe that the iPhone device is an appropriate medium for implementing campus tours. The built-in GPS capability (see section 3.4 for information) helps the iPhone device to serve as a campus tour tool by enabling the tourist to recognize his or her location in unfamiliar places. Together with the use of an application embedded map, the tourist can also visually identify his or her location with respect to surroundings and points of interest from a broader perspective. Depending on the network used by the device, the accuracy and speed of the iPhone's GPS enables the device to find its location within meters in a few seconds. A small defect found during the implementation of this project was that the GPS feature returns approximate, less-accurate location results when the device does not have a clear view of the

sky. This may hinder the iPhone's ability to present a precise location for users while inside a building, hence confusing and frustrating some of them.

Since the application tour's information is retrieved by the iPhone device accessing the Internet, the speed with which this information travels depends on the network used. In October, 2009 AT&T (the only wireless service carrier for the iPhone in the U.S) introduced its 3G network service to Wilmington, NC. Using this network, information is quickly downloaded from the web at rates that vary between 600 Kbps to 1.4 Mbps as opposed to the older EDGE network which had speeds ranging 75-135 kbps [21]. AT&T continues to grow their 3G coverage in the U.S which covers "300 major metropolitan markets" [21]. In addition, the iPhone uses its Wi-Fi capability to connect to the closest hot spots available (when access is granted) allowing the device to download information at broadband speeds. These features allow application to present the user with a fast real-time experience of the materials included on a campus tour through the device.

5.4. Lessons Learned

I learned several valuable lessons during the completion of this project. This section describes some of these lessons along with a discussion of what might be done differently if this project were undertaken today.

1) Before accepting the completion date for a project, it is better to have a clear schedule, with enough time allotted for its successful implementation, than relying on estimates. Although this may be a difficult mission for projects that are using cutting edge technologies, it is worth spending additional time to create a clearer picture of the

activities that will be involved. This does not only help the developer in keeping track of its progress during implementation, but it also helps to allocate and spend time wisely and effectively by concentrating on one problem at a time.

2) The iPhone application approval process can potentially take several weeks. Hence a large cushion time for the application availability on iTunes should be established. Although for this project, I was fortunate enough to have the iTour application approved by Apple within three weeks of its submission, I heard from other developers, stories of applications being delayed up to three months in some cases. Although these delays were in many instances caused by bugs in the software, in other cases they were due to Apple taking longer than expected to review an application.

3) Writing applications for mobile devices is very different from writing applications for desktop computers. Issues like memory, screen size, network bandwidth, and CPU processing power, which are often considered secondary issues for desktop applications, come back to play when developing applications for mobile devices. These devices have limited resources that directly impact the user's experience of an application.

4) One must follow Apple's Human Interface Guidelines (HIG) closely. Even before designing an iPhone application, the HIG should be studied, understood, and followed as rules. Although they are called "guidelines" Apple takes them seriously. An iPhone application that violates a guideline risks being rejected on submission. Rejections, as a consequence, can have a great impact on the deployment date of a product.

5) Using and studying the COCOA frameworks and API libraries early in development facilitates understanding the available functions and features available for programming iPhone applications.

6) Checking memory leaks, using the “leaks” XCode tools instrument, early and throughout the development of an iPhone application can save time and effort in the fine-tuning final stages of the application development process.

7) A related problem to bad memory management is the de-allocation of memory that has already been de-allocated. The objects that suffer this error are called “zombies” and they generate exceptions that crash the application. To identify and correct zombies one must enable the NSZombieEnabled environment variable in Xcode. With this feature enabled, the compiler throws a series of messages that allow zeroing in to the location of these errors for their subsequent correction.

8) Project management software, like Trac, can be helpful in monitoring and keeping track of the many tasks and details involved in the development of a project. For a software developer team of one (as it was in this case) this is especially useful because of the many different aspects and moving parts associated with developing a system of this magnitude. The fact that Trac is a web-based application helps in keeping a central location where to store and retrieve information from anywhere.

One thing that worked particularly well in this project was the separation of the iPhone application from the application content was a great success and worth emulating in the future. This separation allowed the iPhone application to easily grow into a more sophisticated application than initially thought. This separation also extends the useful life of the application by easily allowing changing the content by users without the need of modifying code. Another thing that worked well was the study and understanding of the iPhone user interface guidelines and quick approval of the iTour application were successfully executed and achieved. Apple's approval of submitted applications can be confusing and lengthy at first glance. Apple asks developers to send the executable of the application developed together with its interface files. The black box testing procedures taken by Apple analysts to approve an application are unknown. On top of this, if an application is rejected for, Apple will let the developers know only one factor at a time for the reason of their decision. The only available information about what is and what isn't allowed in an application is found in the Apple human interface guidelines document. This document is a little vague which can lead to misinterpretation of its "guidelines." With all of these adverse factors for the approval of an application, I feared that the approval of my application could be indefinitely delayed. However, the iTour application was approved in less than a month after submitted, and it was only rejected once for a minor bug. At this point I have to give thanks to my fellow undergraduate student iPhone developers, through which I have learned a few reasons why Apple rejects submitted applications. Their experience together with other developers' comments I gathered on blogs in the web allowed me to interpret the HIG faster and more accurately than I could have done by myself.

There are several areas where the project design and implementation could have been improved. Not all software development projects are perfect the first time they are attempted. This is especially true when the developer lacks knowledge about the technology tools that will be used to carry out the project's implementation. Reflecting back on things that could have been done different, I came up with the following important realizations that would improve this or other similar projects.

First, although there is a cost to integrating unit testing into a software system during development, projects ultimately do benefit from using them. These benefits are generally derived from the time savings that unit tests add to the development of the project. Although unit testing was only used partially to check ASP.NET pages basic functionality, it saved time that would have otherwise been used re-testing the same pages later. Unit tests also offer tangible proof that a software product works against certain conditions. This fact provides certain confidence to deploy a software product into a working environment. Having unit tests included in the development of the iPhone application would have proved a difficult undertaking at first, but with the innumerable times that I had to test, and re-test classes and functions, this effort would have proven worthy of taking.

Second, if refactoring the code after each major task had been accomplished, it could have improved the code and made it more maintainable. A clear example where refactoring could be done in the iTour system is offered by the two classes used to parse the XML data from the data source. These classes perform similar tasks, and I believe that with a major effort they could be combined. Currently, the application has two parser classes (AppDelegate and POIParser). These classes could merge into one generic parser class that executes the major and basic parsing functions, and two smaller subclasses to which the more specific details of the

parsed data could be attributed. Now that the application has been fully built, this effort is not trivial, but it would be worth making this change for a future version of the program.

5.5. Future Work

There are several areas that could be investigated for future work. One of the areas involves integrating the iTour application within the family of UNCW applications. What this means is that the iTour application would form part of a single UNCW iPhone mobile application suite that hosts several sub-applications. Additionally, the application would serve as a guide to host all other services that require the use of a map, or location awareness in the campus.

Other features worth considering in the future are:

a) Enabling "tour administrators" to set up guided tours. This would allow the admissions office to have more control over the order in which tourists visit points of interest.
b) Enabling "tourists" to search for specific locations and receiving step by step directions of how to get from their current location to the location they are looking for.
c) Implementing a "sense of direction" based on the movement of the user and display this on the iPhone device. This feature would allow users to have a more clear perspective of their position and surroundings while taking a tour.
d) Integrating bus stops and other locations in real-time. This would allow students to use the application to quickly find the locations of the buses on campus.

References

1. "Blackboard Mobile Platform." Blackboard Inc. Web. October 28, 2009. < http://www.blackboard.com/Mobile/Mobile-Platform.aspx>

2. "Campus Visit Drives College Choice." Art & Science: Higher Education Market Research. January 29, 2004. Art & Science Group, LLC. Web. February 22, 2009. < http://www.artsci.com/StudentPOLL/v5n5/publishers_note.htm>

3. "CampusTours: Virtual College Tours & Interactive Campus Maps." CampusTours Inc. Web. February 10, 2009. <http://www.campustours.com/>

4. "College Campus Tours: What Students Should Expect." *GoCollege.com*. Web. October 15, 2009. <http://www.gocollege.com/admissions/college-search/campus-tours>

5. Dyrli, Ordvard E. "Web-Based Virtual Campus Tours." *BNet*. June 2000. Web. 15 Sept. 2009. < http://findarticles.com/p/articles/mi_m0HJE/is_1_1/ai_65014396/>

6. "Extreme Programming." Wikipedia, The Free Encyclopedia. Web. March 8, 2009. <http://en.wikipedia.org/wiki/Extreme_Programming>

7. *Florida State University Visitor Center*. Florida State University. Web. 7 Aug. 2009. <http://visit.fsu.edu/>

8. Green, Michelle L. "Current students tell you what their colleges won't." *Examiner.com*. 3 Oct. 2009. Web. 26 Oct. 2009. <http://www.examiner.com/x-24637-LA-College-Admissions-Examiner~y2009m10d3-Current-students-tell-you-what-their-colleges-wont>

9. Hockenberry, Craig. "Beta Testing on iPhone 2.0." *Furbo.org*. 6 Aug. 2008. Web. 2 Nov. 2009. <http://furbo.org/2008/08/06/beta-testing-on-iphone-20/>

10. *iPhone Dev Center*. Apple Inc. 2009. Web. November 11, 2009. < http://developer.apple.com/iphone/>

11. *iPhone Developer University Program*. Apple Inc. Web. April 8, 2009. <http://developer.apple.com/iphone/program/university.html>

12. *iPhone OS Enterprise Deployment Guide*. Second Edition. Apple Inc, 2009. Nov 2, 2009. <http://manuals.info.apple.com/en_US/Enterprise_Deployment_Guide.pdf>

13. "Java Blueprints: Model-View-Controller." Sun Microsystems, Inc. 2002. Web. October, 2009. <http://java.sun.com/blueprints/patterns/MVC-detailed.html>

14. Klein, Alana "Reinventing the campus tour: many universities are enhancing the campus tour by better training and paying student tour guides, and utilizing GPS systems and even tram cars". University Business. FindArticles.com. November, 2004. Web. 22 March, 2009. <http://findarticles.com/p/articles/mi_m0LSH/is_11_7/ai_n6367516/>

15. Kontio, Mikko. "Architectural manifesto: Designing Mobile User Interfaces." *IBM Developer Works*. 7 July 2004. Web. 1 Aug. 2009 <http://www.ibm.com/developerworks/architecture/library/wi-arch4/index.html>

16. *NC State Undergraduate Admissions.* NC State University. Web. 7 Aug. 2009. <http://admissions.ncsu.edu/>

17. Rockwell, Janice, M. Moreno. *University of North Carolina Wilmington – Office of Admissions*. UNCW, 2009. Web. 7 Aug. 2009. <http://www.uncw.edu/admissions/visit.html>

18. Schach, Stephen.R. Object-Oriented and Classical Software Engineering (Sixth Ed.). McGraw-Hill, 2004.

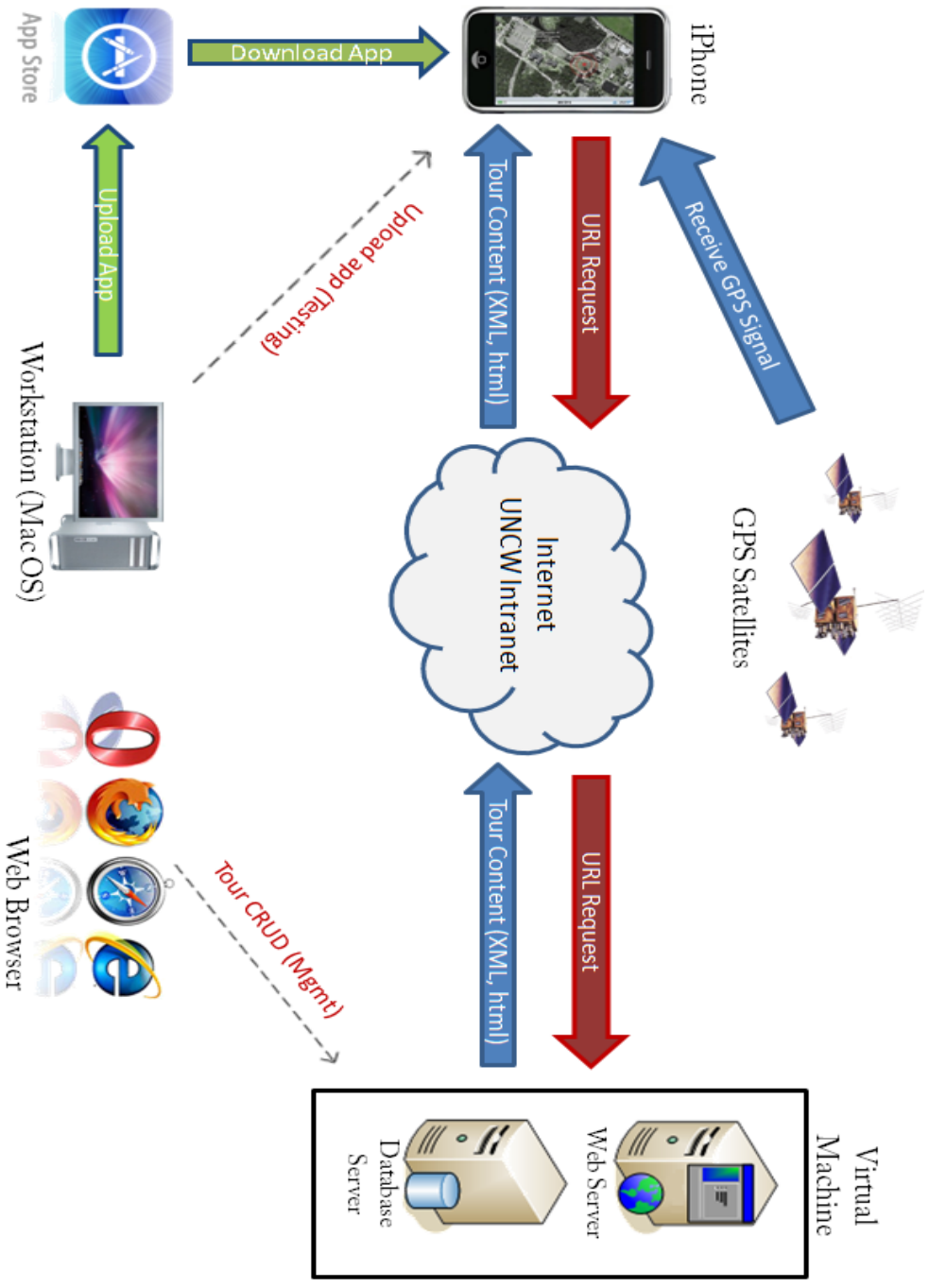
19. Satzinger, Jackson, Burd. Object-Oriented Analysis and Design with the Unified Process. Boston: Thomson, 2005.

20. "Spiral Model." Wikipedia, The Free Encyclopedia. Web. March 8, 2009. <http://en.wikipedia.org/wiki/Spiral_model>

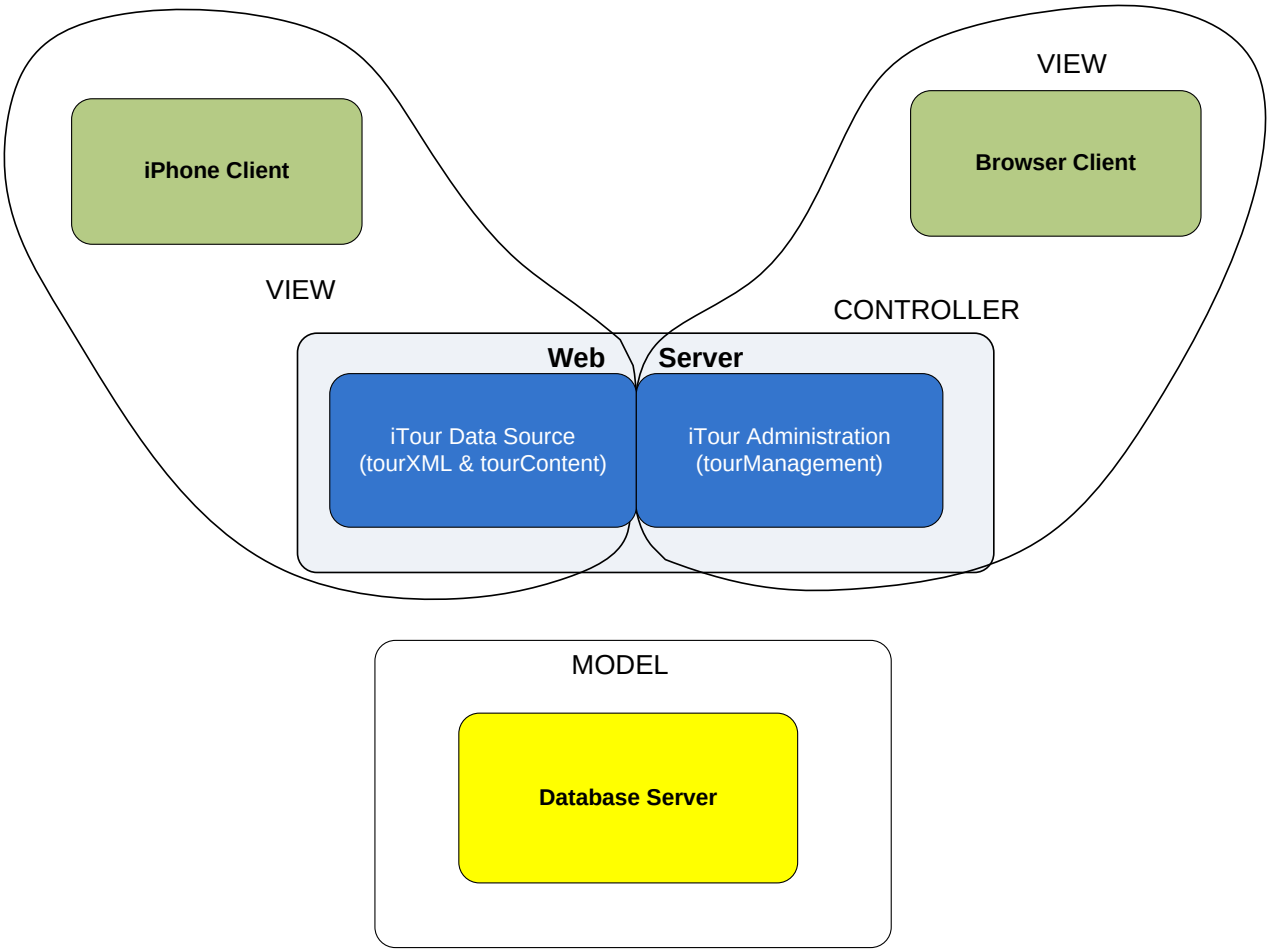
21. "Mobile Phone Technology." AT&T.com. Web. November 29, 2009. <http://www.wireless.att.com/learn/why/technology/>

22. *SeleniumHQ.* OpenQA.org. 2009. Web. December 1, 2009. < http://seleniumhq.org/>

Appendix A – System Architecture

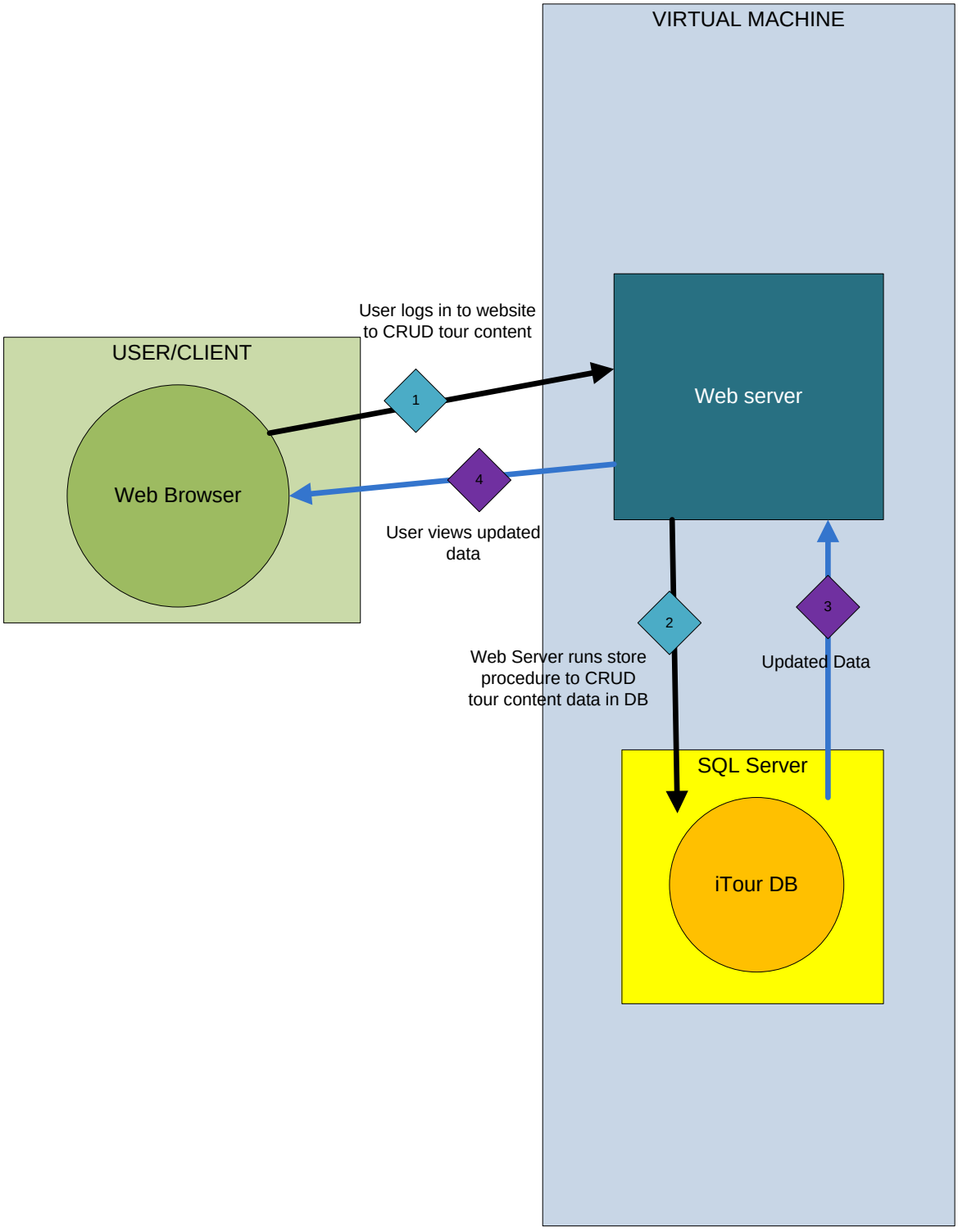


Appendix B – iTour System in MVC

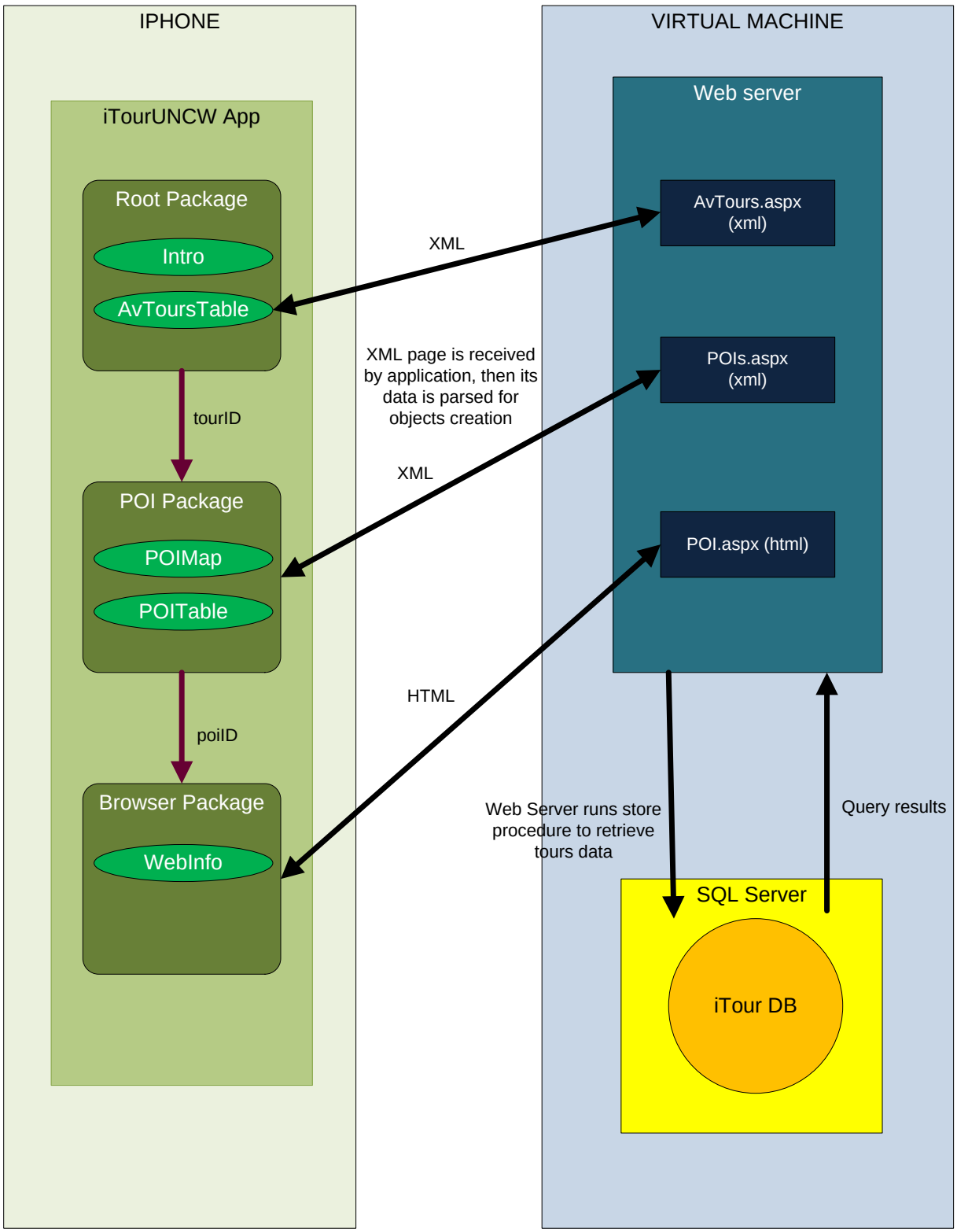


Appendix C – iTour System Diagrams

Browser Client Diagram

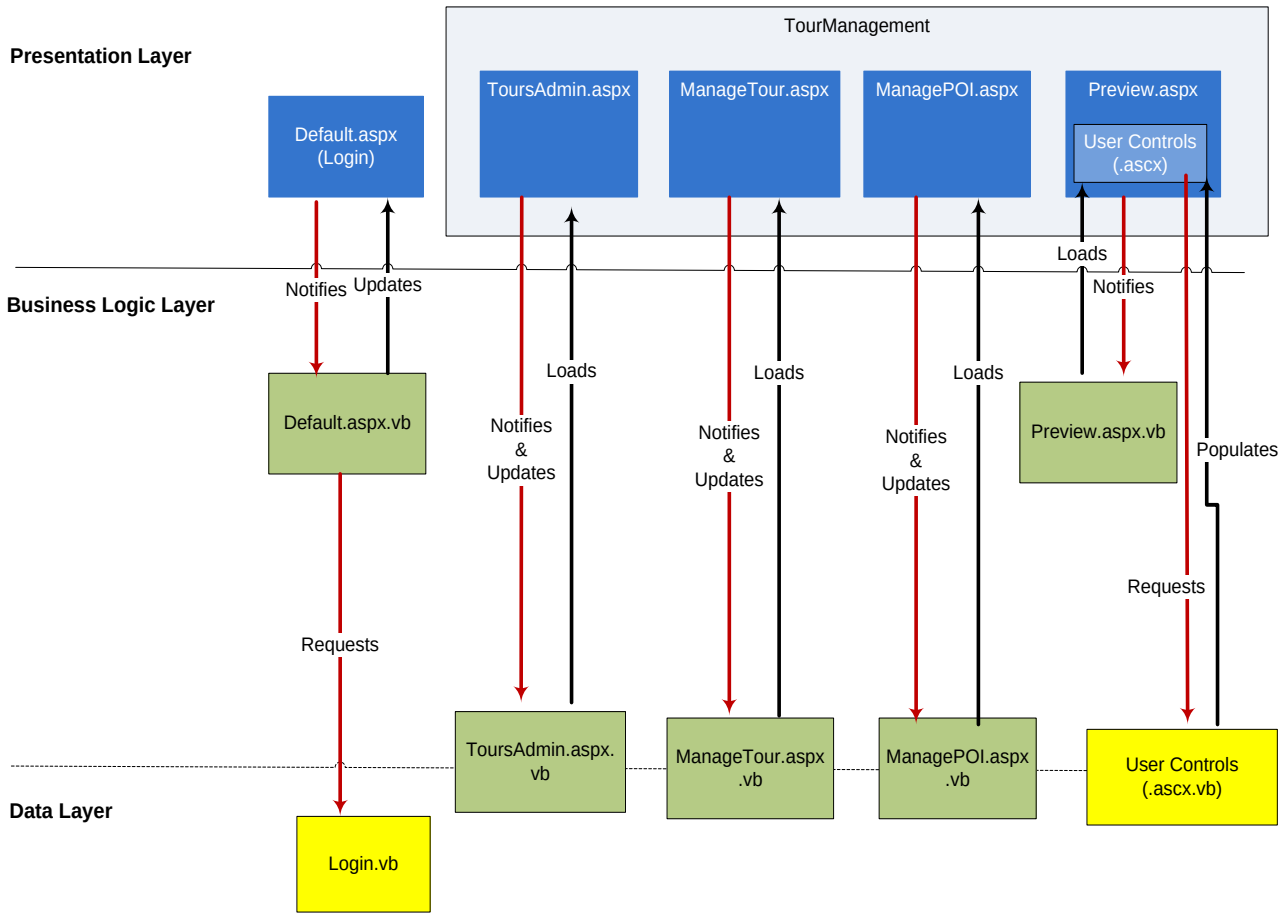


iPhone Client Diagram

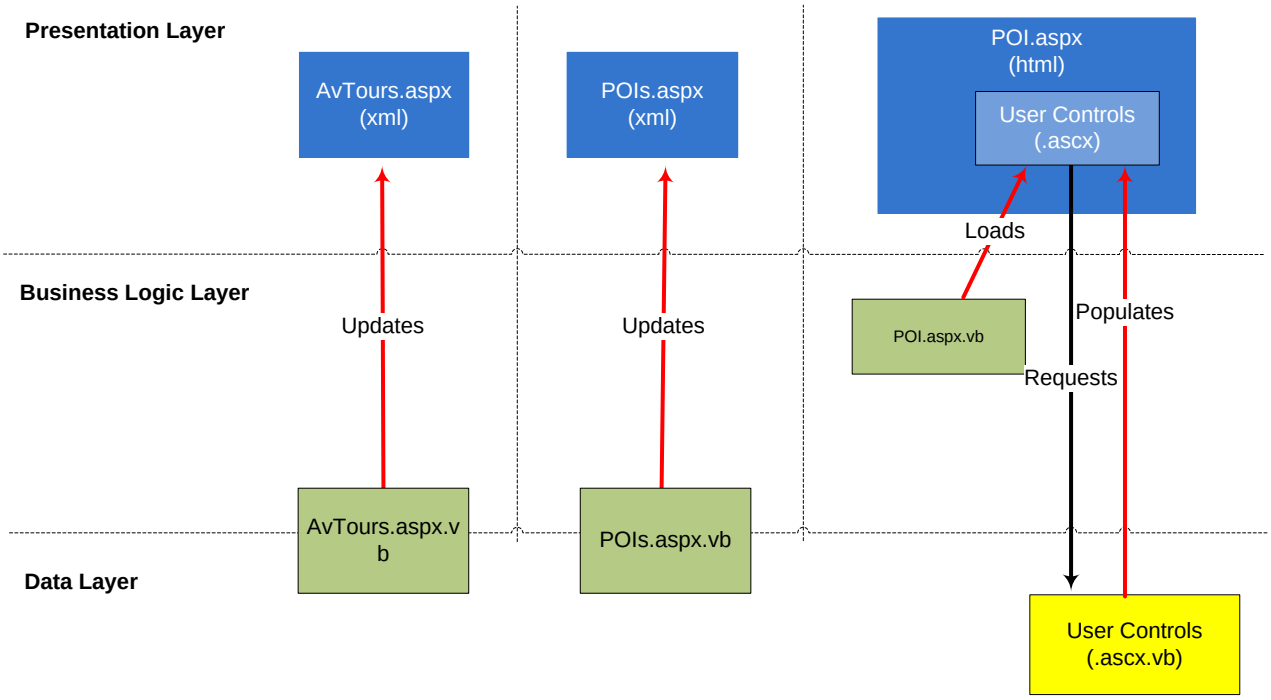


Appendix D – iTour Components Multitier Architectures

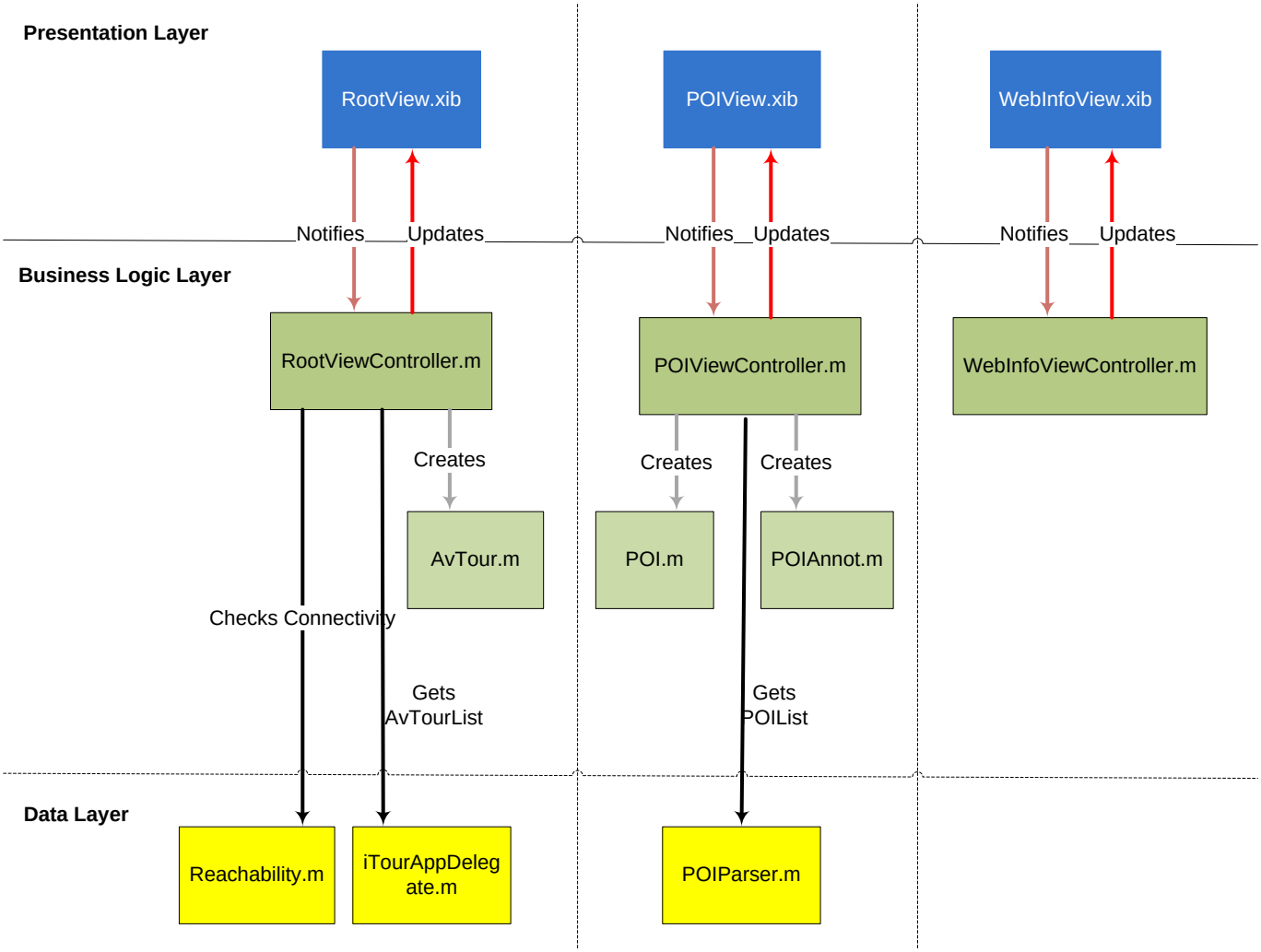
Tour Administration (Web Server)



Data Source (Web Server)



iPhone Client



Appendix E – XML Data Source Feeds

AvTours.aspx (available tours)

```
- <channel>
    <name>Available Tours for location UNCW</name>
  - <description>
      Display available tours        feed to iPhoneApp from SQL Server
    </description>
  - <item>
      <id>11</id>
      <title>Students Tour (Beta)</title>
    - <summary>
        Designed for prospective students who are visiting the UNCW campus.
      </summary>
    </item>
  </channel>
```

POIs.aspx (points of interest)

```
- <channel>
    <name>Points of Interest for Specific Tour</name>
    <description>Display the POIs        feed from iTour SQL Server</description>
  - <item>
      <poiID>1011</poiID>
      <title>Historic Quad</title>
      <summary>James Hall, Hoggard Hall, Alderman Hall</summary>
      <latitude>34.226101</latitude>
      <longitude>-77.876705</longitude>
    </item>
  - <item>
      <poiID>1012</poiID>
      <title>Front of Campus</title>
    - <summary>
        Kenan Hall, Kenan Auditorium, Deloach Hall, Westside Hall
      </summary>
      <latitude>34.227696</latitude>
      <longitude>-77.875873</longitude>
    </item>
```

```
    <poiID>1013</poiID>
    <title>Randall & Campus Commons</title>
  − <summary>
      Campus Commons, Bear Hall, Randall Library, King Hall, Morton Hall, Leutze Hall
    </summary>
    <latitude>34.227463</latitude>
    <longitude>-77.873591</longitude>
  </item>
− <item>
    <poiID>1014</poiID>
    <title>CIS Building</title>
    <summary>Computer-Information Systems Building</summary>
    <latitude>34.226212</latitude>
    <longitude>-77.871788</longitude>
  </item>
− <item>
    <poiID>1015</poiID>
    <title>Chancellor's Walk</title>
  − <summary>
      Social & Behavioral Sciences Building, Cameron Hall, Dobo Hall
    </summary>
    <latitude>34.225491</latitude>
    <longitude>-77.869423</longitude>
  </item>
− <item>
    <poiID>1016</poiID>
    <title>Cahill Drive</title>
  − <summary>
      Friday Hall, Friday Annex, Cultural Arts Building, New School of Nurssing Building
    </summary>
    <latitude>34.226937</latitude>
    <longitude>-77.868519</longitude>
  </item>
− <item>
    <poiID>1017</poiID>
    <title>Education Building</title>
    <summary>Education Building, The Teal Experience</summary>
    <latitude>34.226733</latitude>
    <longitude>-77.86736</longitude>
  </item>
− <item>
    <poiID>1018</poiID>
    <title>Price Drive & Rec Center</title>
    <summary>Wagoner Dining Hall, Student Recreation Center</summary>
    <latitude>34.222901</latitude>
    <longitude>-77.867296</longitude>
  </item>
− <item>
    <poiID>1019</poiID>
    <title>Fisher University Union</title>
    <summary>Fisher University Union</summary>
```

```
      <latitude>34.225784</latitude>
      <longitude>-77.873272</longitude>
    </item>
  −<item>
      <poiID>1020</poiID>
      <title>Fisher Student Center</title>
    −<summary>
        Burney Center, Warwick Center, Fisher Student Center
      </summary>
      <latitude>34.225757</latitude>
      <longitude>-77.874291</longitude>
    </item>
</channel>
```

Appendix F – Admin Website Unit Test Scripts

These test scripts were generated using the open source product “Selenium IDE”

Test: Website basic navigation

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="" />
<title>NavigatingTempTour</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">NavigatingTempTour</td></tr>
</thead><tbody>
<tr>
	<td>open</td>
	<td>/itour/</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxUser</td>
	<td>caa4936@uncw.edu</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxPassword</td>
	<td>123admin</td>
</tr>
<tr>
	<td>verifyTitle</td>
	<td>iTour UNCW</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>User</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Password</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiBtnSubmit</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Tours Administration</td>
```

```
        <td></td>
</tr>
<tr>
        <td>chooseCancelOnNextConfirmation</td>
        <td></td>
        <td></td>
</tr>
<tr>
        <td>click</td>
        <td>ctl00_ContentPlaceHolder1_uiAvToursList_ctl03_LinkButton2</td>
        <td></td>
</tr>
<tr>
        <td>assertConfirmation</td>
        <td>Are you sure you want to delete?</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>ctl00_ContentPlaceHolder1_uiAvToursList_ctl03_LinkButton1</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Manage Tour</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Tour's Name</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Tour's Purpose</td>
        <td></td>
</tr>
<tr>
        <td>chooseCancelOnNextConfirmation</td>
        <td></td>
        <td></td>
</tr>
<tr>
        <td>click</td>
        <td>ctl00_ContentPlaceHolder1_uiPOIList_ctl02_LinkButton2</td>
        <td></td>
</tr>
<tr>
        <td>assertConfirmation</td>
        <td>Are you sure you want to delete?</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>ctl00_ContentPlaceHolder1_uiPOIList_ctl02_LinkButton1</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Manage POI</td>
        <td></td>
</tr>
```

```
<tr>
	<td>verifyTextPresent</td>
	<td>Name</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Longitude</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Latitude</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Coordinates</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Description</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Purpose</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Long Description</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Main Pictures (URL)</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Main Picture 1</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Main Picture 2</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiSubmit</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Manage Multimedia Links</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
```

```
        <td>Related Links</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Multimedia Links</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Link Name</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Media Type</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>ctl00_ContentPlaceHolder1_uiPreviewButton</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Preview POI</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>link=Manage Multimedia Links</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>link=Manage POI</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>link=Manage Tour</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>link=Tours Administration</td>
        <td></td>
</tr>
</tbody></table>
</body>
</html>
```

Test: Create a new tour

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="" />
```

```
<title>CreateNewTour</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">CreateNewTour</td></tr>
</thead><tbody>
<tr>
	<td>open</td>
	<td>/itour/</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxUser</td>
	<td>caa4936@uncw.edu</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxPassword</td>
	<td>123admin</td>
</tr>
<tr>
	<td>verifyTitle</td>
	<td>iTour UNCW</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiBtnSubmit</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Tours Administration</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiCreateTourButton</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Manage Tour</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Tour's Name</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Tour's Purpose</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTourNameTxtBox</td>
	<td>Unit Test &amp;quot;new tour&amp;quot;</td>
</tr>
```

```
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTourPurposeTxtBox</td>
	<td>Unit testing the creation and deletion of a new tour only.</td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiSubmitButton</td>
	<td></td>
</tr>
<tr>
	<td>verifyValue</td>
	<td>ctl00_ContentPlaceHolder1_uiTourPurposeTxtBox</td>
	<td>Unit testing the creation and deletion of a new tour only.</td>
</tr>
<tr>
	<td>verifyValue</td>
	<td>ctl00_ContentPlaceHolder1_uiTourNameTxtBox</td>
	<td>Unit Test &amp;quot;new tour&amp;quot;</td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Manage Tour</td>
	<td></td>
</tr>
</tbody></table>
</body>
</html>
```

Test: Delete existing tour

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="" />
<title>DeleteNewTour</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">DeleteNewTour</td></tr>
</thead><tbody>
<tr>
	<td>open</td>
	<td>/itour/</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxUser</td>
	<td>caa4936@uncw.edu</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxPassword</td>
	<td>123admin</td>
</tr>
<tr>
```

```
		<td>verifyTextPresent</td>
		<td>User</td>
		<td></td>
</tr>
<tr>
		<td>verifyTextPresent</td>
		<td>Password</td>
		<td></td>
</tr>
<tr>
		<td>clickAndWait</td>
		<td>ctl00_ContentPlaceHolder1_uiBtnSubmit</td>
		<td></td>
</tr>
<tr>
		<td>verifyTextPresent</td>
		<td>Tours Administration</td>
		<td></td>
</tr>
<tr>
		<td>verifyTextPresent</td>
		<td>Delete</td>
		<td></td>
</tr>
<tr>
		<td>verifyTextPresent</td>
		<td>Unit Test &amp;quot;new tour&amp;quot;</td>
		<td></td>
</tr>
<tr>
		<td>click</td>
		<td>ctl00_ContentPlaceHolder1_uiAvToursList_ctl04_LinkButton2</td>
		<td></td>
</tr>
<tr>
		<td>assertConfirmation</td>
		<td>Are you sure you want to delete?</td>
		<td></td>
</tr>
</tbody></table>
</body>
</html>
```

Test: Create new POI

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="" />
<title>CreateNewPOI</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">CreateNewPOI</td></tr>
</thead><tbody>
<tr>
		<td>open</td>
```

```
        <td>/itour/</td>
        <td></td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiTxtBoxUser</td>
        <td>caa4936@uncw.edu</td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiTxtBoxPassword</td>
        <td>123admin</td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>User</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Password</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>ctl00_ContentPlaceHolder1_uiBtnSubmit</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Tours Administration</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Temp tour</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>ctl00_ContentPlaceHolder1_uiAvToursList_ctl03_LinkButton1</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Manage Tour</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td></td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Testing</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>ctl00_ContentPlaceHolder1_uiAddNewPOIButton</td>
        <td></td>
```

```
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiPOINameTxtBox</td>
	<td>Unit Test POI</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiLongitudeTxtBox</td>
	<td>33</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiLatitudeTxtBox</td>
	<td>66</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiPurposeTxtBox</td>
	<td>Find the car</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiLongDescriptionTxtBox</td>
	<td>Give the coordinates and pics of my car!</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiPicture1TxtBox</td>
	<td>http://cache.jalopnik.com/assets/resources/2007/12/KITT%20Knight%20Rider%20Mustang.jpg</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiPicture2TxtBox</td>
	<td>http://www.instylecars.com/wp-content/gallery/5/mustang-gt500-kr-3230.jpg</td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Manage POI</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiSubmit</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Manage Multimedia Links</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Link Name</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>URL</td>
	<td></td>
</tr>
<tr>
```

```
		<td>verifyTextPresent</td>
		<td>Media Type:</td>
		<td></td>
</tr>
<tr>
		<td>type</td>
		<td>ctl00_ContentPlaceHolder1_uiMMDesxcriptionTxtBox</td>
		<td>Mustang cars</td>
</tr>
<tr>
		<td>type</td>
		<td>ctl00_ContentPlaceHolder1_uiMMURLTxtBox</td>
		<td>http://www.instylecars.com/mustang/the-most-powerful-mustang-has-ever-created</td>
</tr>
<tr>
		<td>select</td>
		<td>ctl00_ContentPlaceHolder1_uiMediaTypeDDlist</td>
		<td>label=Web Page</td>
</tr>
<tr>
		<td>click</td>
		<td>ctl00_ContentPlaceHolder1_uiAddMMLinkButton</td>
		<td></td>
</tr>
<tr>
		<td>verifyTextPresent</td>
		<td>Manage Multimedia Links</td>
		<td></td>
</tr>
<tr>
		<td>clickAndWait</td>
		<td>ctl00_ContentPlaceHolder1_uiPreviewButton</td>
		<td></td>
</tr>
<tr>
		<td>verifyTextPresent</td>
		<td>Related Links</td>
		<td></td>
</tr>
<tr>
		<td>verifyTextPresent</td>
		<td>Mustang cars</td>
		<td></td>
</tr>
</tbody></table>
</body>
</html>
```

Test: Delete POI

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="" />
<title>DeleteNewPOI</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
```

```
<thead>
<tr><td rowspan="1" colspan="3">DeleteNewPOI</td></tr>
</thead><tbody>
<tr>
	<td>open</td>
	<td>/itour/</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxUser</td>
	<td>caa4936@uncw.edu</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxPassword</td>
	<td>123admin</td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiBtnSubmit</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiAvToursList_ctl03_LinkButton1</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Unit Test POI</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Delete</td>
	<td></td>
</tr>
<tr>
	<td>click</td>
	<td>ctl00_ContentPlaceHolder1_uiPOIList_ctl03_LinkButton2</td>
	<td></td>
</tr>
<tr>
	<td>assertConfirmation</td>
	<td>Are you sure you want to delete?</td>
	<td></td>
</tr>
</tbody></table>
</body>
</html>
```

Test: Potential user mistakes

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="" />
```

```
<title>DumbUserUnitTest</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">DumbUserUnitTest</td></tr>
</thead><tbody>
<tr>
	<td>open</td>
	<td>/itour/</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxUser</td>
	<td>wronguser</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxPassword</td>
	<td>wrongpassword</td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>User</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Password</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiBtnSubmit</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>User and Password combination is invalid!!!</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxPassword</td>
	<td>123admin</td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiBtnSubmit</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>User and Password combination is invalid!!!</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxUser</td>
	<td>caa4936@uncw.edu</td>
</tr>
```

```
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxPassword</td>
	<td>wrongpassword</td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiBtnSubmit</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>User and Password combination is invalid!!!</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiTxtBoxPassword</td>
	<td>123admin</td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiBtnSubmit</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Tours Administration</td>
	<td></td>
</tr>
<tr>
	<td>chooseCancelOnNextConfirmation</td>
	<td></td>
	<td></td>
</tr>
<tr>
	<td>click</td>
	<td>ctl00_ContentPlaceHolder1_uiAvToursList_ctl03_LinkButton2</td>
	<td></td>
</tr>
<tr>
	<td>assertConfirmation</td>
	<td>Are you sure you want to delete?</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Students Tour (Beta)</td>
	<td></td>
</tr>
<tr>
	<td>chooseCancelOnNextConfirmation</td>
	<td></td>
	<td></td>
</tr>
<tr>
	<td>click</td>
	<td>ctl00_ContentPlaceHolder1_uiAvToursList_ctl02_LinkButton2</td>
	<td></td>
</tr>
<tr>
	<td>assertConfirmation</td>
```

```
        <td>Are you sure you want to delete?</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Students Tour (Beta)</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>ctl00_ContentPlaceHolder1_uiCreateTourButton</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Manage Tour</td>
        <td></td>
</tr>
<tr>
        <td>click</td>
        <td>ctl00_ContentPlaceHolder1_uiAddNewPOIButton</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Please enter a name for the tour to continue...</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Please enter the tour's purpose to continue...</td>
        <td></td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiTourNameTxtBox</td>
        <td>No purpose</td>
</tr>
<tr>
        <td>click</td>
        <td>ctl00_ContentPlaceHolder1_uiAddNewPOIButton</td>
        <td></td>
</tr>
<tr>
        <td>verifyTextPresent</td>
        <td>Please enter the tour's purpose to continue...</td>
        <td></td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiTourNameTxtBox</td>
        <td>New purpose test</td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiTourPurposeTxtBox</td>
        <td>Proposito</td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>ctl00_ContentPlaceHolder1_uiSubmitButton</td>
        <td></td>
```

```
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Manage Tour</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>ctl00_ContentPlaceHolder1_uiAddNewPOIButton</td>
	<td></td>
</tr>
<tr>
	<td>click</td>
	<td>ctl00_ContentPlaceHolder1_uiSubmit</td>
	<td></td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>This field is required!</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiLongitudeTxtBox</td>
	<td>900000000000000000000</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiLatitudeTxtBox</td>
	<td>hffjfgjfgfgfh</td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Invalid longitude!</td>
	<td></td>
</tr>
<tr>
	<td>click</td>
	<td>ctl00_ContentPlaceHolder1_uiSubmit</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiLongitudeTxtBox</td>
	<td>56</td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiLatitudeTxtBox</td>
	<td>-3564215890755890-87</td>
</tr>
<tr>
	<td>verifyTextPresent</td>
	<td>Invalid latitude!</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>ctl00_ContentPlaceHolder1_uiLatitudeTxtBox</td>
	<td>77</td>
</tr>
<tr>
```

```
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiPOINameTxtBox</td>
        <td>New POI</td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiPurposeTxtBox</td>
        <td>Check POI</td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiLongDescriptionTxtBox</td>
        <td>Testing Unit testing</td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiPicture1TxtBox</td>
        <td>www.google.com</td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiPicture2TxtBox</td>
        <td>http://www.google.com</td>
</tr>
<tr>
        <td>type</td>
        <td>ctl00_ContentPlaceHolder1_uiPicture1TxtBox</td>
        <td>http://www.google.com</td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>ctl00_ContentPlaceHolder1_uiSubmit</td>
        <td></td>
</tr>
</tbody></table>
</body>
</html>
```

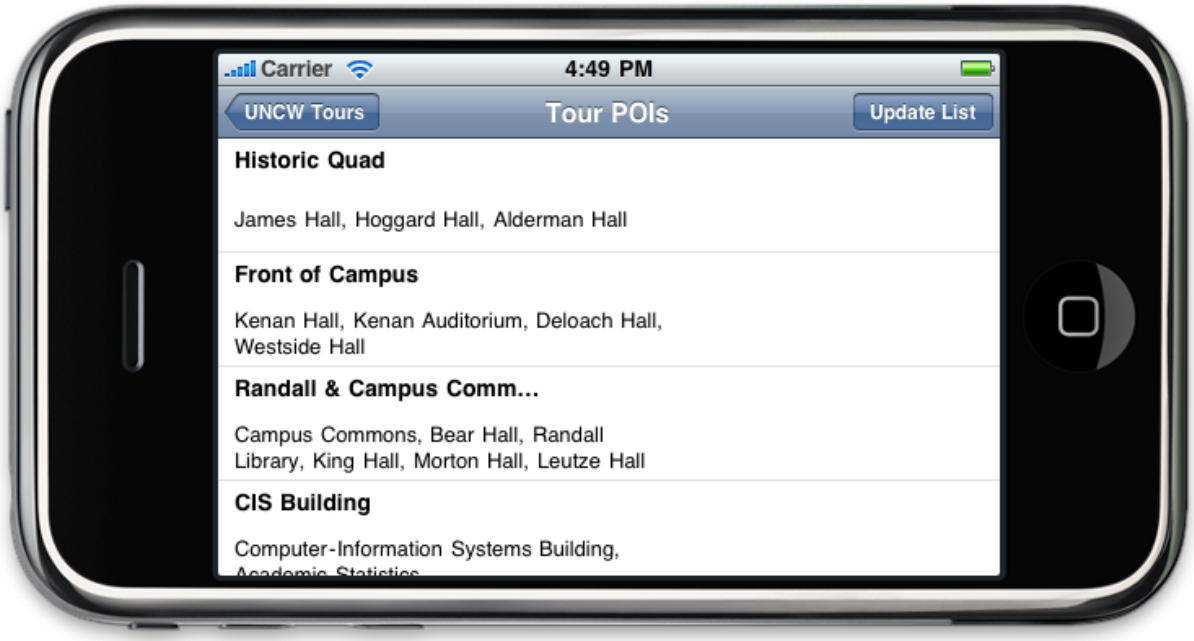
Appendix G – Final Interface Screenshot (iPhone Application)

Welcome View

Available Tours View

Points of Interest View (map)

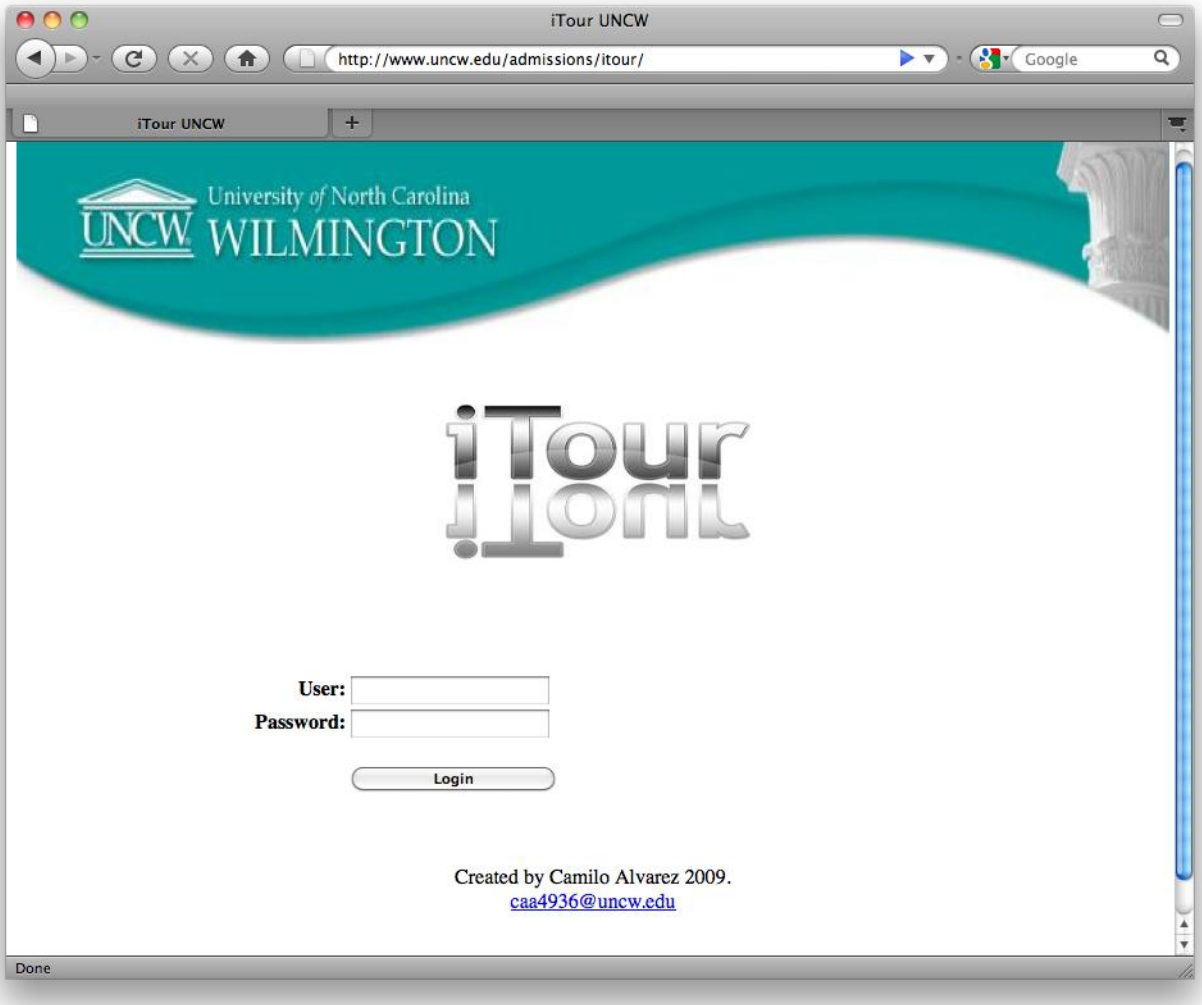
Points of Interest View (table)



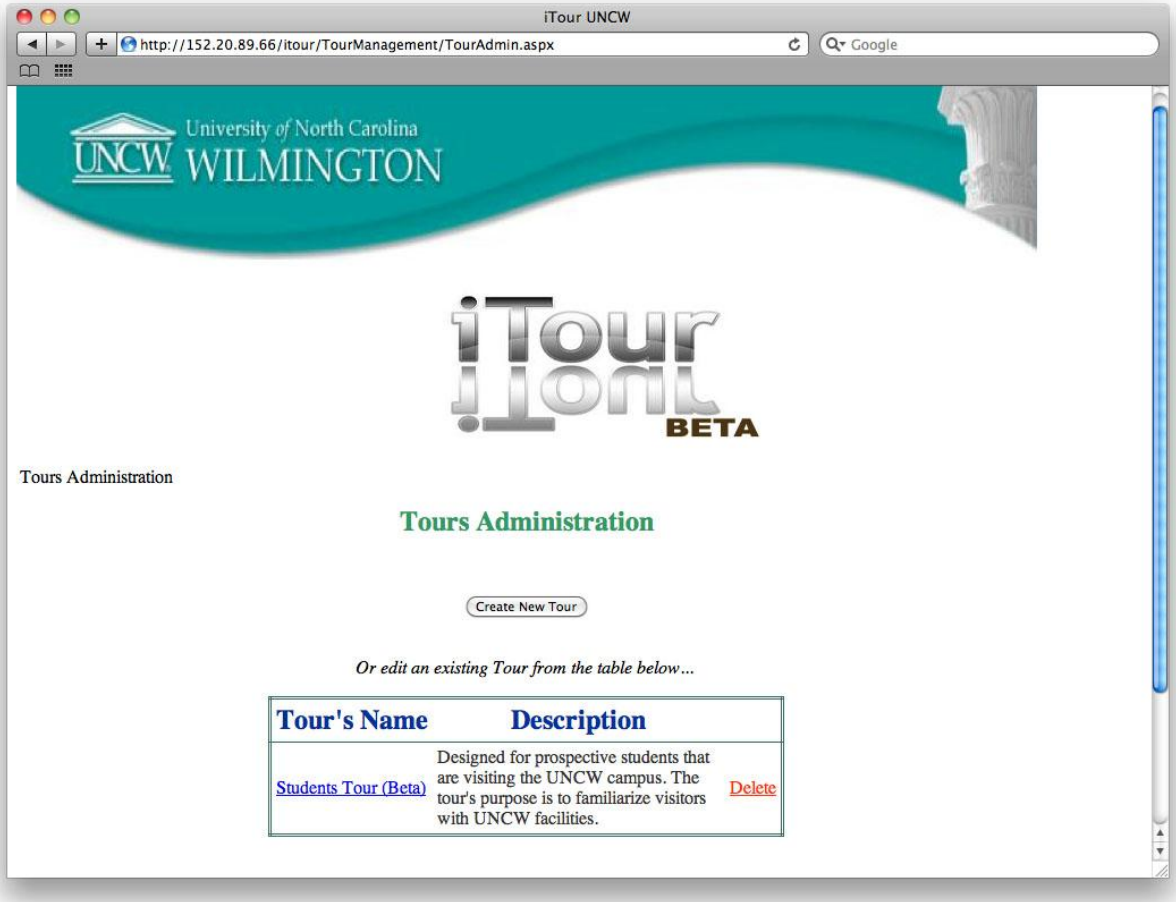
Web Information View

Appendix H – Final Interface Screenshot (Administration Website)

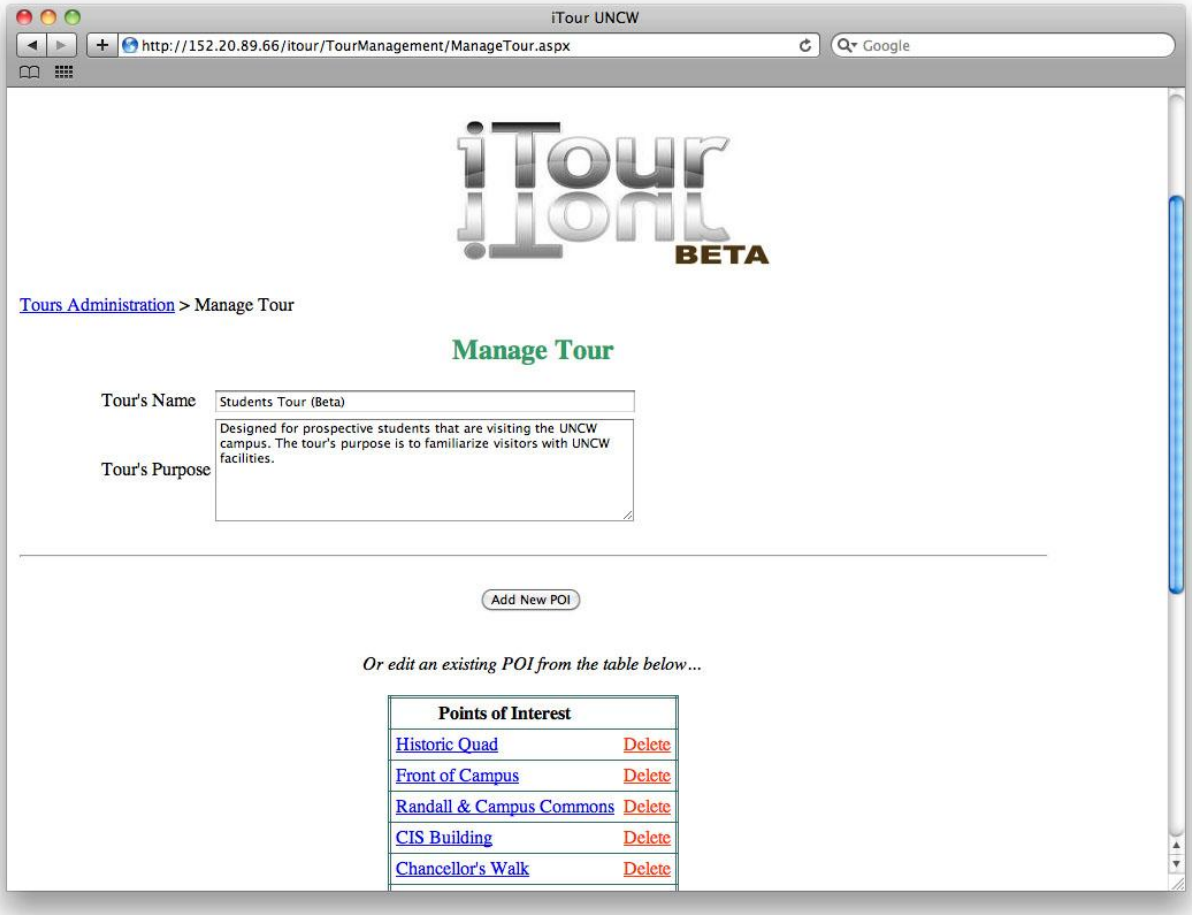
Login



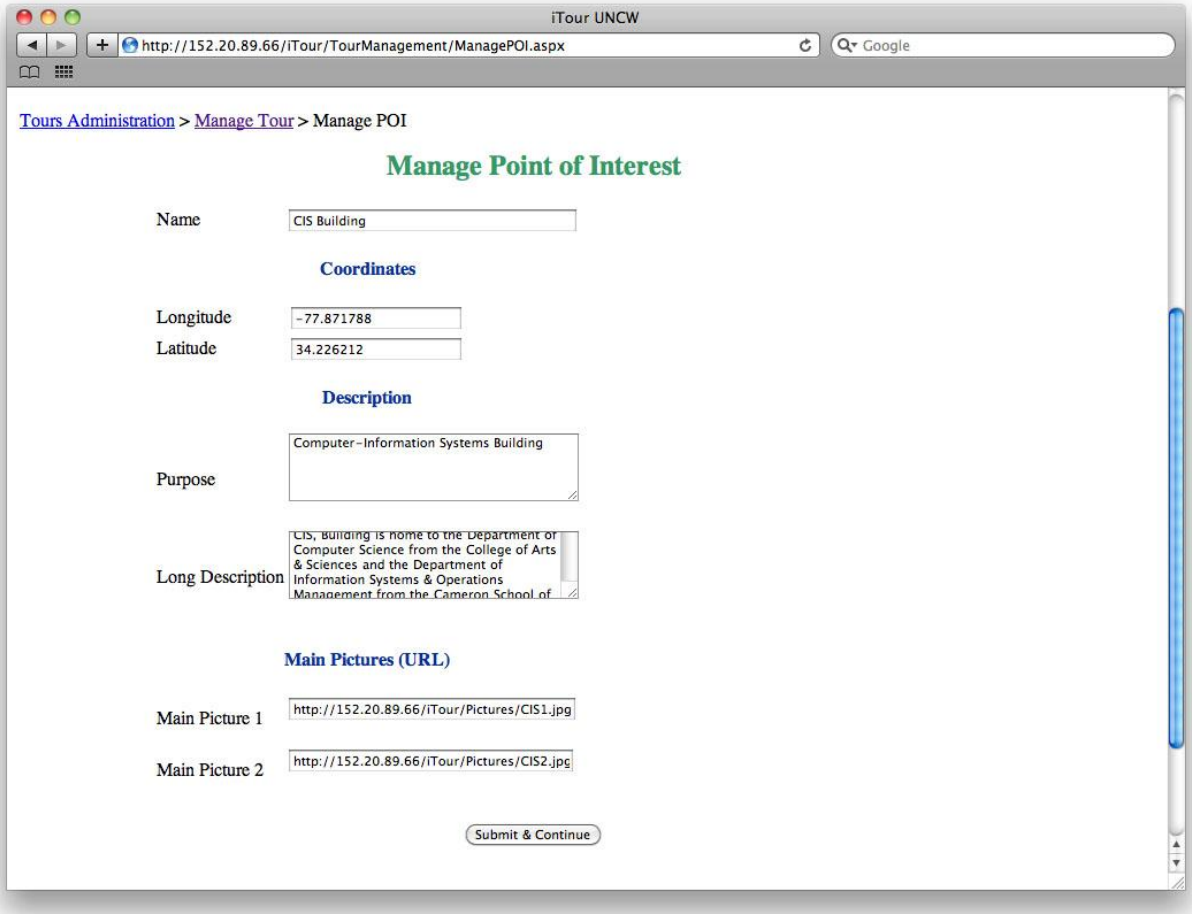
Tours Administration



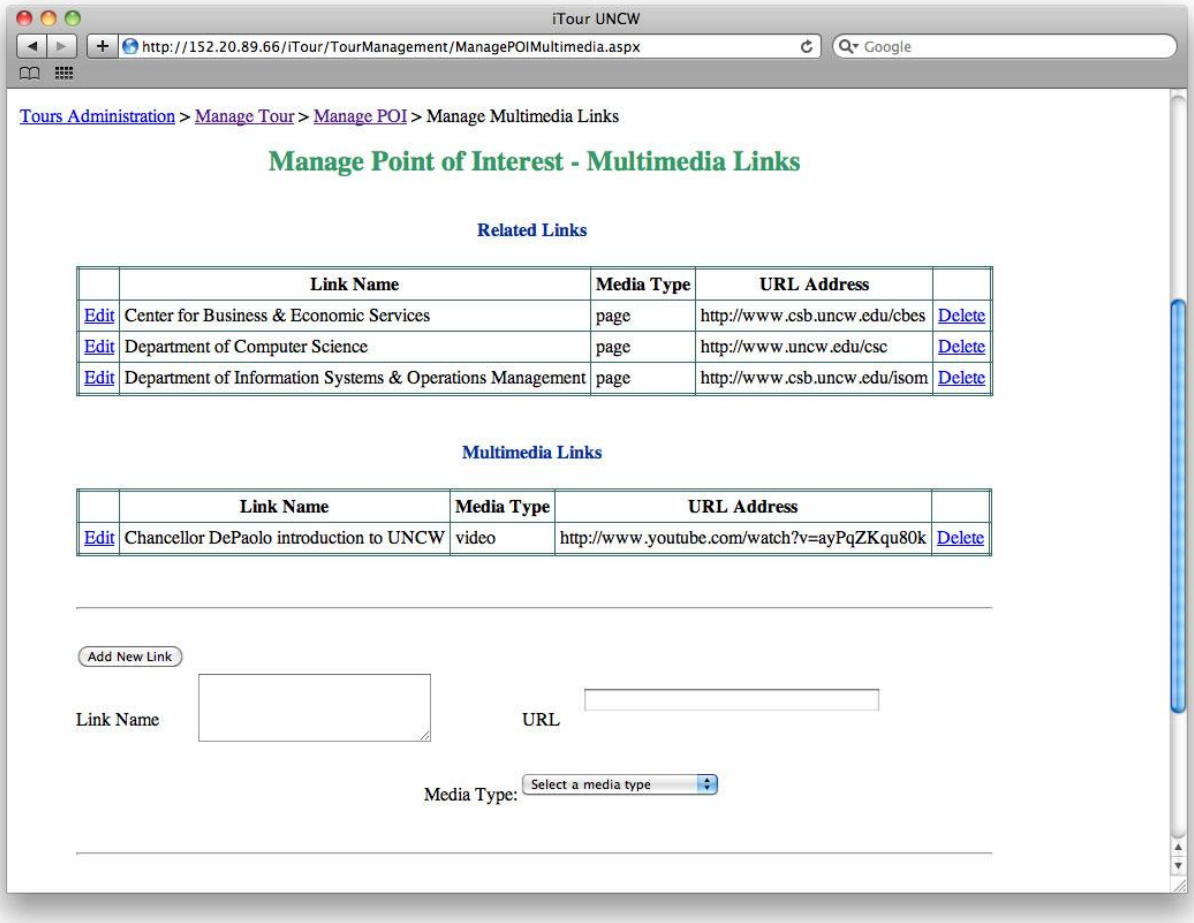
Manage Tour


iTour UNCW
http://152.20.89.66/itour/TourManagement/ManageTour.aspx
Google
iTour
BETA
Tours Administration > Manage Tour
Manage Tour
Tour's Name
Students Tour (Beta)
Tour's Purpose
Designed for prospective students that are visiting the UNCW campus. The tour's purpose is to familiarize visitors with UNCW facilities.
Add New POI
Or edit an existing POI from the table below...
Points of Interest
Historic Quad Delete
Front of Campus Delete
Randall & Campus Commons Delete
CIS Building Delete
Chancellor's Walk Delete

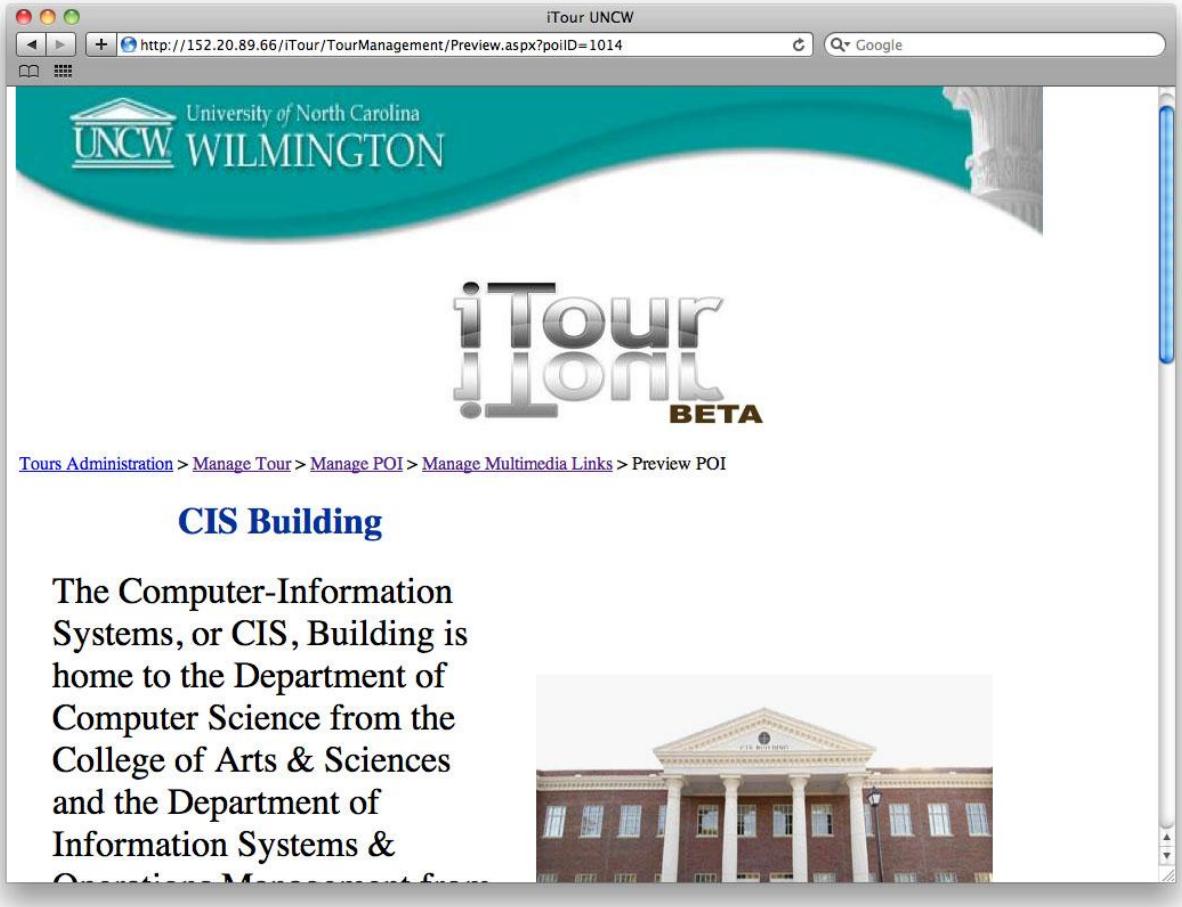
Manage Point of Interest



Manage Multimedia Links



Preview POI


iTour UNCW
http://152.20.89.66/iTour/TourManagement/Preview.aspx?poiID=1014
Google
University of North Carolina
UNCW WILMINGTON
iTour
BETA
Tours Administration > Manage Tour > Manage POI > Manage Multimedia Links > Preview POI
CIS Building
The Computer-Information Systems, or CIS, Building is home to the Department of Computer Science from the College of Arts & Sciences and the Department of Information Systems &

Appendix I – Use Cases

Login – Tour Administrator

Use Case Name	Login	
Scenario	Tour administrator attempts to access the iTour UNCW web portal (system content administration website).	
Brief Description	Depicts the steps necessary for a tour admin to log in to the "iTour" system content administration website.	
Actors	Tour Administrator	
Related Use Cases	None	
Preconditions	Actor has been granted permission to access the system. A username and password exist for the actor**.**	
Post-conditions	If use case is successful, actor is now logged into the system, if not, then system is unchanged. Actor credentials are stored in session.	
Flow of Events	**Actor**	**System**
	2. Actor enters his/her username and password	1. Requests user's username and password 3. System validates the username and password given and allows actor into system 4. Stores actor credentials in session
Exception Conditions	1. Entered username and password are invalid. 2. Database with login information is unavailable. 3. Network is unavailable.	

Edit Tour – Tour Administrator

Use Case Name	Edit Tour	
Scenario	Tour administrator edits an existing tour.	
Brief Description	The tour administrator accesses the iTour UNCW web portal, chooses an existing tour and updates the tour's name and description.	
Actors	Tour Administrator	
Related Use Cases	• Login	
Preconditions	Actor is logged into the system.	
Post-conditions	Tour's information is updated.	
Flow of Events	**Actor**	**System**
	1. On the "iTour Management" web page the actor presses the "modify" button next to the tour to be edited	2. Actor is redirected to the "Manage Tour" webpage. Fields for the tour that have been previously inserted in the database are retrieved and displayed
	3. On the "Manage Tour" webpage the actor updates the fields: a. Tour Name b. Tour's purpose and description	
	4. Actor presses the update button	5. System updates all data in the database for the selected tour and goes back to the "Manage tour webpage"
Exception Conditions	1. Database Server unavailable 2. User exists iTour UNCW web portal unexpectedly 3. User loses internet connection 4. Login session times out	
Alternate Paths	1. Create New T our 2. Delete Tour	

Edit POI – Tour Administrator

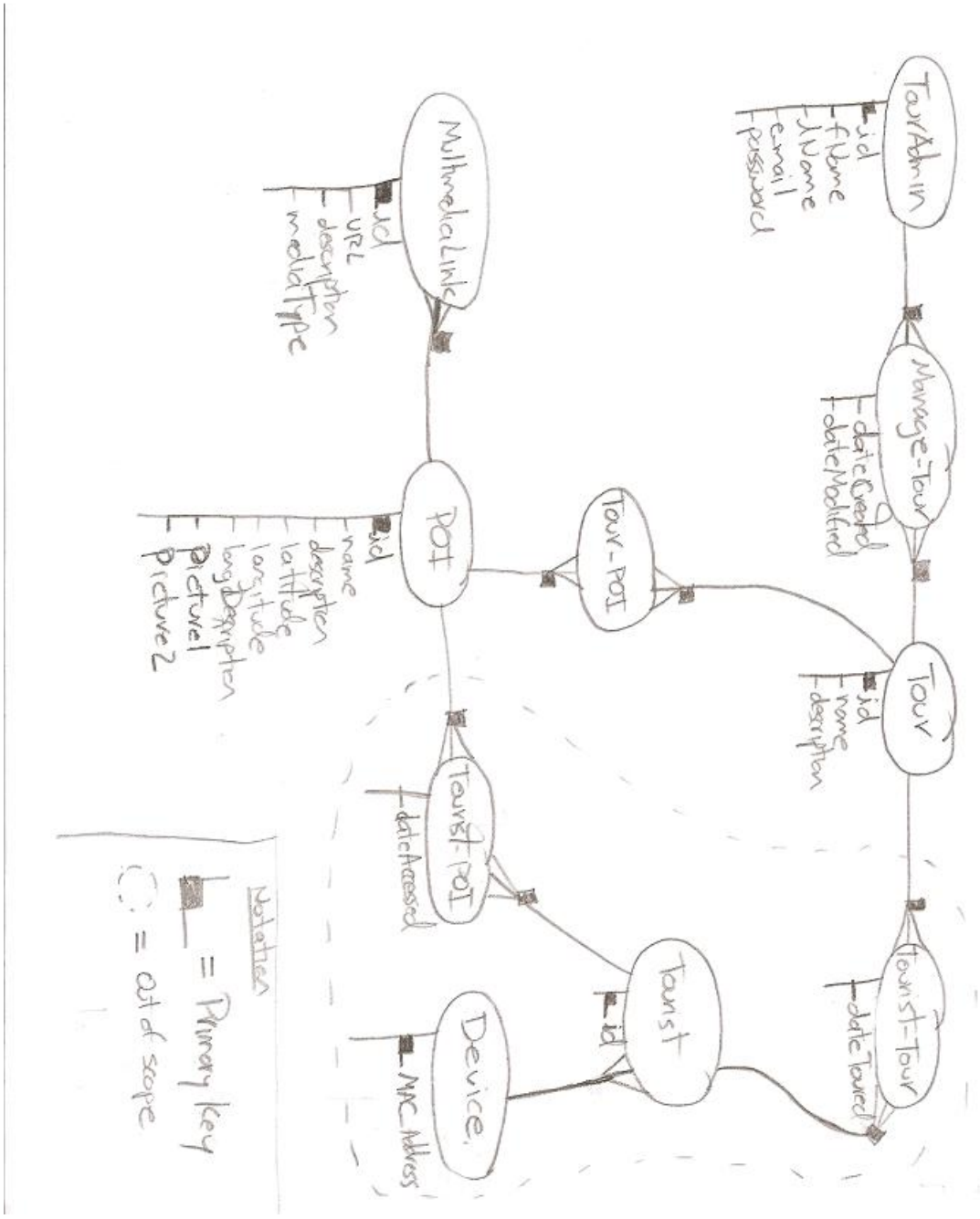
Use Case Name	Edit POI	
Scenario	Tour administrator edits a point of interest (POI) within a tour.	
Brief Description	Based on new information available for a POI, the tour administrator accesses the iTour UNCW web content management system, finds the tour that hosts such POI and updates its information.	
Actors	Tour Administrator	
Related Use Cases	• Login • Edit Tour	
Preconditions	Actor is logged into the system and has selected either a new tour or an existing tour. Actor is at "manage tour" interface.	
Post-conditions	POI's information is updated.	
Flow of Events	**Actor**	**System**
	1) The actor sees a list of existing POIs 2) The actor clicks "modify" next to the point of interest to be edited 4) Actor modifies the fields: Name (POI) POI coordinates (longitude, latitude) POI brief description POI pictures URLs POI's "related links" URLs and descriptions POI's "multimedia" pictures and/or videos URLs and descriptions 5) "Update" button pressed 7) "Done" button pressed	3) Actor is redirected to the "Manage POI" webpage. Fields for the POI that have been previously inserted in the database are retrieved and displayed 6) System updates all data in the database for the edited POI, and displays a preview page of the POI as it will be seen by the tourists 8) System goes back to the "Manage Tour" screen
Exception Conditions	1. Database Server unavailable 2. User exists iTour UNCW web portal unexpectedly 3. User loses internet connection 4. Login session times out	
Alternate Paths	1. Create New POI 2. Delete POI	

Touring – Tourist

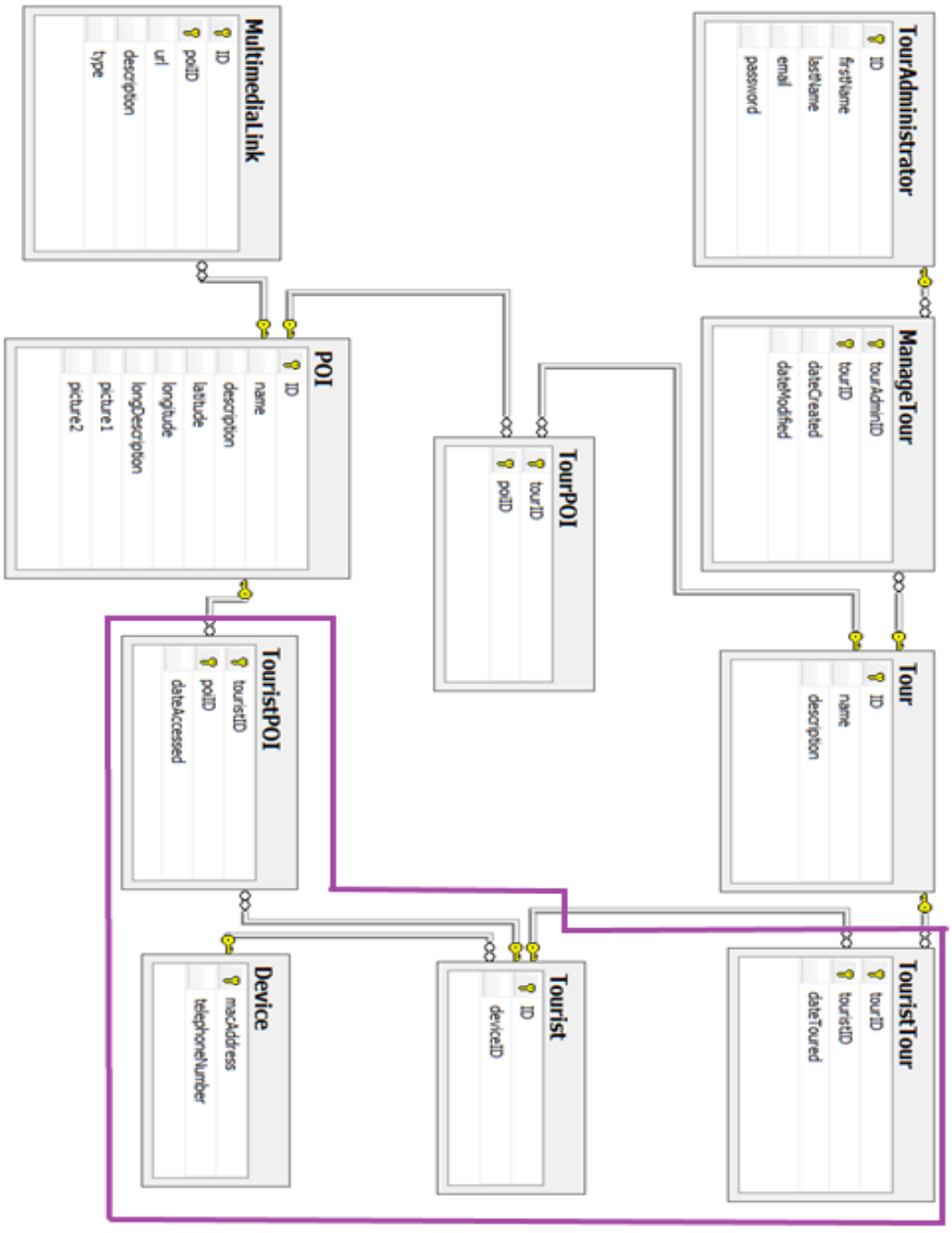
Use Case Name	Touring	
Scenario	Tourist looks for information of a Point of Interest (POI) in the iPhone application.	
Brief Description	The tourist has either obtained an iPhone device with the iTour application pre-installed or has installed the application. The actor starts the application choses an available tour of the university, and in the map he/she taps a landmark that is of interest.	
Actors	Tourist	
Related Use Cases	None	
Preconditions	• iTourUNCW application has been installed on the device • A tour for the respective location exists (e.g UNCW) • A POI for the respective tour exists (e.g CIS building) • Actor is sees a table of available tours.	
Post-conditions	None	
Flow of Events	**Actor**	**System**
	1. Selects a tour from the available tours table 3. Taps a POI in the map 5. Taps more info button	2. Launches the POI view featuring a map of the tour's points of interest (POIs) and the user's location 4. Displays a description message giving the name of the POI tapped and displays a disclosure button for more information 6. Launches embedded web browser and display "poi web view" featuring additional info about the chosen POI. Browser has a back, and "BACK TO TOUR MAP" buttons for application navigation
Exception Conditions	1. Database Server unavailable 2. Web Server unavailable 3. User loses internet connectivity 4. iPhone device battery is drained 5. Actor presses the "home" button	
Alternate Paths	1. User receives and takes a phone call 2. User receives and reads a text message SMS	

Appendix J – ER Diagram

Sketch

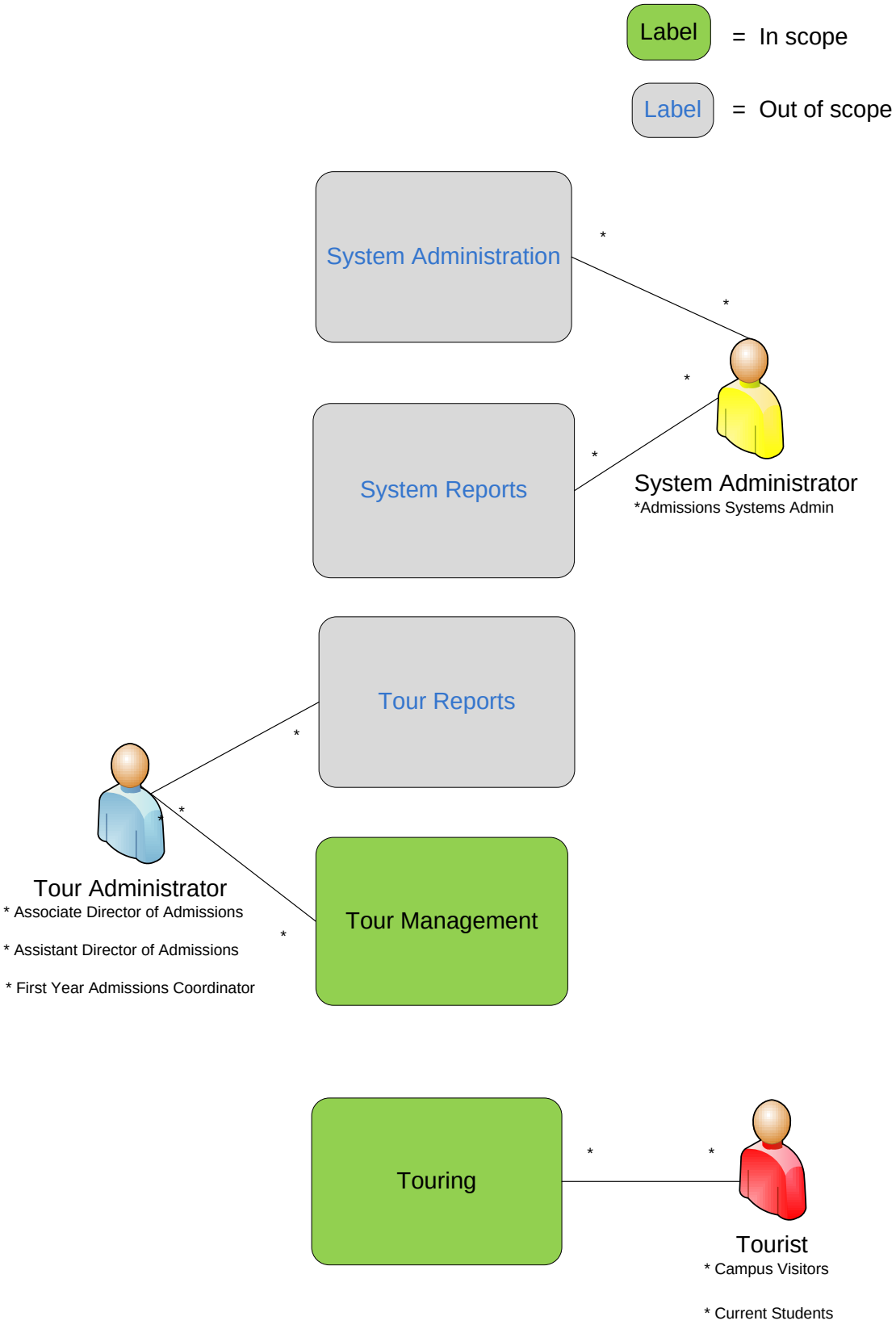


Final Database Diagram



= Encloses tables not used by the current system

Appendix K – Actor Diagram



Appendix L – User Evaluation Survey

UNIVERSITY OF NORTH CAROLINA WILMINGTON

Evaluation of UNCW iTour Application

UNCW iTour Application User Survey

This survey will be used by MS CSIS graduate student Camilo Alvarez to evaluate the design and use of the UNCW iTour Application.

Please be honest in your assessment of this iPhone application as survey results will be used for further refinements of the software.

Thank you in advance for completing the survey!

1. Have you ever used an iPhone before? *
   - Yes
   - No

2. To what group do you belong?*
   - Admissions Office Personnel
   - Current UNCW Student
   - Current UNCW Faculty/Staff
   - Other, please specify

3. Have you ever taken a university virtual tour through a website?*
   - Yes
   - No

4. Please rate the following features of the UNCW iTour, with 5 being the highest rating:*

	1	2	3	4	5
Attractive Appearance					
Ease of Use					
Intutitive					
Readability					
Responsiveness					
Completeness of tour					

5. Please rate the UNCW iTour in comparison to other virtual tours.*

	Worse	Same	Better
Ease of use			
Information availability			
Completeness of tour			

6. If you were a prospective student how would the UNCW iTour Application meet your expectations?*
   - UNCW iTour has greatly exceeded my expectations

- UNCW iTour has slightly exceeded my expectations
- UNCW iTour has met my expectations
- UNCW iTour has fallen short of meeting my expectations
- UNCW iTour has completely failed to meet my expectations
- Other, please specify

7. If UNCW iTour failed to meet your expectations, could you please tell us what the problems are?

8. What additional features would you like to see included in the UNCW iTour application?

9. I would recommend UNCW iTour to others.*
   - Strongly Agree
   - Agree
   - Neutral
   - Disagree
   - Strongly Disagree

10. Please enter your email address if you would like to be entered to win a FREE $25 Starbucks card. We will contact you by email if you win.

Done Cancel

Appendix M – User Evaluation Survey Results

1. Have you ever used an iPhone before?

	Response Total	Response Percent
Yes	12	92%
No	1	8%
Total Respondents		13

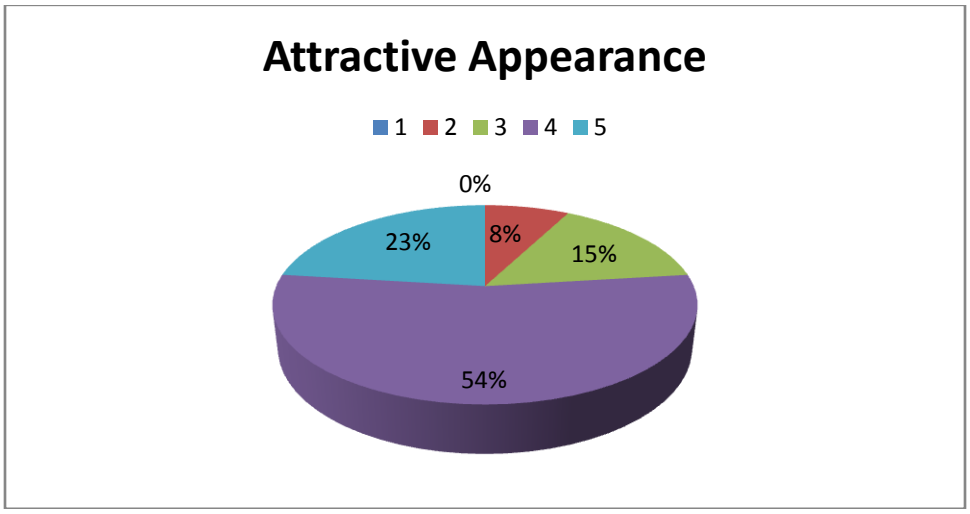
2. To what group do you belong?

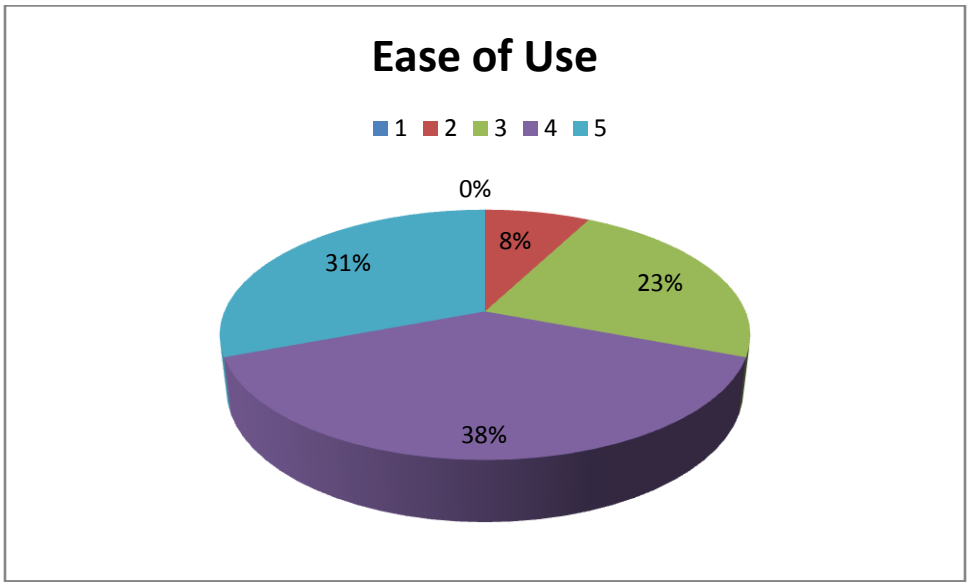
	Response Total	Response Percent
Admissions Office Personnel	2	15%
Current UNCW Student	8	62%
Current UNCW Faculty/Staff	3	23%
Other, please specify	0	0%
Total Respondents		13

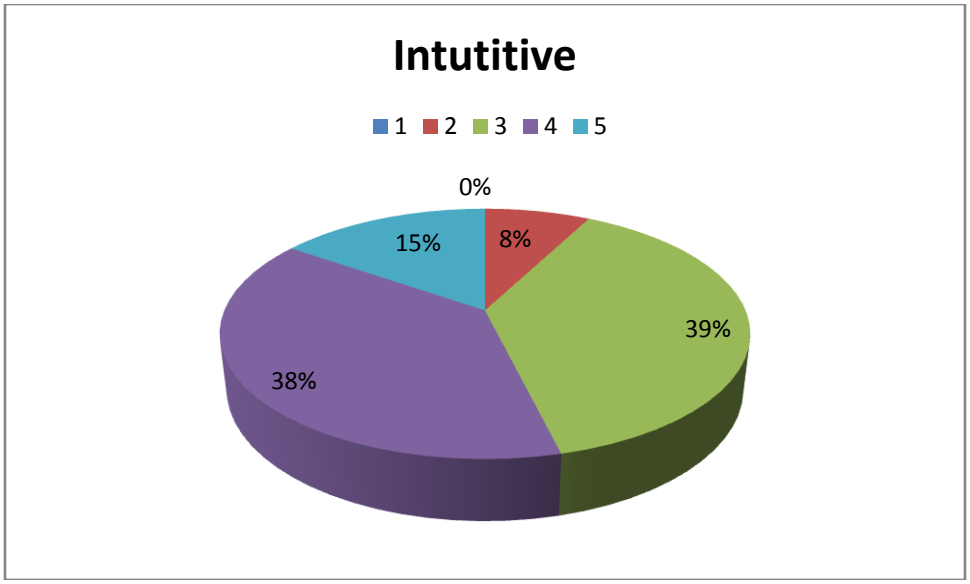
3. Have you ever taken a university virtual tour through a website?

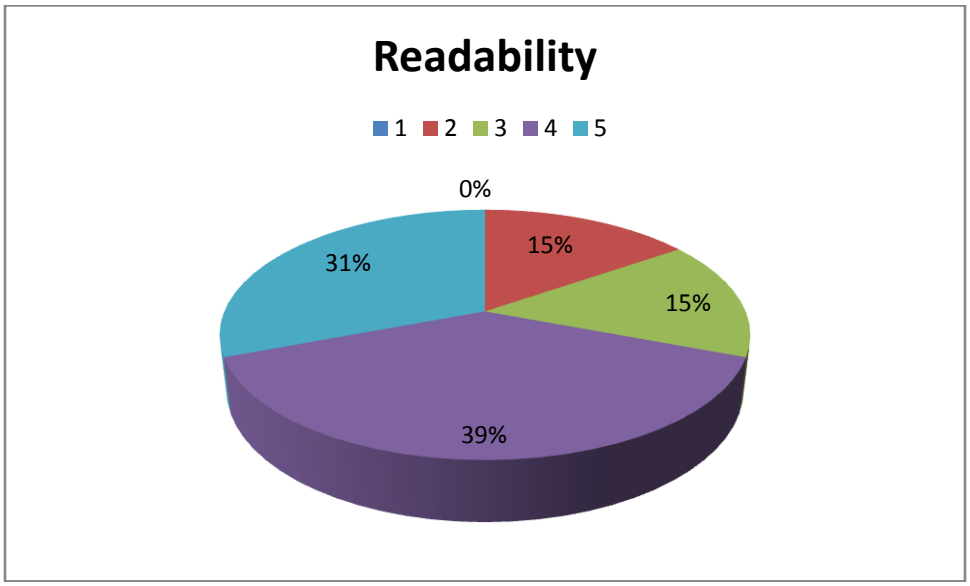
	Response Total	Response Percent
Yes	9	69%
No	4	31%
Total Respondents		13

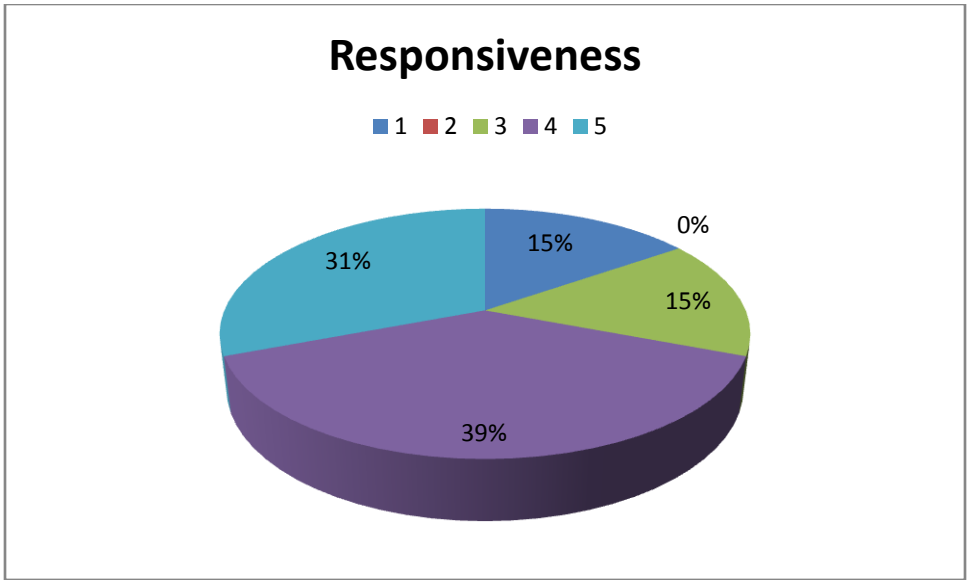
4. Please rate the following features of the UNCW iTour, with 5 being the highest rating:

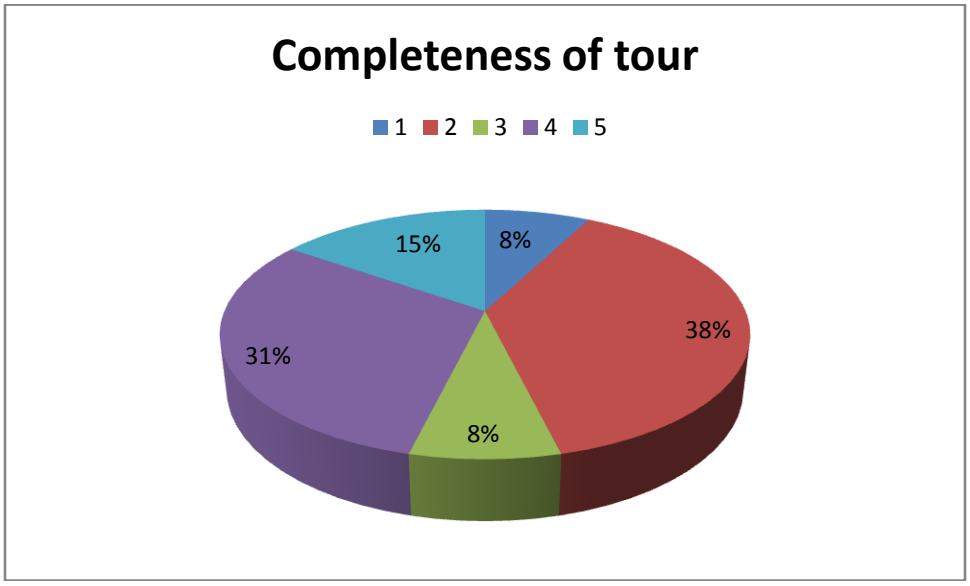



Ease of Use
1
2
3
4
5
0%
8%
23%
38%
31%

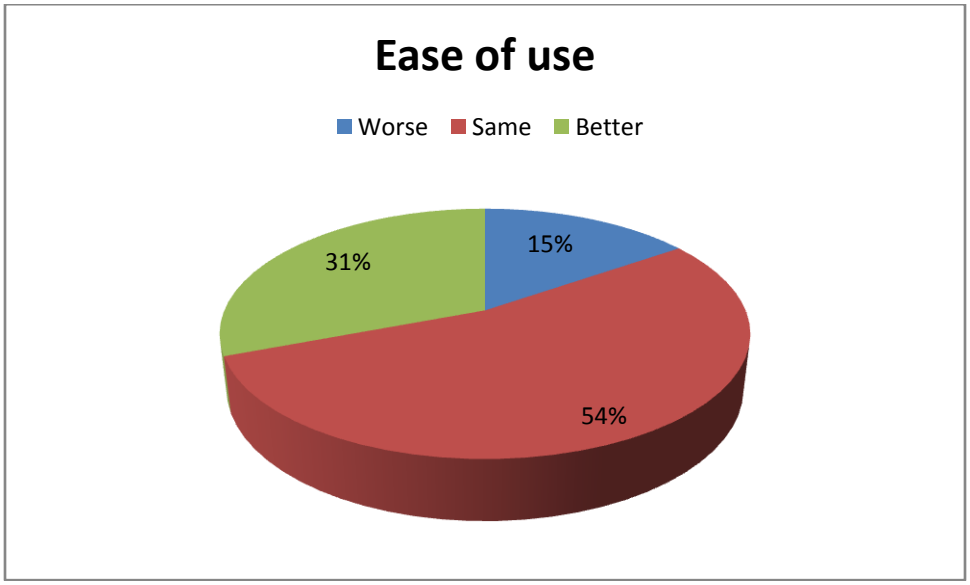

Intutitive
1
2
3
4
5
0%
8%
39%
38%
15%

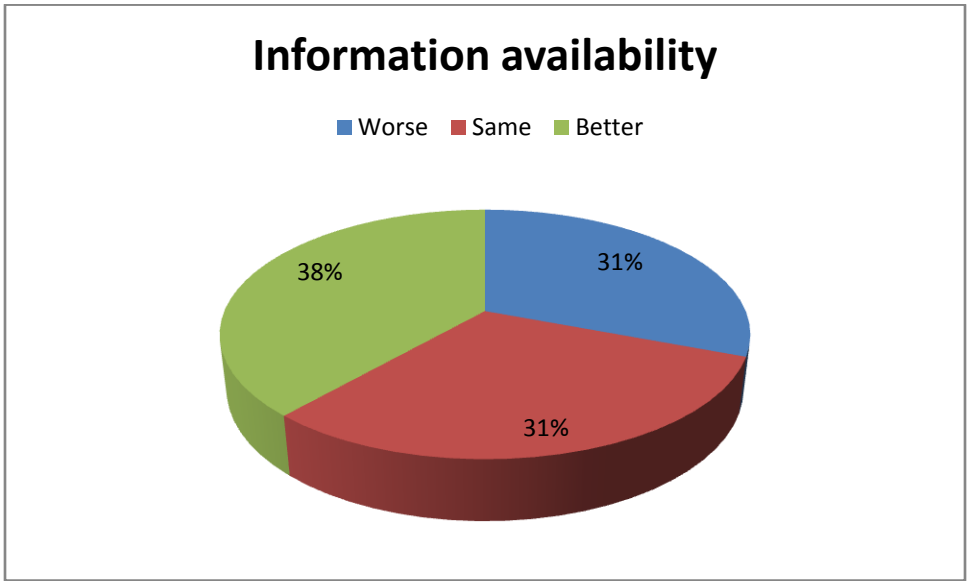

Readability
1
2
3
4
5
0%
15%
15%
39%
31%

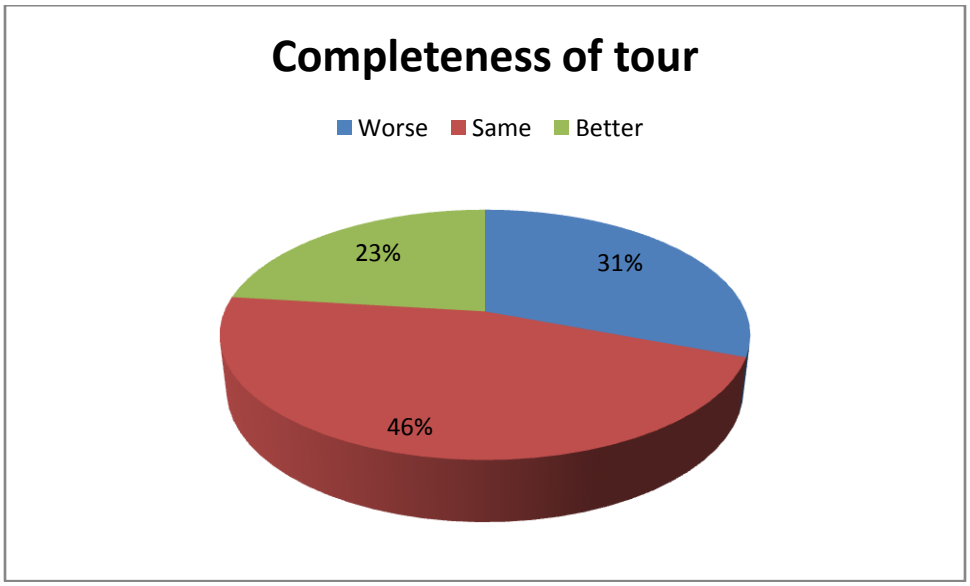

Responsiveness
1
2
3
4
5
15%
0%
15%
39%
31%



5. Please rate the UNCW iTour in comparison to other virtual tours.




Information availability
Worse
Same
Better
31%
31%
38%


Completeness of tour
Worse
Same
Better
31%
46%
23%

6. If you were a prospective student how would the UNCW iTour Application meet your expectations?

	Response Total	Response Percent
UNCW iTour has greatly exceeded my expectations	1	8%
UNCW iTour has slightly exceeded my expectations	2	15%
UNCW iTour has met my expectations	4	31%
UNCW iTour has fallen short of meeting my expectations	4	31%
UNCW iTour has completely failed to meet my expectations	0	0%
Other, please specify view	2	15%
Total Respondents		13

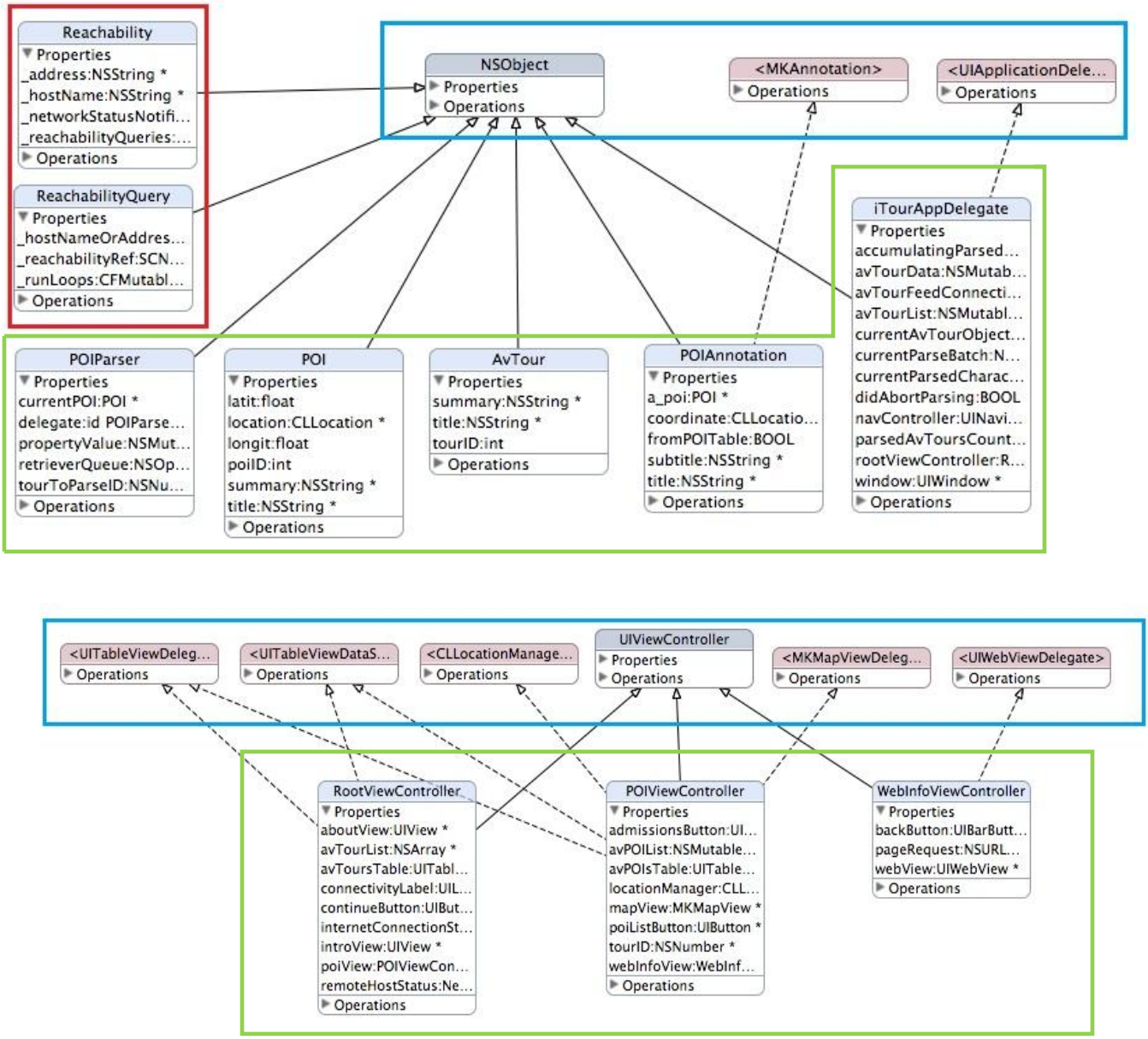
7.	If UNCW iTour failed to meet your expectations, could you please tell us what the problems are?	Full Response
1.	The application is just not up to par with other competing schools and applications.	view
2.	It was not very ressponsive and was slow to load. Even when i was in full bars of service.	view
3.	It was very slow. Trouble loading the tours wasted a lot of time.	view
4.	Need to have more complete information about the points of interest. Points of interest need to be defined as buildings, not regions of campus. That would make it easier to know where you are. Visitors who do not know where they are do know know about the general areas of campus (like Chancellors Walk).	view

8.	What additional features would you like to see included in the UNCW iTour application?	Full Response
1.	Once I entered the tour and went to the list view, I had difficulty getting back to the original view of my GPS signal (blue) back to the original state.	view
2.	Side views of each campus building. Also a notification of buildings near you. Or as you walk through campus the mac "voice" tells you of buildings near you.	view
3.	Photos, Class Tools (Blackboard, CourseCompass, etc), visually more appealing	view
4.	A more completed tour and possibly some video feeds about each area(POI)	view
5.	A Google Street View of campus	view
6.	Nothing it is a good application but I do not prefer the virtual tours.	view
7.	More content for each point of interest.	view

9. I would recommend UNCW iTour to others.

	Response Total	Response Percent
Strongly Agree	2	15%
Agree	6	46%
Neutral	2	15%
Disagree	1	8%
Strongly Disagree	2	15%
Total Respondents		13

Appendix N – Class Diagram



LEGEND

= Classes

= Inheritance

= Apple provided classes

= Categories

= Protocol implementation

= Framework classes

= Protocols

= Implemented classes

Appendix O – Source Code

--- Contact the author ---