

Michael Tilahun Damena

Windows Phone Application Development for the Contextual Activity Sampling System

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

10 February 2013

Author(s) Title	Michael Tilahun Damena Windows Phone Application Development for the Contextual Activity Sampling System
Number of Pages Date	66 pages + 6 appendices 10 February 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Kari Salo, Lecturer Petri Vesiviki, Lecturer
<p>The project was carried out to develop a Windows Phone application for CASS-Query tool. The Contextual Activity Sampling System is a research methodology for the contextual tracking of activities. The main purpose of the project was to make the query tool available on Windows platform as requested by University of Helsinki. The goal of the project was to develop the application in a way that users could smoothly interact with it giving them full functionality, making the whole query tool more reliable.</p> <p>To achieve the purpose and goals, the Model View Controller software architecture was followed. A couple of iterations were carried out to settle on the UI design that it finally attained. Following specific UI design rules and software architectural methods, a Windows Phone native application was developed and made available on the Marketplace of Microsoft.</p> <p>The query tool used to have a client application that was developed using J2ME and was available on limited mobile devices. Now a day, a couple of platforms have come into play and the already existing application was getting out dated. Making a Windows version of a CASS-Q client has increased the platforms in which it is available, making it more reachable by number researchers. This makes the whole query tool more powerful and out useful to many.</p> <p>The result of the thesis final year was a complete Windows Phone application that could be used as the CASS-Q client application to assist research data collection.</p>	
Keywords	Windows Phone XML Survey Tool CASS-Q

Contents

Abbreviations and Terms	1
1 Introduction	2
2 Contextual Activity Sampling System	3
2.1 Introduction to CASS	3
2.2 System Architecture of CASS-Q	5
2.3 User Groups of CASS-Q system	7
3 Windows Phone Application Platform	8
3.1 Windows Phone Application Platform Architecture	9
3.2 XNA Framework	10
3.3 Silverlight Framework	10
3.3.1 Major Components of Silverlight Framework	10
3.3.2 Programming Features of Silverlight	11
3.4 Windows phone application development	12
3.4.1 General Design Principles	13
3.4.2 Application Structure and Navigation Models for Windows Phone	16
3.4.3 Essential Graphics, Visual Indicators and Notifications	19
3.5 Requesting for, Parsing and Building XML in WP	22
3.5.1 XML Web Request	23
3.5.2 Parsing XML	24
3.5.3 Building XML	26
4 Requirement analysis	27
4.1 Stockholders	27
4.2 Goals	27
4.3 Functional Requirements	28
4.3.1 Context Diagram	28
4.3.2 Input, Operation and Output Requirements	29
4.3.3 Dataflow Diagram	30
4.3.4 Use case	33
4.3.5 Sequence Diagrams	37
4.4 Non-Functional Requirements	39
5 Functional Specifications	40

5.1	Target achievements	40
5.1.1	Mandatory criteria	40
5.1.2	Desired criteria	41
5.2	External Interface Requirements	41
5.2.1	Hardware requirements	41
5.2.2	Software requirements	42
5.3	Product Functions	42
5.4	Product Data	43
5.5	Development environment and Global test scenarios	43
6	Graphical Design and Software Architecture Considerations of CASS-Q Client	45
6.1	Graphical user interface	45
6.1.1	Extensible Application Markup Language	45
6.1.2	Guidelines Followed in the Designing of UI of CASS-Q	46
6.2	Software architecture	47
6.2.1	Model-View-Controller Design Pattern	48
6.2.2	MVC Concept of CASS-Q Client	48
7	Implementation of the CASS-Q Client Graphical Design	51
7.1	Application Banner	51
7.2	Launcher and Application Bar Icons	52
7.3	Input Controls	52
7.4	Layouts	54
8	Implementation of the CASS-Q Business Logic	57
8.1	Making XML Web Request	57
8.2	Parsing XML	58
8.3	Building UI	59
8.4	Accepting User Answer	60
8.5	Building XML and Uploading User Answer	61
8.6	Settings	62
9	Testing and Publishing	63
9.1	Usability testing	63
9.2	Submitting CASS-Q to Windows Phone Marketplace	64
10	Conclusion	65

Appendices

Appendix 1. Screen shots of CASS-Q client

Abbreviations and Terms

App	Application
CASS-Q	Contextual Activity Sampling System-Query
CLR	Common Language Runtime
DFD	Data Flow Diagram
DLR	Dynamic Language Runtime
GUI	Graphical User Interface
MVC	Model View Controller
MVP	Model View Presenter
MVVM	Model View View Model
OS	Operating System
UI	User Interface
URL	Uniform Resource Locator
WCF	Windows Communication Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language
XNA	It is the next generation of DirectX

1 Introduction

In the recent years, it has become prominent that mobile applications appear to be one major mechanism to get work done. This has made work to be done in a more effective and timely way without geographical restrictions. Some of the applications are totally independent doing all the work on the mobile apparatus itself. However, some other applications interact with server side programs to fully accomplish the purpose they have been developed for.

A project that aimed at developing a tool that would help researchers in collecting social data was launched under the 6th EU Framework Programme Research and Development. This tool was named Contextual Activity Sampling System (CASS). The CASS-Q tool has been developed in the KP-Lab project (<http://kp-lab.org>). It allows registering users, adding queries and retrieving collected data in Excel format.

The CASS-Q methods and tools provide contextualized data that allow the analysing and modelling of data within person changes across time. For that to be accomplished, a user has to first submit answers to the questions retrieved from the server by a client application. Therefore, the Cass-Q is a tool for conducting questionnaire-based researches using the Internet as the media for communication between different devices. The main goal of this thesis is to develop a client mobile application for the CASS-Q System that runs on a Windows phone. This would make the application available to a wider range of users making the research data collection more effective. Using the client application, Windows phone users would be able to give answers to the survey questions retrieved from the CASS-Q System and send back the answers to the CASS-Q System with a time stamp.

Researchers and clients can have different kinds of mobile device and Windows mobile phones are gaining more popularity these days. Developing a Windows version of the client application would make the research tool more reachable, powerful and useable by different users.

2 Contextual Activity Sampling System

The Contextual Activity Sampling System Query tool (CASS-Q) is technically a web application that can be complete and functional with a mobile client application. This chapter describes the CASS-Q tool in detail and the concept that it is based on. In addition, the architecture is described briefly.

2.1 Introduction to CASS

A project aimed at developing a tool that helps researchers in collecting social data for contextual tracking was launched under the 6th EU Framework Programme Research and Development. This tool was named Contextual Activity Sampling System (CASS). The CASS-Q tool has been developed in the KP-Lab project (<http://kp-lab.org>). It allows registering users, adding queries, allowing respondents give answers and retrieving collected data in Excel format. The concept behind the tool was the frequent sampling of participants' learning and working practices over a defined period of time. The system makes it possible to collect a big amount of data in an organized manner within a short period of time.

CASS-Q focuses on collecting data about participants' feelings and thoughts in certain situations and activities at a certain time. This is the main advantage of using CASS-Q tool over the commonly used method of collecting user data which focuses on gathering information that mostly is participants' beliefs and opinions. To attain the aforementioned advantage, CASS-Q implements three types of data collection methods.

If a query is set with *fixed time* data collection method then it is sent to respondents at specific times of a day. A different query can be sent to each of the times set but the same query can be sent at different times in a day to collect feelings and thoughts at different moments of a day. The other collection method is called *fixed interval* data collection method. In this method a specific start time and an interval is defined. At the start time, the first query is sent to the respondents then at the defined interval, other queries are sent. For example, if the specific start time is 8:00am and the interval is set to be 40 minutes, then the first query is sent to the respondent at 8:00am then the consecutive queries are sent to respondents in a 40 minute interval like 8:40am, 9:20am and the like. The last data collection method is the *event contingent* data collection method. If this method is used, the server sends only one set of query but the respondents can answer it any time and as of as they wish to.

Based on the three methods of data collection, a researcher would be able to collect data about the participants' feelings and thoughts and with that a useful statistics would be attained to tell how a participant feels challenged and competent about certain activity during a day. The correlation between being feeling challenged and competent suggests how a person's work-flow is affected by certain activities during a day.

The CASS-Q application that resides on the server has a built-in tool that lets a researcher categorize respondents' feelings into a four-channel model. *Figure 1* below shows the graph that is generated after some statistics is visualized on the server side.

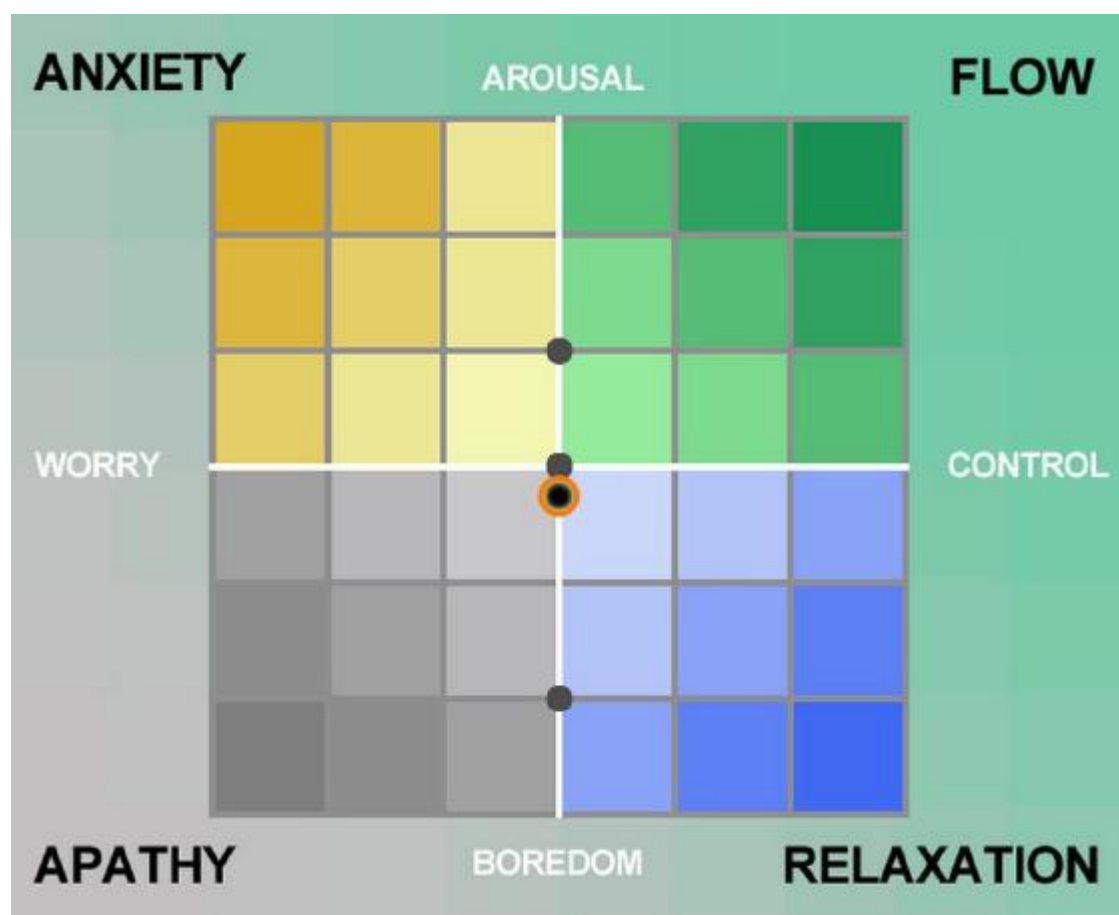


Figure 1. Statistics visualization (four-channel model). Reprinted from Research on knowledge practices with the Contextual Activity Sampling System (2009) [1]

As shown in *figure 1*, the four-channel model for statistics visualization has names to verbally tell what the statistics signifies. If a participant values his/her competence and challenge during an activity as very high, then an optimal flow would be used to tell the

statistics. If a participant feels gives low value to his/her competence and challenge during an activity, then he/she will be having the opposite state to the flow and it is named apathy. In case of high competence and low challenge the participant experiences relaxation state. If a participant feels less competent but the challenge remains high, then the state would be anxiety.

Researchers can use the four-channel visualization tool to easily identify how participants feel in some activity. This knowledge in return would let the researchers trace the development in learning and working environments. [1]

2.2 System architecture of CASS-Q

The CASS-Q system has two parts that should interact with each other to make the tool fully functional. It includes a web application that resides on a server and a mobile client application that should be installed on participants' mobile devices. As shown in *figure 2*, the research administrator is responsible for creating the survey and assigning the respondent.

A part of the application that resides on the server is responsible for giving necessary response to the client application that may make a request. All research information and data related to users of the system is stored on a database. A research administrator is responsible for creating researches, adding queries and setting the data collection methodology. The data that is retrieved from client applications are stored in the database and necessary statistical interpretation of the data is provided to the researcher by the web application.

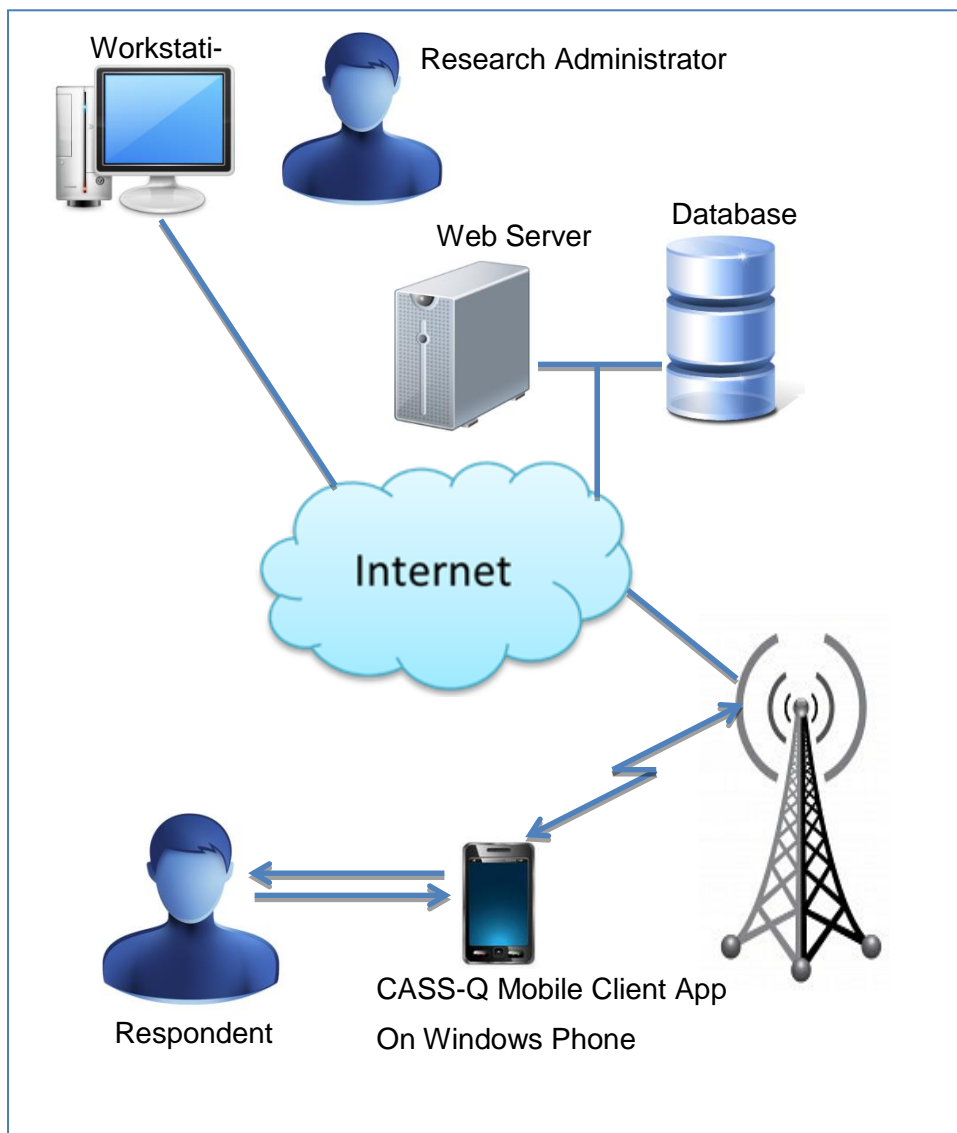


Figure 2. CASS-Q System architecture.

As shown in *figure 2* above, the Internet is the media for the web application and the mobile client application to communicate. Therefore the CASS-Q to function, both the mobile device and the server should have an Internet connection.

2.3 User groups of CASS-Q system

The whole system has four groups of users which perform different tasks using the CASS-Q web based as well as a mobile client application. The users of the CASS-Q System are briefly described in the following paragraphs.

Super administrator is the user who can register other users, manage user accounts and configure and assign roles to users. Roles can be used to give or revoke certain functionalities that the CASS-Q system can provide to users. The roles that a system administrator can grant to other users are the following:

- Super administrator
- Research administrator
- Researcher
- Respondent

Other than managing the roles that a certain user should have, the user with the *Super Administrator* role can delete other users.

Research administrator is a user who administers a research in the system. He/she is allowed to create researches, queries and questions. Added to that, the research administrator can append respondents to an already created research. During or after the research, he/she can collect the research data as well. As a *researcher*, one is only allowed to collect data from the system. The last group of users in the CASS-Q System is the *respondent* which is a collection of people who answer query using the Cass-Q mobile client. Respondents have to be assigned to the research before being able to access any of the research questions from the server side application of CASS-Q.

3 Windows Phone Application Platform

By using existing Microsoft tools and technologies, Windows phone Application Platform provides all the necessary methods to enable developers in creating appealing user experiences running on a Windows phone. The existing Microsoft tools and technologies that are used in the Windows Phone Application Platform are Visual Studio, Expression Blend, Silverlight and XNA Frameworks. [2]

The platform is developed so that it can support different screen and device types giving familiar user experiences though the applications are developed for different types of devices with different sizes and types of screen. *Figure 3* shows how the application development platform is organized.



Figure 3. Common Windows Phone Application Platform. Reprinted from Microsoft (2012) [2]

As shown in *figure 3*, the cloud is the main source of common information for different users even if different devices may be used. Maintaining the information on the cloud makes it easier and convenient for the users to migrate from device to another device as the need arises. However, the user experience they obtain being n different types of screens and devices is equally important as the data they obtain from the cloud from different devices. To make this happen, a common development platform is crucial. This is the reason why the Windows Application Development Platform is the common ground to develop applications for different types of devices. [2]

3.1 Windows Phone Application Platform Architecture

The architecture is made up of four components to be complete. The four components can be grouped into two sub categories; one that includes the runtimes and the tools used to build secure graphically rich applications and display them on the user device's screen. The second category is the cloud category, which includes the cloud services and the portal services. These concepts are better depicted in *figure 4* below.

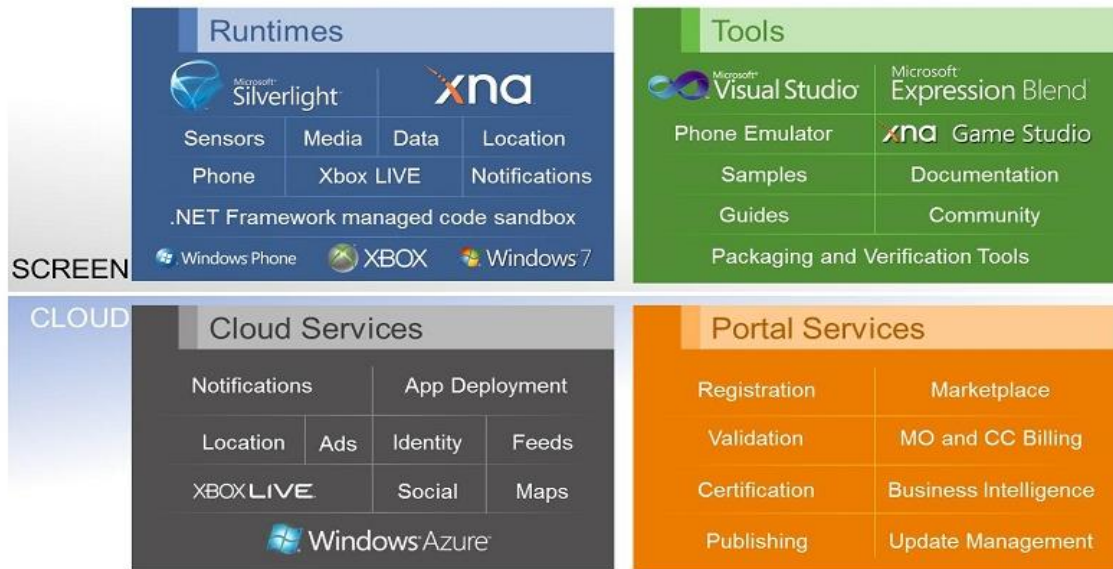


Figure 4. Windows Phone Application Platform Architecture. Reprinted from Microsoft (2012) [2]

As shown in *figure 4*, XNA and Silverlight Frameworks provide the environment to build an application that runs on the client device. This makes the runtime part of the architecture. The tools used to design, develop, debug and quickly update applications make up the tool part of the architecture. The codes written on the server to give service to client phone applications with better mobility services are part of the cloud services. Finally, the tools we use to upload, certify and sell the applications we develop make up the portal service part of the whole architecture. [2]

3.2 XNA Framework

XNA Framework is one of the application development platforms provided by Microsoft through which games can be developed. To do so, the XNA Game Studio is needed. While developers have been developing games for years without the XNA Framework, its existence has made the making of games attainable by the majority of game developers as well as programmers.

Added to the free and less complicated game development environment, the framework has provided a mechanism for games developed using XNA Game Studio Express to be playable on Microsoft's Xbox 360.

3.3 Silverlight Framework

As mentioned in section 3.1, the Windows phone Application Platform provides two frameworks for developing applications; these are Silverlight and XNA Frameworks. As the project work targeted at developing a mobile client application that fetches data from a server side application, the features provided by the Silverlight Framework were sufficient. The project needed an event driven XAML-based application development with an easy to use user interface design and the right kind of windows phone framework for such task was found to be Silverlight Framework. Below are the capabilities that the Silverlight Framework provides: [3]

- A XAML based, event driven application framework
- Rapid creation of a Rich Internet Application-style user interface
- Windows phone controls
- Embed video inside your application
- An HTML web browser control.

3.3.1 Major Components of Silverlight Framework

The Core Presentation Framework and the dot Net Framework for Silverlight are the two major components that make up the whole Silverlight Framework. In addition to these, the Silverlight Framework consists of two other parts, which are the installers and update component. [3;4]

The Core Presentation Framework provides services that enable UI design and interaction. It is this component that enables data binding in XAML, media play back, digital right management and presentation features. Other than the data binding facility, it is under the Core Presentation Framework component that the Extensible Application Mark-up Language (XAML) is found. [4]

Being the subset of the .NET Framework, the .NET Framework for Silverlight provides libraries, garbage collection and the Common Language Runtime (CLR). Therefore when developing applications, some parts of the .NET Framework that contain the libraries are shipped together. The parts that are shipped include UI controls, XLINQ, Syndication (RSS/Atom), XML Serialization and the Dynamic Language Runtime (DLR). [4]

The third parts the make up the Silverlight Framework are the installer and the updater. The Installer facilitates the installation of the software for first time users. On the other hand the installer provides the service of updating the software with a very little impact on the whole software. [4]

3.3.2 Programming Features of Silverlight

The .NET Framework provides programming features for Silverlight making it more powerful and functional. The partial programming features list of the .NET Framework for Silverlight is discussed below.

Data is the core that most applications rely on. Silverlight supports Language-Integrated Query (LINQ) and LINQ to XML features which makes the manipulation of data from different sources easy and effective. With this feature XML data from different web services can be downloaded into the client machine and manipulated as needed. Though different from the library provided to manipulate data, the *base class library* is another .NET Framework libraries are included in the Silverlight. It is there to give functionalities such as string manipulation, input and output, collections, reflection, regular expression and globalization which are essential in programming. [4]

If there is a need to communicate with a web service from the applications to be built, then *Window Communication Foundation (WCF)* provides the means to have access to the remote services and data. The features that are provided are HTTP request and

response object, browser object, and support for a cross-domain HTTP requests, support for RSS/Atom Syndication feeds, JSON, PX and SOAP services. The other important part of the .Net Framework is the *Common Language Runtime (CLR)*, which provides all the facilities that provide memory management, garbage collection, type safety checking and exception handling. [4]

The other important feature of the .Net Framework is the *Windows Presentation Foundation (WPF)*. When the Silverlight SDK is installed into Visual Studio, more controls other than common controls integrated into the Studio are added. WPF provides a set of controls that make the UI design and functionality more versatile and easy to develop as well as to use by the end user. [4]

The .Net Framework feature named *Isolated Storage* is used to store local data in Windows phones. Through isolated storage, managed applications can create and maintain local storage. Though files can be stored locally on the phone and get manipulated, all the I/O operations are restricted to the isolated storage only, which in turn leaves local files of the operating system safe and secure from all unauthorized access.

In order to save an object in a Windows phone internal memory, a database or a file for later usage, it needs to be converted into a stream of bytes. Serialization gives the facility to do the conversion of an object into a stream of bytes. [4;5]

3.4 Windows Phone Application Development

In any application development, there is a better practice to deliver an application that could provide an optimum user experience. Windows phone provides good opportunity to develop applications that are used by clients in a better and exciting way. The applications can be built to be as efficient and engaging as application built for desktop computing platforms. In the following few paragraphs, the best design practices are discussed.

3.4.1 General Design Principles

Before starting development a Windows phone application, it is critical to know that Windows phone applications differ from traditional desktop and browser-based applications. The way users interact with applications on a phone is different and the screen is limited unlike in desktops. Another factor that should be considered while designing applications is that the user is always on the go. These two factors should not be taken as negative aspects while developing a mobile application but rather as advantages to leverage and use the phone for what it is best suited for, applications for a digitally connected mobile lifestyle.

In their book titled *Heuristic evaluation of user interfaces* (1990), Nielsen and Molich suggest the general criteria that can be used for inspection of UI usability. While developing an application in any platform, the design principles refined by Nelson and Molich are better considered than left aside. The following are those principles as directly quoted from their website [6;7]:

Visibility of system status: The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world: The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom: Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards: Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention: Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Recognition rather than recall: Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use: Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design: Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation: Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large. [6;7]

The following are the design guidelines that should be followed while developing a Windows phone application, and the connection of the guidelines with Nelson's and Molich's design principles are briefly explained.

Being *modern, simple, readable* and *minimalistic* is needed when designing application for Windows phone. The design principle that Windows phone has based on states that modern design in a touch application is less decorated, free of chrome elements, and minimally designed. In Windows phone, the content of the application creates the major interface. The design, navigation and way information is passed to the user should be kept as simple, readable and minimalistic as possible. Adding more graphical objects and cluttered screens makes the application harder to learn and will often block users from the simplified experience they ought to get from the application. Nelson described this in his *Aesthetic and minimalistic* design principle. [8]

The application should be "*on the go*" capable. Mobile application users are not most of the time stationed at one place like in their office or home. The application that is developed should be able to provide users maximum information and experience with little effort and interaction. Applications should provide new information without the user going deeper into the application's navigation hierarchy. This could be accomplished with the help of notifications and other facilities that the Windows phone platform. Nelson's principles of making a design *flexible, efficient, aesthetic* and *minimal* are essential to attain the *Be "On the Go" Capable* Windows phone application. [8]

Applications should be designed in a way that users can get to the core functionality of the application as quickly as possible. Once users have opened a Windows phone application, they should quickly navigate to the feature that they are interested in.

Applications should be designed to give a *clean* user experience. To have a *clean* design will solve the problem that comes with the limited screen size. It is always certain that people's touch would be precise to click around controls. Enough space around controls and contents should be there so as to tolerate clumsy fingers' touch. This will simply avoid the frustrations that come because of mis-taps. Windows phone applications should be designed in a way that does not challenge and test users' ability to touch a control precisely or forces them to make frequent taps. If Nelson's *aesthetic* and *minimal* design points are considered, then more space can be achieved around controls making the application more intuitive to use. The Project Emporia application can be taken as an example. *Figure 5* shows a screen shot of the application. [8]

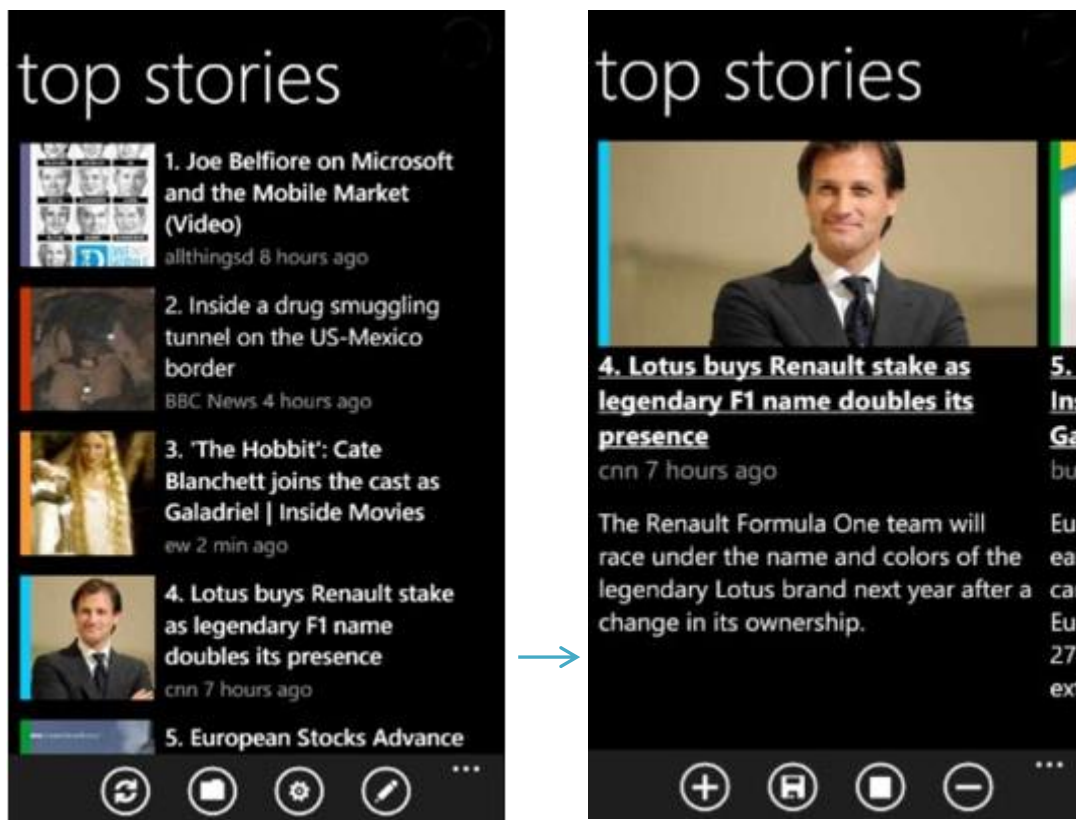


Figure 5. The Project Emporia application. Reprinted from Microsoft (2012) [8]

The Project Emporia is a Windows phone application that provides daily stories to readers. It clearly shows four summaries of stories at a time on the front page. As *figure 5* shows, the user is less probable to make a wrong touch on the screen. Touching on one of the four stories takes the user to a detailed page of that story. [8]

A design that has *motion* in it allows users to be more active and attracted. By design, object that move better inspire people to do more. Considering this concept while designing, Windows phone applications should keep the users active to the simple, familiar sensation of physical manipulation. In Windows phone using the proper types of movement of object, applications create the impression of tangibility by making on-screen objects flexible and interactive. These animations related to on-screen objects inform the user about navigation and operations. This design idea of Microsoft for Windows phone addresses Nelson's *visibility system status* and *match between the system and real world* design principles by showing application status with motion and letting users to easily recognize than recall an activity. [8]

A design should follow a *consistent* flow from the beginning to the end. Microsoft has published a design guideline so as to create a pleasant phone experience. The manner of interaction needs to be consistent within and across applications. Users should feel different routine of doing certain things in an application like navigating, changing settings, performing list deletion and other related interactions.

As stated in the design principles of Nelson, certain standards should be followed in order to create a consistent user experience. This will allow end users to get acquainted with application in a short period of time. Windows phones have their own way of performing some activity that is initiated by users and users are already used to some way of interaction that is set by Microsoft. So by the time a developer designs a Windows phone application, those interaction models available on the phone should be taken into consideration. Added to the above general design principles, a Windows phone application should be authentic and include innovative ideas that could sometimes replace a desktop version the application. [8]

3.4.2 Application structure and navigation models for Windows Phone

Every application, whether desktop or mobile, perform two tasks to be usable: present information and collect input. To provide application users pleasant experience in using his/her application, one should more emphasis on the design of his/her information and how it is navigated. The navigation model determines what is on each screen and how one get from screen to another. Each of the Windows phone interaction types are discussed below and one can pick a type based on the kind application that he/she is de-

veloping and the type of navigation that is determined to be best fit for the application. [8]

One application structure is the *central application hub with home page menu (panorama and pivot control)*. An application may have general of features in it. These different features can be organized to be in their district areas. These areas are sub-views of the application that the user will want to visit. From the central UI hub, the user can navigate to any area in the application, even to a different page of the application. Once taken to a specific subarea, the application can present a user with whatever UI is necessary at that point.

Microsoft provides a UI control called the Panorama control, which can be used as the central application hub. This control allows a user to navigate into all of the areas of functionality in the application. As an example, can be considered *figure 6* below, which is mobile application Windows Phone Marketplace. [9]

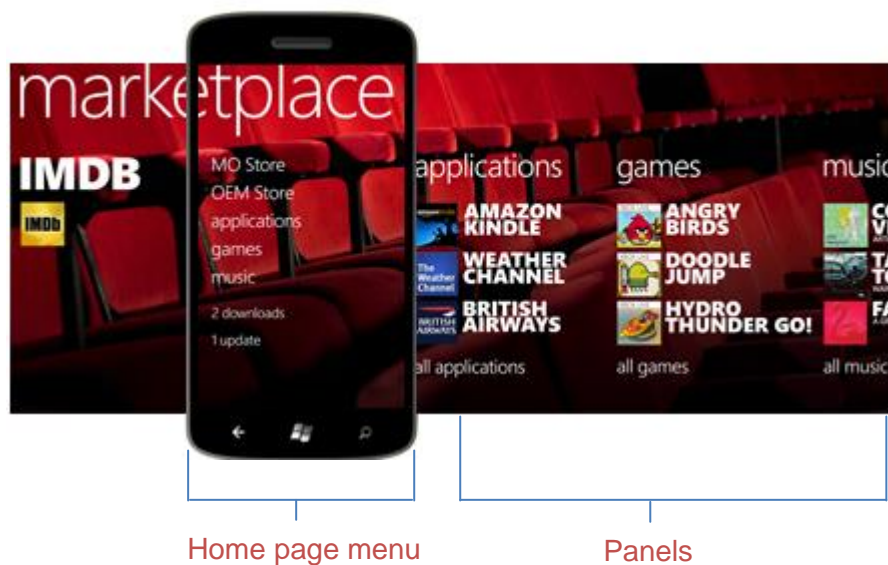


Figure 6. Windows Phone Marketplace mobile application. Reprinted from Microsoft (2012) [9]

As shown in *figure 6*, the Windows Phone Marketplace application is designed with central application hub containing *panorama* control as there are a lot of content to be displayed. The central UI hub is the area that comes first when the application is launched. In this case, the view that is being displayed in the phone's display is the central application hub. The links such as applications, games and music to the right of

the phone are the subareas that the user can navigate to. From the subareas, the user can go to another page based on the link that is clicked.

The second available application structure is the *Central application hub with panel areas (panorama control)*. This model is similar to *central application hub with home page menu*. It also has different areas of functionality to expose to the user. The difference here is that all the main UI can be presented at the top level without the need for a home page menu. When there is no need for subareas or views in an application, all the UI can be put in one set of horizontally accessible panels. To accomplish this, Windows phone provides a *panorama* control. It is our design method that avoids the home page from the *panorama* view. [10]

The third application structure that can be used if an application that has different pages of UI is to be developed, then *Application tabs (pivot control)* is convenient choices. This choice is good if an application is based around one single theme. If an application is only to show a single type of data with different type of fileting then Application Tab style is the most convenient one. The pivot control can be used to implement the Application Tab UI style. The control allows users to navigate right and left through each page called a *pivot page*. The CASS-Q Windows phone client is developed using the pivot control but as an example we can take a look at the Outlook Client application from Microsoft. *Figure 7* below shows an Application Tab style and in what kind of applications the style can be best used. [8]

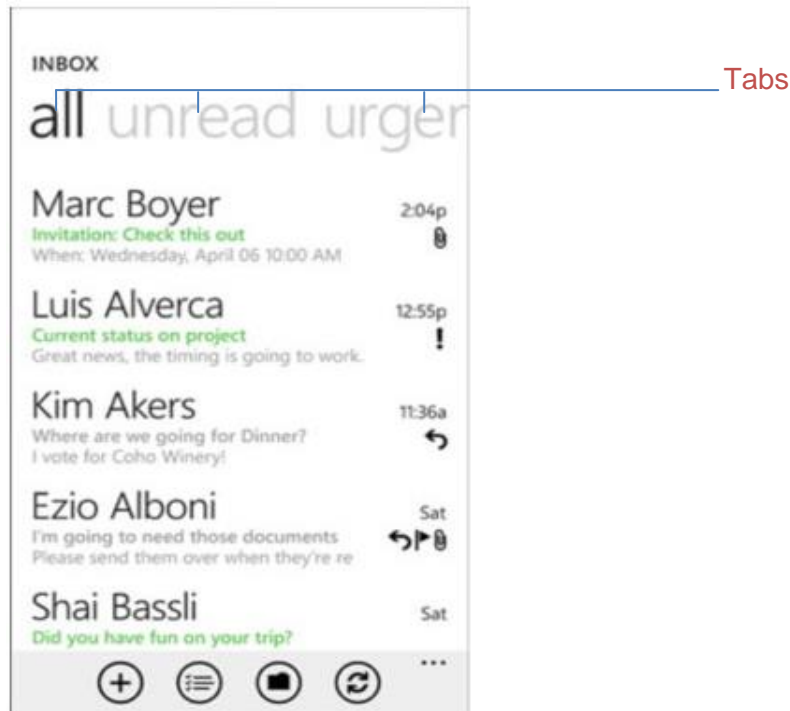


Figure 7. Outlook client application. Reprinted from Microsoft (2012) [8]

Microsoft Outlook uses the style because it shows a single type of data, which is a list of emails with different filtering applied on each pivot page. This makes finding of urgent email easy and fast.

3.4.3 Essential graphics, visual indicators and notifications

In Windows phone, the elements in start tile, splash screen, icons, controls and navigations that may be included in an application should draw attentions to tasks, priorities and/or operations in an application and be able to give relevant information in a more appealing manner. All elements in an application should have a purpose for being included. No graphics should get introduced for mere decorative purpose. This helps in gaining simplicity which is one way to add attractiveness to an application. The following paragraphs mention about the controls and graphics that are most commonly used in Windows phone application development. [8]

A *Start Tile* is an easily recognizable visual shortcut that can launch an application. In Windows phone, a user can manage which tiles to show in Start screen by “pinning” an

application's Tile to the Start view. The size of a Tile image should be 173 pixels x 173 pixels at 256 dpi and the format should be PNG. If the image is larger or smaller than what is specified, then the extra part will be cropped or it will be scaled up to meet the requirement. If an application does not have a Tile image, then Windows phone will assign a generic Tile image. [8]

As there are simple applications, there are many heavy applications that may take time to load for so many reasons. The moment an application could take can be used as an opportunity to introduce a user with the application and for this purpose a *splash screen* can be used. As a splash screen gets displayed for few seconds, no text is advised to be included rather a splash screen should be a graphical advertisement to the application that is being launched. [8]

The other essential graphic component in Windows phone is the *visual indicators*. There are three different types of *visual indicators* to show task progress, scroll views, signal strength, battery life and other relevant information. They are the *scroll indicator*, *scroll viewer* and *progress indicator*. The *scroll indicator* is used to indicate a content that exceeds the bounds of the visible page and a user pans or flicks. The *scroll viewer* allows users to navigate to content that is not directly viewable within the frame of the application like long section of text or an image. A *progress indicator* tells users the progress of some activity in an application. It is a Windows phone control that is integrated with the *Status Bar*. [8]

There are two primary components in Windows phone OS chrome, the *status bar* and *application bar*. As the name indicates, the status bar is a bar where system level status information is placed in a reserved portion of the application workspace. Based on what is happening, the bar automatically updates to provide different notifications and give user system-level status information. Below *figure 8* shows what the status bar looks like.



Figure 8. Windows Phone status bar.

As the *status bar* is a system-reserved place and the information displayed is controlled by the operating system, it cannot be modified but can be hidden. The status information that are displayed on the *status bar* are the signal strength, data connection, call forwarding, roaming, wireless network signal strength, Bluetooth status, ringer mode, input status, battery power level and system clock. [8]

The other main graphical component in Windows phone application is the *application bar*. From the *application bar*, a user can access tasks that are most frequently performed in an application through icons. In the application bar, a developer can place up to four icon buttons. Added to the icon buttons, an optional context menu called the *application bar menu* can be placed and it gets active when a user taps the visual indicator of the sequential dots, or flicks up the application bar. Tasks that are not frequently used can be placed in the application bar menu. To illustrate more about application bar and application bar menu, *figure 9* is added below. [8]

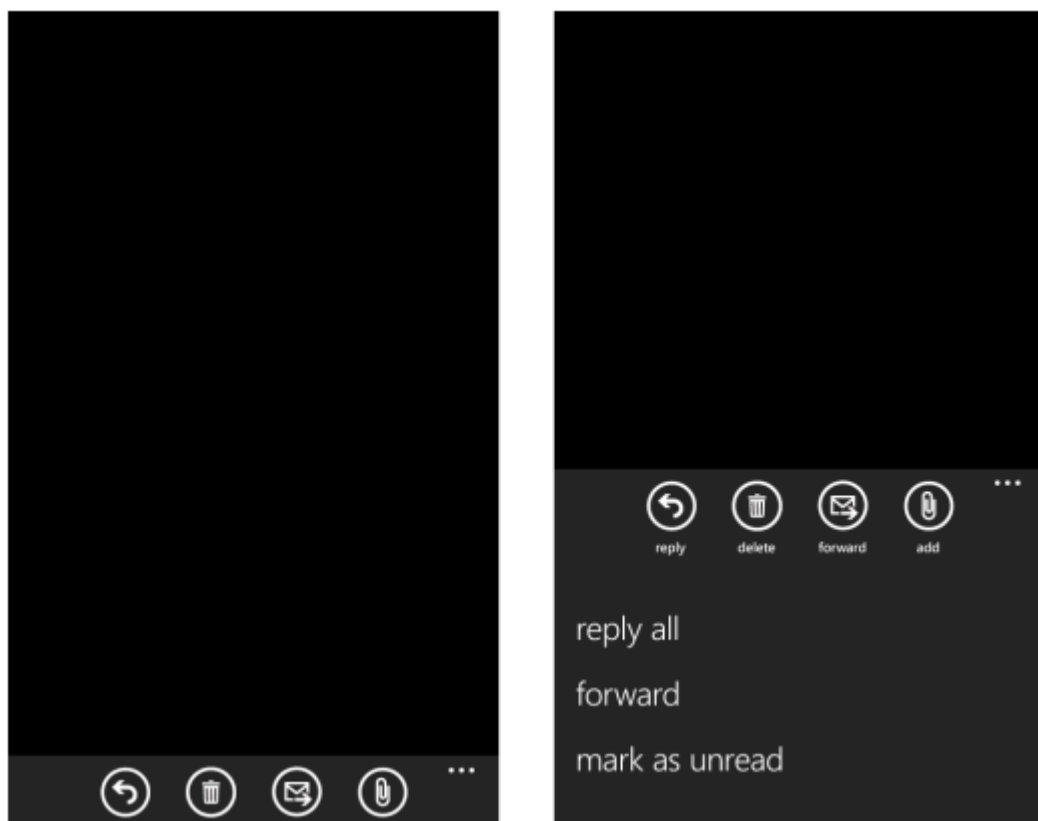


Figure 9. Application bar and application bar menu.

As figure 9 shows, the *application bar* can contain a maximum of four icon image, each of them related to one frequently used task. The icons can have a short text below them telling more about the purpose of each icon. The three dots are used to make an *application bar menu* visible if it is added to an application. Clicking the dots or sliding the bar from the dots' side will make the menu visible. To avoid scrolling, it is advised that the menu is better with a maximum of five items but according to the need more can be added. [8]

Push notification is the other essential graphical component that gives a notification service to users. Some services from the cloud may have notifications that should get to a mobile device. For this to happen, a more reliable and persistent channel is needed. When a cloud service needs to send a push notification then it sends a notification request to the *push notification Service*, which in turn routes the notification to the application or the device as a tile, toast or raw notification.

Tile notifications come to the *start tile* and they don't interrupt the normal work-flow of the user. They are used for awareness-only notifications like weather, email arrival and the like. *Toast notifications* can also be used to give some awareness related information to the user but they are meant to notify a user with notifications that may be generated by a web service. They are used specially to request an action from the user. *Toast notifications* are displayed for 10 seconds only before disappearing again. Raw notifications are those which require action inside an application. They are fully controlled by the application and affect only that application. They can be generated by the application itself or a web service.

3.5 Requesting for, parsing and building XML in WP

Countless applications are being developed and released on a daily basis but not only using one platform and language. At the same time, the need of data exchange between different applications has reached its apex point. This has created the need for convenient way of data exchange between different platforms and languages. XML is one major way to carry data between different applications. So it can be said that XML is a means to carry data between applications as well as a way to store data. In this subsection, concepts about XML web request, parsing XML and building XML are discussed.

3.5.1 XML web request

In this project, there was a need to make a web request to get an XML file from CASS-Q server. For a web request to be valid and successful there must be a Uniform Resource Identifier (URI). Some server specified with the URI can provide the functionality expected of them without asking for a request for authentication from the client making the request. However, at times there might be a need for strict authentication.

There are layers that a request passes through to reach the handler on the server and the same is true for the response to reach the client making the request. Any time a client side application makes a request for a web service, and then it passes through all the layers that exist in the client-server architecture in the area of a web service. Any time a client application wants resource from a web service then there should be a programmable URL through which a request for data can be made. As shown in *figure 10*, a request for a resource is made through the proper request object from the client application and the object that is responsible to handle the request processes and sends response back to the requesting client. In this project's case, an XML-HTTP object on the client side handles the request and the HTTP object on the server sides is responsible to process the request and respond.

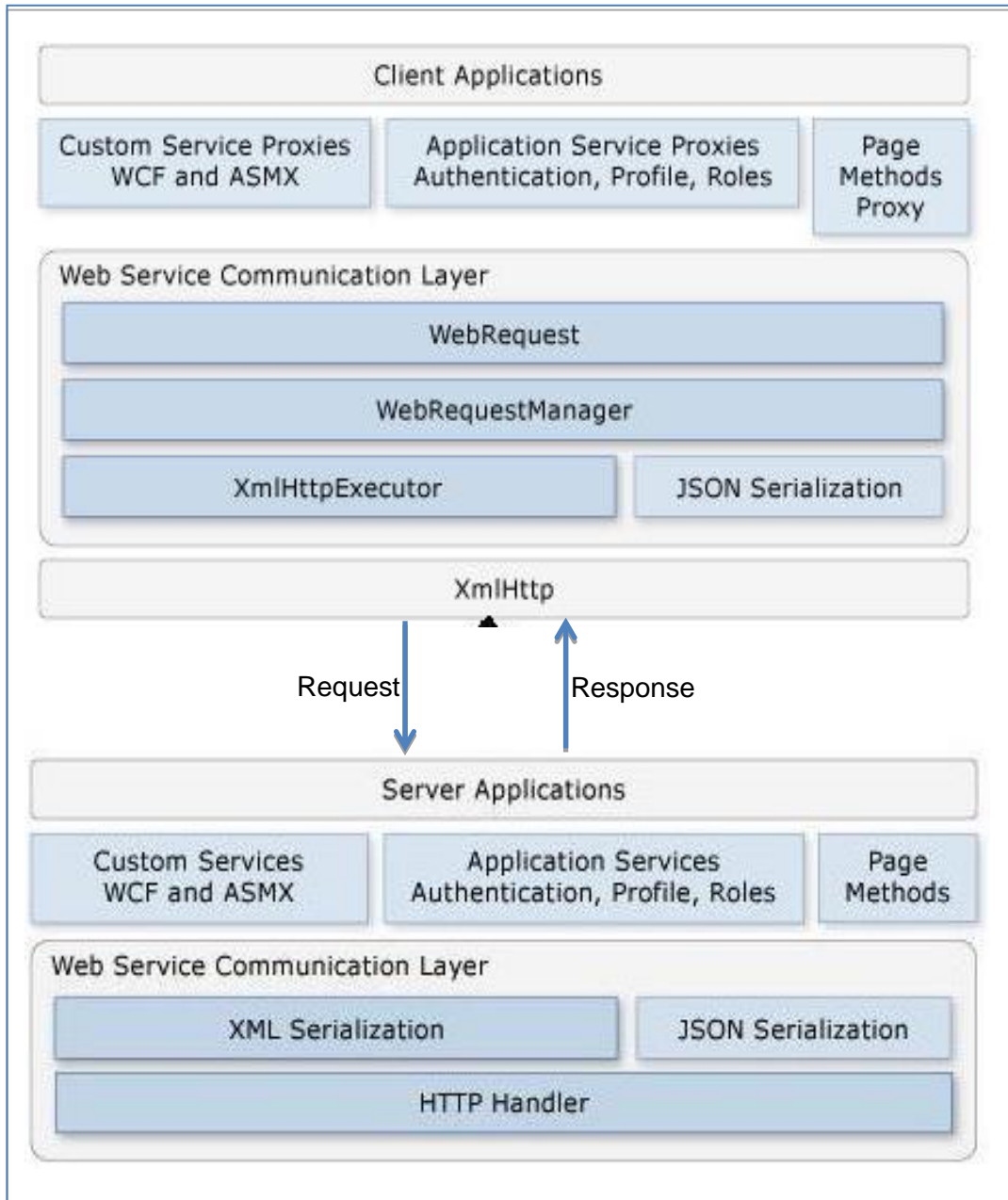


Figure 10: Web service architecture. Reprinted from Microsoft (2012) [4]

After the XML is retrieved, proper parsing should be implemented in order to get the needed data from the XML file. It is up to the platform and programming language used that the parsing method depends on.

3.5.2 Parsing XML

Sometimes parsing XML files can be a tiresome task that can be time consuming and tricky. Before .NET came into play, programmers were forced to read XML as text file

line by line and use string functions and possibly regular expression, so as to get the data embedded in the XML structure. [10]

But because of the powerful new ways of parsing XML are included in the .NET framework, programmers no more spend hard time reading XML text file line by line to get the data. In .NET, there are five fundamentally different ways to parse XML, and the suitability depends on the particular development situation involved and the style of programming preferred. Each technique is based on a different .NET Framework class and its associated methods are:

- *XmlTextReader*
- *XmlDocument*
- *XPathDocument*
- *XmlSerializer*
- *DataSet*.

Parsing an XML file with *XmlTextReader* has a traditional, pre-.Net feel. The *Read()* and *ReadElementString()* and *GetAttribute()* methods, which are important in reading XML file with *XmlTextReader* walk through the XML file sequentially. Using *XmlTextReader* to parse XML file is effective and appropriate when the file is relatively simple and consistent in its structure. [10]

If there is a need to extract data from XML file in a non-sequential manner, then *XmlDocument* is the preferred .NET class for parsing the XML file. The *XmlDocument* class is modeled on the W3C XML Document Object Model and has a different feel to it than many .NET Framework classes. [10]

The third object, *XPathDocument*, has features from both *XmlTextReader* and *XmlDocument* giving it a hybrid feel that is both procedural and functional. Parts of the XML document can be selected using the *Select()* method of an *XPathNavigator* object and also move through the document using the *MoveNext()* method of an *XPathNodeIterator*. Using the *XPathDocument* is suitable when the XML is highly nested or has a complex structure. It is also good to consider using the *XPathDocument* if other parts of the application being developed use this class so that the code maintains a consistent coding look and feel. [10;11]

The *XmlSerializer* is the method that gives a very elegant solution to the XML file parsing problem. Compared to all the other four objects mentioned above, this object provides a mechanism of parsing where the details are hidden. It has the highest level of abstraction hiding the algorithmic details from a programmer. Though it makes parsing a lot easier, it has its downside; it gives less control to the programmer on parsing. [8;9]

The *DataSet* object operated at a middle level of abstraction compared to the other parsing methods mentioned. It has a very relational database feel. The *ReadXml()* method hides a lot of details but traversing through relational tables is a must. It is appropriate to use the *DataSet* object when the application being developed is using ADO .Net classes so as to maintain consistency in the look and feel of the code being written. If performance is an issue then, using the *DataSet* would create a problem as it has high overhead.

XML data files are key components in Microsoft's .Net developer environment. Each of the aforementioned techniques significantly differ from each other in terms of coding mechanics, coding mindset and usage scenarios. [10]

3.5.3 Building XML

Some application may contain a task that involve creating XML file in order to save data locally, pass it to another program as an output or upload it to a server. To accomplish creating XML document *XmlWriter* is one of the oldest and most reliable methods available in the .Net Framework. It is this method that is used in the project and the techniques and mechanisms involved are discussed in *section 8.5*.

4 Requirement analysis

It is always because of some needs that software is developed and those needs can be modeled properly if the requirements are properly studied in a more organized and professional manner. A requirement analysis gives the opportunity for a developer to get a clear understanding of the problems to be solved.

This chapter covers the requirements of the CASS-Q client application. Before going deeper into the requirements, the participants of the project and the main goals that are to be attained are covered briefly.

4.1 Stockholders

Stockholders are those who are involved with the client application in a direct or indirect way. The participating bodies or those involved in the project in different areas are listed below.

- Client: University of Helsinki
- End users: People taking part in a research as respondents
- Developers: Students
- Development support: Advisors of the CASS-Q project
- Usage area: Europe
- Validators: Advisers of the project, personals responsible about CASS-Q tool from University of Helsinki
- Maintenance and support: Students and advisors of the CASS-Q client
- Project license: Copyrighted to Helsinki Metropolia University of Applied Sciences under the General Public License (version 2.0 or higher)

Some are directly involved with the test, usage and publication of the application and others are involved with the development and maintenance of the application.

4.2 Goals

The main goal of the project is to develop a CASS-Q client for the Windows Lumia phone. The specific goals are listed below:

- The client application should download survey questions
- The client application should display the research questions
- The client application should let an end user answer survey questions

- The client application should be able to upload respondents' answers to a CASS-Q server as long as data connection is available on the phone.

For the application to be considered complete, the above four mandatory requirements should be fulfilled.

4.3 Functional requirements

In the following section, the different requirements that are fulfilled by the CASS-Q mobile client application are discussed in detail using different mechanisms of portraying the requirements in software engineering.

4.3.1 Context Diagram

The context diagram in *figure 11* gives the overall description about the interaction between the client, the mobile client application and the server side application of CASS-Query tool. The mobile should be connected to the Internet in order to download the survey questions and upload the answers to the CASS-Q server.

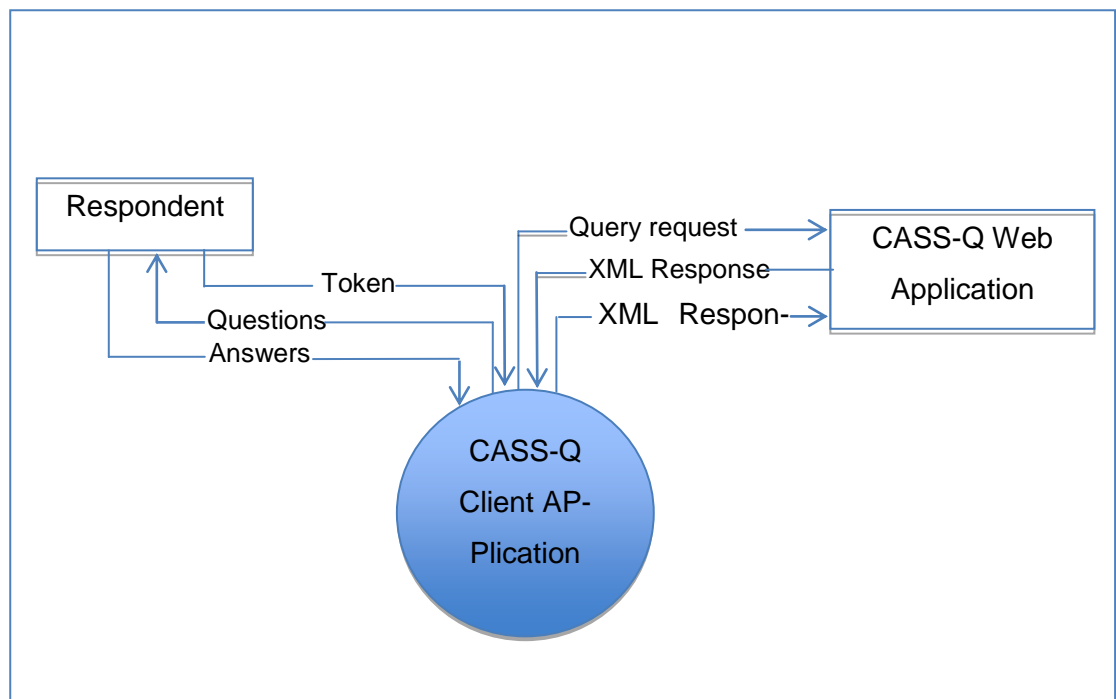


Figure 11. Context diagram of CASS-Q Windows mobile client application

Before a request is made by a respondent to download survey questions from the server, a research administrator should create a survey. This can be accomplished using the web application of the CASS-Q system. The survey is created with a unique survey id, so that answers to that survey can be related to later identify which answer is meant for which survey.

After providing right credentials, through the CASS-Q mobile client application, the respondent downloads the queries assigned to him/her by a research administrator. Then, after all the questions are answered and validated, they are submitted to the server side application that handles all the necessary parsing and storing of the data to be analyzed by the researcher afterwards.

4.3.2 Input, operation and output requirements

The inputs, operations and outputs of the CASS-Q windows phone client application are as follows:

Inputs to the CASS-Q Windows phone application are the token, XML file from the CASS-Q server and respondent's answers. The mobile client application should pass through the necessary authentication process. That is possible only if the respondent provides the correct token for a specific survey. After being authorized, it should be able to download the survey questions from the CASS-Q server application. After all the questions are displayed to the respondent, then the client application should enable the respondent to enter his/her answers. Therefore, the token, XML data from the server and the user answers are the inputs to the system.

Operations that the CASS-Q System performs are downloading the XML file, parsing XML, building UI based on the parsed data and uploading answers as XML and media files. The mobile client application should make a request when it is online and be able to download the XML file from the server. After downloading is completed, the XML file should be parsed and stored. The researcher can create nine types of questions. Based on the question type, which can be found in the parsed data, the UI should be generated and displayed. This makes the UI dynamic based on the the type of questions downloaded from the CASS-Q server.

Output of the CASS-Q System is the XML and media files. After the respondent finishes answering the survey questions, the client application should be able to send the answers to the server as an XML file and the media files with the supported file extensions.

4.3.3 Dataflow Diagram

At its simplest, a data flow diagram looks at how data flows through a system. It is concerned about the “from and to” concepts of data flow. It helps more on identifying existing business process so as to have a strong ground while advancing to business process reengineering. *Figure 12* below is that dataflow diagram of CASS-Q client application.

As shown in *figure 12*, the application starts with a *Request for Survey* process. The request is made by sending token to the CASS-Q server using the client application. After an XML file is downloaded successfully, it should be parsed to get the data and the *Parse XML File* process is responsible for this particular task. After the XML is parsed then the data should get stored for later use and the *Store Questions Temporarily* represents this process. The *Generate UI* process then uses the data obtained to generate user interface based on the type of questions downloaded from the CASS-Q server. After the proper UI is displayed, then the respondent gives answers to the survey questions and *Accept Answers* process accepts and validates user input and passes the input data to the *Generate and Upload XML* process, which generates and uploads XML to the server.

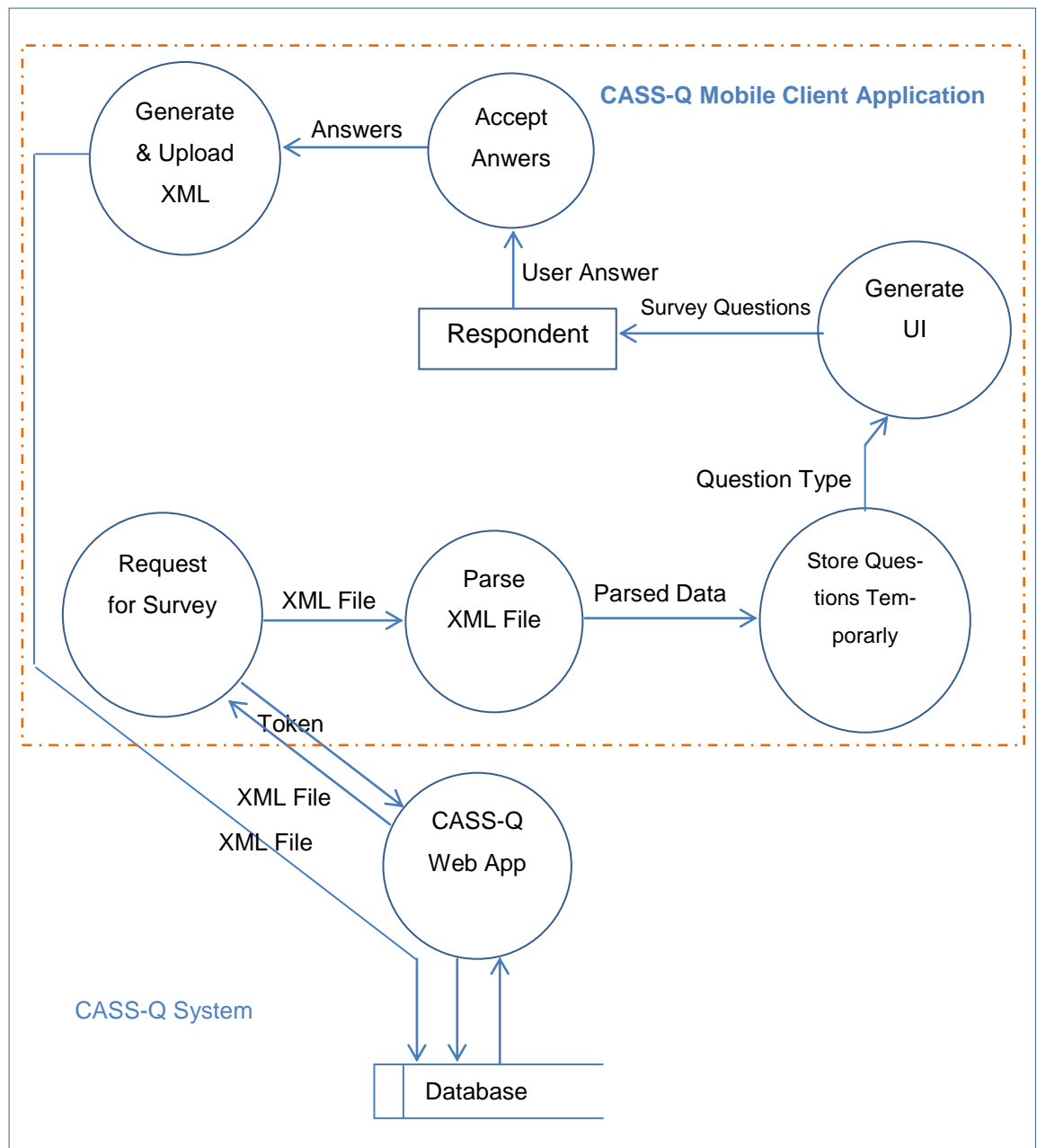


Figure 12. Data flow diagram of the CASS-Q client application

As shown in *figure 12*, the CASS-Q mobile client application has different parts that work in conjunction. The URL to which a request by the *Request for Survey* process is:

<http://54.247.115.29/cass/MobileIO/XMLGen.php?uid=t46c3890e5f1>

The alphanumeric value that comes at the end of the URL is the token that the server uses to authenticate the client with.

The *XMLGen.php* is the server side PHP file that authenticates the request and provides the necessary XML file to the client application.

The generated XML file will be sent to the server application using the following URL:

<http://54.247.115.29/cass/MobileIO/mobileanswer.php>

As shown in figure 6, the processes encompassed in the dashed rectangle are the ones that make up the Windows phone client application. Together with the server side CASS-Q application, the client application makes the functional concept of the whole system creating the CASS-Query tool.

4.3.4 Use case

A use case describes a sequence of interactions between a user and application. *Figure 13* describes the use case of CASS client application. There are two actors involved in the system, the client and the server side application which provides the web service to the respondent. Each use case of the diagram is described in detail.

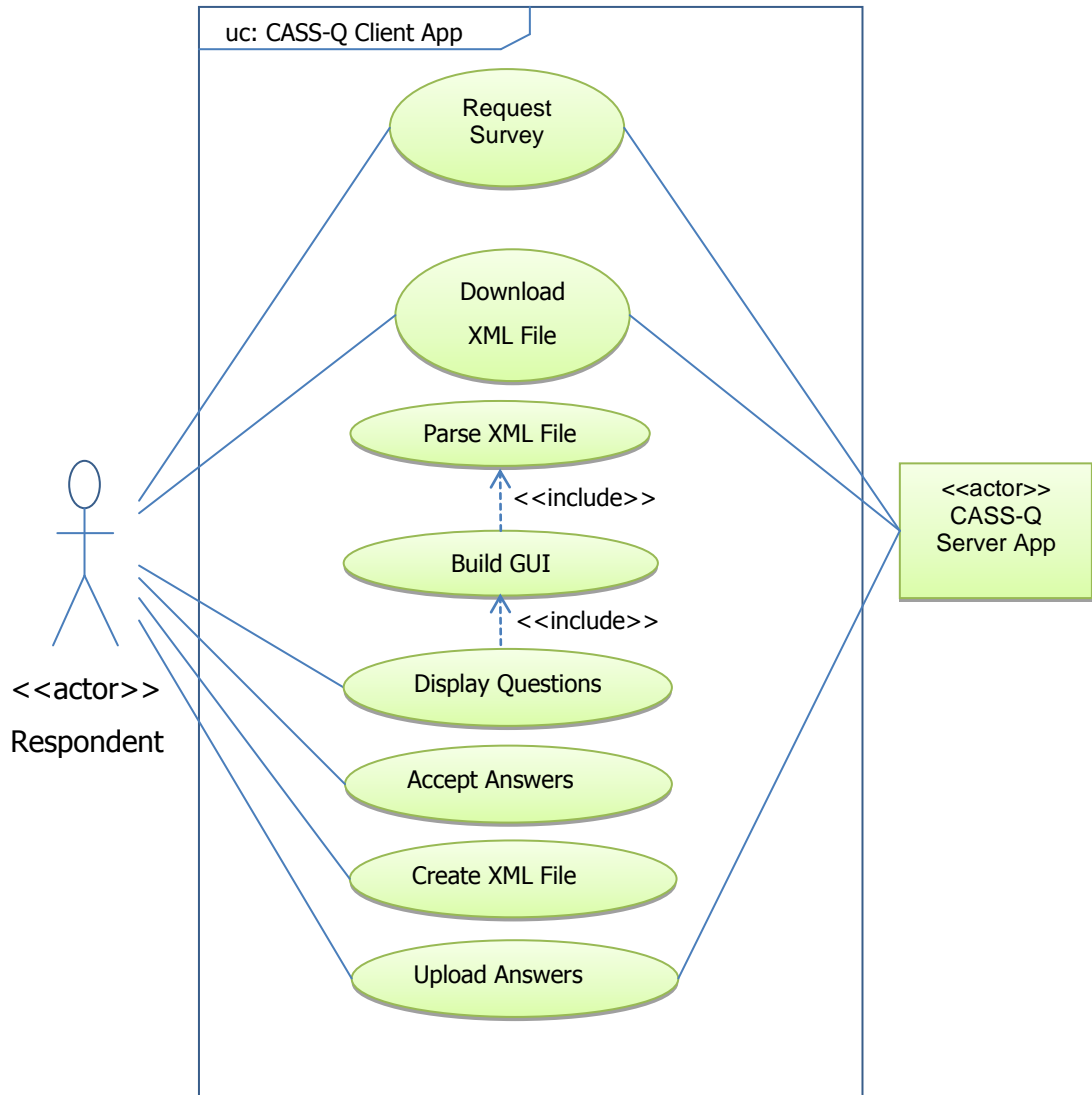


Figure 13: Use case diagram of CASS-Q client

After the user enters a valid token, the application logs in and makes a request for an available survey from the server. *Table 1* describes the use case related to this activity.

Table 1. Use case description for requesting survey

Use case ID:	UC1
Use case name:	Request survey
Description:	The application makes a request to download survey
Pre-condition:	Token should be provided and survey should exist
Standard flow:	<ol style="list-style-type: none"> 1. User inputs token 2. Users presses the Log in button
Post condition:	Client should be able to download XML file

After the server authenticated the user and finds a survey that match the user, it lets the client application download the XML related. *Table 2* shows the use case related to downloading an the XML.

Table 2. Use case description for downloading XML

Use case ID:	UC2
Use case name:	Download XML file
Description:	The application downloads XML file from the server
Pre-condition:	The token provided in UC1 should valid and survey should exist
Standard flow:	After the user logs in, the XML is downloaded
Post condition:	XML file to be parsed will be ready

After the XML file is downloaded, the client application parses to get the data to display to the user. *Table 3* describes the use case related to parsing XML file.

Table 3. Use case description for parsing XML file

Use case ID:	UC3
Use case name:	Parse XML File
Description:	The downloaded XML file is parsed and the data is made ready
Pre-condition:	UC2 must happen
Standard flow:	<ol style="list-style-type: none"> 1. Data is extracted from XML 2. Data is temporarily stored
Post condition:	XML file is parsed and data is stored for later use

After all the necessary data is obtained through parsing, the user interface is built in a way it lets the user understand the survey questions easily. *Table 4* describes the use case related to building the UI.

Table 4. Use case description for building the UI

Use case ID:	UC4
Use case name:	Build GUI
Description:	User interface is built based on the type of questions downloaded
Pre-condition:	UC3 must be done
Standard flow:	<ol style="list-style-type: none"> 1. Temporarily stored data is accessed 2. Question type is identified 3. Proper UI controls are added 4. UI is displayed
Post condition:	Proper UI is displayed to end user

After the proper UI is displaced, the system lets the user to input replies which would later be sent the server as answers. *Table 5* describes the use case related to accepting user inputs/answers.

Table 5. Use case description for accepting user input/answers

Use case ID:	UC5
Use case name:	Accept Answers
Description:	After all the questions are displayed, the client accepts user inputs
Pre-condition:	UC4 must happen
Standard flow:	<ol style="list-style-type: none"> 1. UI is displayed 2. User inputs answers
Post condition:	All the answers should be provided and be ready for upload

After all the questions are answered, the user has to submit the answers to the server and before being uploaded, an XML is constructed. Table 6 is the description of the use case related to building an XML.

Table 6. Use case description for building XML from user answers

Use case ID:	UC6
Use case name:	Create XML

Description:	XML will be built from the answers of the respondent
Pre-condition:	All questions should be answered
Standard flow:	<ol style="list-style-type: none"> 1. User answers all survey questions 2. XML is built from the answers
Post condition:	An XML that is to be uploaded to the server is built

After the XML is built, the answers are uploaded to the server in the form of XML as the server accepts only an XML. *Table 7* shows the description related to the use case about uploading user answers.

Table 7. Use case description for uploading user answers

Use case ID:	UC7
Use case name:	Upload Answers
Description:	User answers are uploaded to the CASS-Q server
Pre-condition:	A valid XML should be built from user answers
Standard flow:	<ol style="list-style-type: none"> 1. User clicks the upload Application Bar Content Menu 2. XML is uploaded to the server 3. Media files are uploaded to the server
Post condition:	Confirmation about the upload success if displayed

The other task that the application is able to do is the saving of the token for the next login. After the user is logged in, he/she can save the token by going to the settings. *Table 8* describes the use case related to token saving.

Table 8. Use case description for saving token

Use case ID:	UC8
Use case name:	Save token
Description:	A token is saved so that the user doesn't have enter it the next time he/she has to log in
Pre-condition:	A valid token should be provided and client is logged in
Standard flow:	<ol style="list-style-type: none"> 1. User clicks the settings Application Bar Content Menu 2. Settings popup windows pops up 3. User clicks the save button in the popup window
Post condition:	Token is saved

4.3.5 Sequence diagrams

A sequence diagram is a Unified Modeling Language(UML) which is used to elaborate the interaction between different processes in a program.

With the following sequence diagrams, more formal refinements made while developing the CASS-Q client application are shown. When making a request for survey questions, getting the questions from the server and submitting the answers back the server application, the CASS-Q client application goes through different sequences. [12]

Figure 14 below shows the interaction between the objects involved in making a request for survey questions from the CASS-Q server.

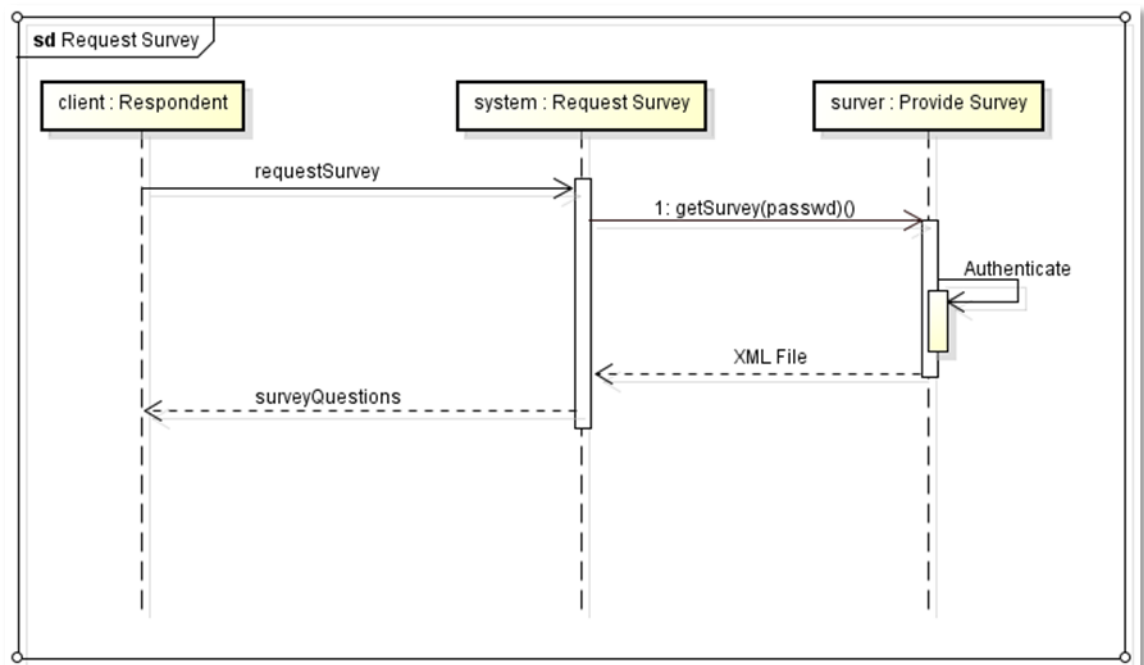


Figure 14: Survey request sequence diagram of the CASS-Q client application

The respondent makes a request by providing the proper authentication parameter named as *token* to the CASS-Q server which authenticates the client and finally provides the survey as an XML file.

Figure 15 below shows the sequence in which a graphical user interface is created and displays survey questions in an easy way to understand and lets the user interact smoothly.

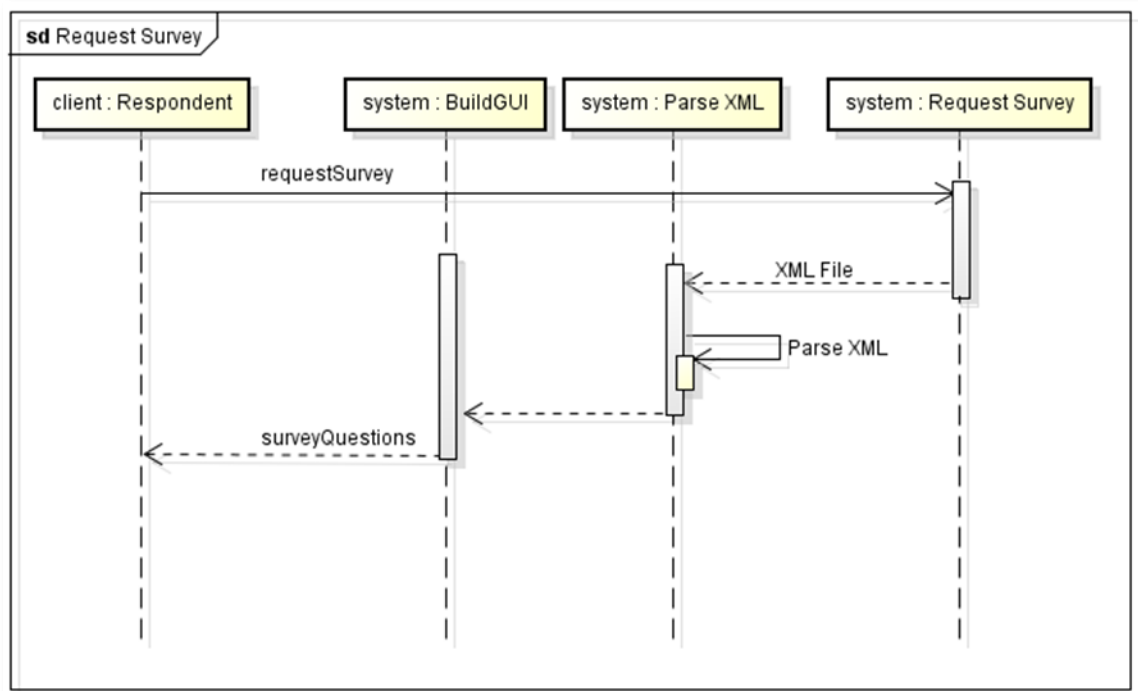


Figure 15: GUI displaying sequence diagram of the CASS-Q client application

After downloading the XML file, the application adds UI controls to the phone screen based on the type of each question.

Sequence diagram for uploading a respondent's answers

After completing answering all survey questions, the CASS-Q mobile client application uploads the answers as XML and media files with supported file extensions. As shown in figure 16, generation of XML file should come first before sending the answers. Then at the end, confirmation about the upload is displayed.

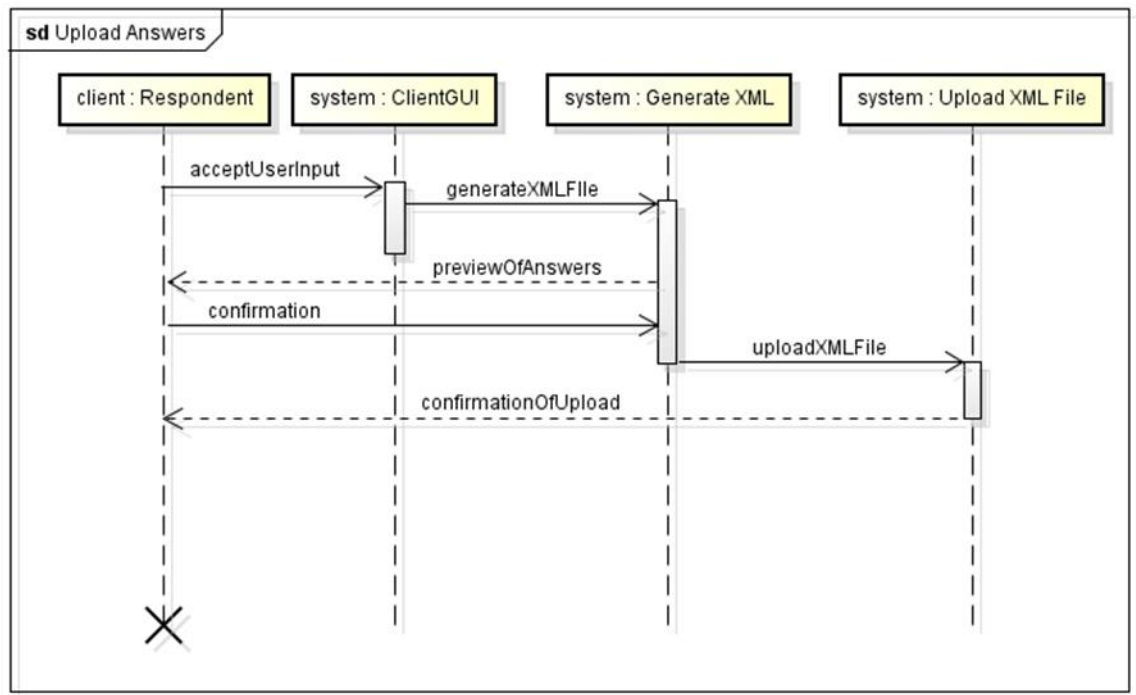


Figure 16: User answer uploading sequence diagram

4.4 Non-functional requirements

The following requirements are not needed for the CASS-Q client to run and function properly but are worth considering in order to measure the quality of software.

- Compliance of the Windows phone coding and design guidelines
- Definition of well-structured software architecture
- Definition of a well-structured graphical user interface
- Compliance of the software product quality requirements and evaluations according to ISO/IEC 25000 which is concerned with functionality, reliability, usability, efficiency, maintainability and portability.

5 Functional specifications

Based on the requirement analysis described in *chapter 4*, specifications and the requirements to be fulfilled were obtained. This chapter covers the requirements of the project which later helped as criteria for the acceptance of the software developed.

5.1 Target achievements

Before starting developing the CASS-Q clients, there were certain conceived achievements which had to be attained at the delivery of the product. At the same time, there were some other desired achievements which could make the software more functional.

5.1.1 Mandatory criteria

The *device* on which the application can run must have the minimal requirements concerning the OS and hardware. So the device is the main consideration and it must have some capabilities that need to be fulfilled for the application to be functional. The application runs on Windows phone OS version 7.1 or higher. It as well makes use of the device hardware such as the camera and voice recorder and needs data connection.

The *business logics*, which are the other main criteria that are followed while developing the CASS-Q Windows phone application, are the following:

- Surveys are downloaded from the server after proving valid token.
- The answered surveys are sent back to the server application.
- The client application is implemented in a software architecture that ensures the expandability as well as reusability of software components.
- Photo, video and audio files can only be found on the server after the user exits the application.

The *graphical user interface* plays a crucial role in the usability of an application and the criteria that are considered in the area are the following:

- Proper controls are displayed to show questions and accept answer from user.
- Application bar items are provided.
- Menus items are provided to upload answers, change setting and see help

- New icon is provided for CASS-Q client application.

5.1.2 Desired criteria

Even if what is currently implemented is giving the services that it was required for, it is always a good to see further ideas that could be added in the future. The following are ideas that can be integrated into the developed application in the future.

- Ability to let a user answer questions while offline and upload them while online
- Notifications service to let users know if a new survey is available.
- User statistics are created and graphically displayed.
- The application demonstrates its functionality at the first start.

5.2 External interface requirements

For the application to run properly and give the functions it is supposed to give fully, there are certain hardware and software requirements to be fulfilled.

Hardware requirements are those needs which are directly related to the kind of device specifications needed for the application to run. As the application is developed for Windows based mobile phones, basically the phone should be a Windows phone. Next to that, some of the survey questions need multimedia to be answered and this requires the phone to have some extra hardware features such as camera and voice recording facilities. The following are the hardware specifications taken from Microsoft for Windows phone that is running Windows OS 7.1[13]

- A common set of hardware controls and buttons that include the Start, Search, and Back buttons.
- A large WVGA (800 x 480) format display capable of rendering most web content in full-page width and displaying movies in widescreen.
- Capacitive 4-point multi-touch screens for quick, simple control of the phone and its features.
- Support for data connectivity using cellular networks and Wi-Fi.
- 256 MB (or more) of RAM and 8 GB (or more) of flash storage.
- Primary camera.
- Front-facing camera.

Software requirements are the software environments needed for the application to run properly. The CASS-Q Windows phone client application needs Windows phone OS. The version of the OS should be 7.1 or greater. Operation of CASS-Q client is not optimized for version prior to 7.1.

5.3 Product Functions

In this section, the functions that are provided by the application are listed with brief descriptions.

Logging in a user is a functionality that the CASS-Q client application provides to enable a respondent to log into the application after providing valid token. Based on the type of survey he/she is assigned for, survey questions are displayed or proper message is displayed. After a user is logged in, he/she is provided with a list containing all the questions in a survey. *Selecting question* allows a user to get the detailed view. When the user selects a specific question from the home screen which contains all the list of questions, the UI goes to the view where the selected question is displayed in detail. The user is able to provide answers.

Accepting questions enables respondent answer questions. User gives answer to the question displayed in detail view. Based on the type of question, the answers and the way to answer it vary. For media type questions, the application launches the camera or sound recorder in order to let the user take a picture, record video or sound. The other functionality that the application provides is the *editing of answer*. The user is able to edit the answers even if they were answered prior. From any detailed view, the user can go back and forth or directly to the home screen using the home icon on the application bar and select the question to edit its answer.

After all the questions are answered, the user is able to submit the answers to the survey. *Submitting survey answers* is the other main functionality the application provides so that the answers get uploaded to the server. Therefore, after all the survey questions are answered, the user can submit the survey using the *Submit menu item* that is located under the Application Bar Content Menu of Windows phone.

A user is also able to *view help* by tapping on the *Help* menu that is under the Application Bar Content Menu. Added to that user can *save his/her token* by tapping on the

Settings menu that is under the Application Bar Content Menu. Other than the token, the URLs to make a survey request as well as submit answers to is displayed under the Settings but are not made editable as end users should be able to change URLs even the server location changes. Changing URL should be done by releasing another version of the application on the Marketplace.

5.4 Product Data

The first data to the application is the *questions* which are downloaded from the server. The questions parsed from the XML file downloaded are temporarily stored in one of the structure definitions inside the *Model*. The questions are lost when the application is closed.

The *answers* are the other product data. They are also stored temporarily according to the structure defined in the *Model*. The answers are lost after the survey is submitted or the application is closed. Answers in the form of media files are stored temporarily according to the structure definition inside the Model and are lost after the application closes or the survey is submitted to the CASS-Q server.

The other product data is the *token* used by the user to log into the system. User's token is stored inside the phone in an isolated storage that is allocated for the CASS-Q client inside Windows phone. The data can be deleted by the user only once it is saved.

5.5 Development environment and Global test scenarios

To develop the application, there were both hardware and software requirements that are specific to Windows phone application development. While developing, the goals that were meant to be achieved were listed out, in order not to leave some functionality out.

The development environment variables that were needed while developing the CASS-Q client are Microsoft Windows 7 Professional Service Pack 1, Windows phone SDK 7.1 Assemblies, Silverlight 4 SDK and DRT and WCF Data Service Client for Windows Phone. The tools that were used are the Microsoft Visual Studio 2010 Express for WP, Windows Phone Emulator and Microsoft Expression Blend SDK for Windows Phone OS 7.1. The last development environments are related to a hardware and they are the standard personal computer and Nokia Lumia 800 smart phone with the Windows OS version 7.1.

The global test cases are concerned with the main functionalities that the application should give after completion. The application should be able to meet the test scenarios in order to be called functional. The following is the list of test cases that were taken into consideration.

- Log in: The user logs in.
- Navigate between questions: The user navigates through questions.
- Select and answer question: The user selects a question and answers it.
- Edit answered question: The user selects and edits an answered question.
- Send answer: The user sends answer back to the server.
- Save token: The user saves token.

6 Graphical design and Software architecture considerations of CASS-Q client

Based on the analysis done about the requirements of CASS-Q client in section 3.4.1, the considerations made on the UI design and the software architecture while developing the application are discussed in this chapter.

6.1 Graphical user interface

A Graphical User Interface (GUI) is a human-computer interface which allows users to interact with an application giving a high level of abstraction to the commands carried out by the application. GUIs stand in sharp contrast to command line interface (GLIs) which is only gives a text based interaction with an application.

While iterating through the possible GUI considerations, three suggestions were brought to a table discussion. The first one was a list view which could contain all the questions and the answering mechanism below each question. However this design idea had a down side in terms of letting the user navigate to a specific question. The second option was similar to the first suggestion but having a collapsible area under each question to hide and unhide the controls which could allow the user input answers. However adding the hiding and showing the controls only save scree area leaving the confusion that may be created during navigation intact. The third and accepted option was the one having all questions listed in one page, but having pages with a detailed view of a question. So the user only has to click on any question in the list to navigate to the page that shows the detailed view as well as swipe back and forth to go sequentially through all the questions.

6.1.1 Extensible Application Markup Language

“XAML is the language behind the visual presentation of an application that you develop in Microsoft Expression Blend, just as HTML is the language behind the visual presentation of a Web page.” [14]

In the Windows phone application design, the designs built and controls included into an application either through coding or Microsoft Expression Blend have an Extensible Application Markup Language (XAML) behind them where all the definitions are saved.

It is an XML-based markup language developed by Microsoft. Every design made is saved as an xaml file and the Windows Presentation Foundation (WPF) parser reads the file to present it in *Design* view. [14]

6.1.2 Guidelines followed while designing the UI of CASS-Q

The UI for the application should contain controls and an easy navigation scheme that helps the user in participating in a survey without being distracted. The Windows phone UI design principle gives attention on simplicity; the UI should be presented being very simple creating smooth interaction with the user. Since the Metro design rules that Windows phones follow are considered in designing the UI for CASS-Q client, the principles of design that are taken into account by Metro are directly quoted from Microsoft [15].

Typography: Type is beautiful. Not only is it attractive to the eye, but it can also be functional. The right balance of weight and positioning can create a visual hierarchy. Additionally, well placed type can help lead you to more content. [15]

Motion is what brings the interface to life. Transitions are just as important as graphical design. By developing a consistent set of motions or animations, a system is created that provides context for usability, extra dimension and depth and improves the perceived performance of the whole interface. [15]

Content not Chrome is one of the more unique principles of Metro. By removing all notions of extra chrome in the UI, the content becomes the main focus. This is especially relevant due to the smaller screen size and gesture-based interactions. [15]

Honesty: Design explicitly for the form factor of a hand held device using touch, a high resolution screen and simplified and expedited forms of interaction. In other words, be “authentically digital”. [15]

6.2 Software architecture

Software application architecture is an important concept in the designing and development of software. It helps in defining a structured solution which is optimized for future expandability and reusability. It needs a series of decisions based on a wide range of factors and each of these decisions can have a considerable impact on the quality, performance, maintainability and overall success of the application. [16]

Philippe Kruchten, Grady Booch, Kurt Bittner, and Rich Reitman derived and refined a definition of architecture based on the work by Mary Shaw and David Garlan (1996)[16]. Their definition is as follows:

Software architecture encompasses the set of significant decisions about the organization of a software system including the selection of the structural elements and their interfaces by which the system is composed; behavior as specified in collaboration among those elements; composition of these structural and behavioral elements into larger subsystems; and an architectural style that guides this organization. Software architecture also involves functionality, usability, resilience, performance, reuse, comprehensibility, economic and technology constraints, tradeoffs and aesthetic concerns. [16]

There are three software design patterns that are widely being used based on the requirements of the application being developed. The architectures are the Model View Controller (MVC), Model View Presenter (MVP) and Model View View Model (MVVM).

In the MVC design architecture, the inputs are directed to the Controller first, not the view. After the input is interfaced with the Controller, some functionality could be started. There is a many-to-one relationship between the Controller and the View. A single Controller may select different Views to be rendered based on the executing operation. The MVP design architecture is very similar to MVC but there are a few distinct differences between them. The input begins with the View rather than the Presenter and there is a one-to-one mapping between the View and the associated Presenter. The Presenter reacts to events being triggered from the View. In MVVM, the input starts with the View, not the View Model. While the View holds reference to the View Model, the View Model has no information about the View. In the development of CASS-Q application, the data that is downloaded from CASS-Q server is not taken to the UI (View) directly; rather a controller is responsible to access data prior to any other activity and then renders the different types of Views based on the data retrieved. This need

in the development has made the MVC a convenient design pattern. In the following section, MVC design pattern is discussed in detail.

6.2.1 Model-View-Controller design pattern

Model-view-controller (MVC) is a software development pattern which isolates the business logic from the user interface. The architectural pattern separates the application into three main components named the model, controller and view. [16;17]

Models implement the logic for the application's data. Most of the time, if the project is involved with storing data in a database, then models operate on the data and finally update the database. But sometimes structures can be used to create the way data is stored in a model. In the latter case, the application will not have a physical model layer as it only reads data sets and updates views. [16;17]

Views are the components that display the UI elements of an application. Views get data from models of an application and data retrieved from the views can also update the data in a model. [16;17]

Controllers are independent components which create the communication between a view and a model. User interactions with an application, model update and rendering of a UI based on data from a model is handled by controllers. [16;17]

The MVC pattern helps in creating applications that separate the different aspects of an application (input logic, business logic, and UI logic), while providing a loose coupling between these elements. This in turn makes code reuse, scalability and future improvement easier. [18]

6.2.2 MVC concept of CASS-Q client

Taking easier code reuse, scalability and possible future improvement into account, the MVC software architecture was chosen and implemented in CASS-Q client. Data that is parsed from an XML file is stored in model objects. The controller classes retrieve data from model objects and make it ready for a view to get generated. Each time a survey is downloaded, a new UI is rendered and each time a user inputs answer, the model that stores user input is updated. As *figure 17* shows, both the model and view

communicate directly with the controller and the controller creates indirect communication between the model and view.

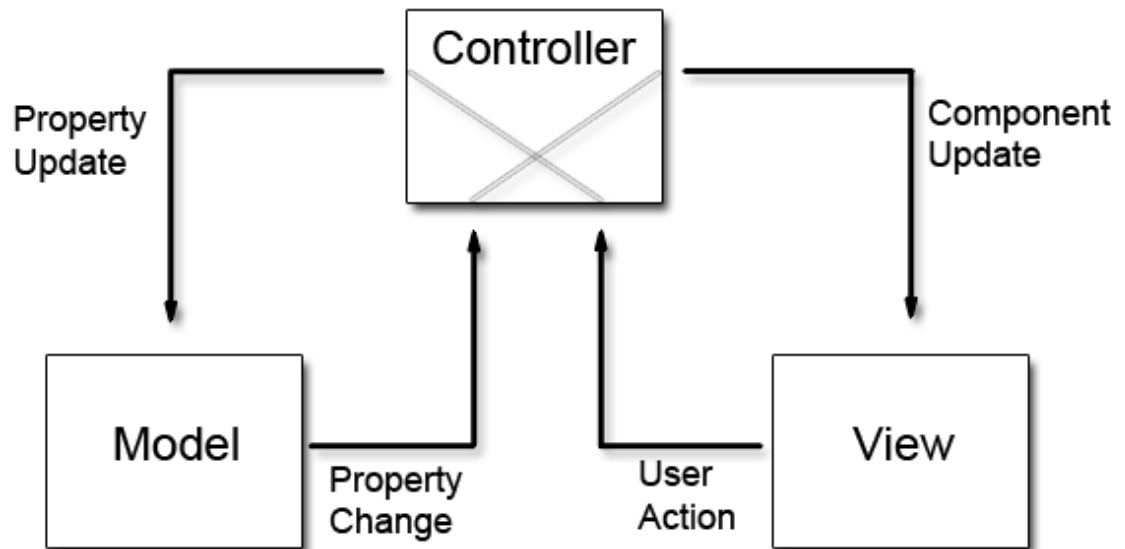


Figure 17. MVC software architecture. Reprinted from Code Project (2013)[19]

The model definitions of CASS-Q client contain properties which tell the attributes related with each type of objects. In *figure 18*, definitions of the major model are shown.

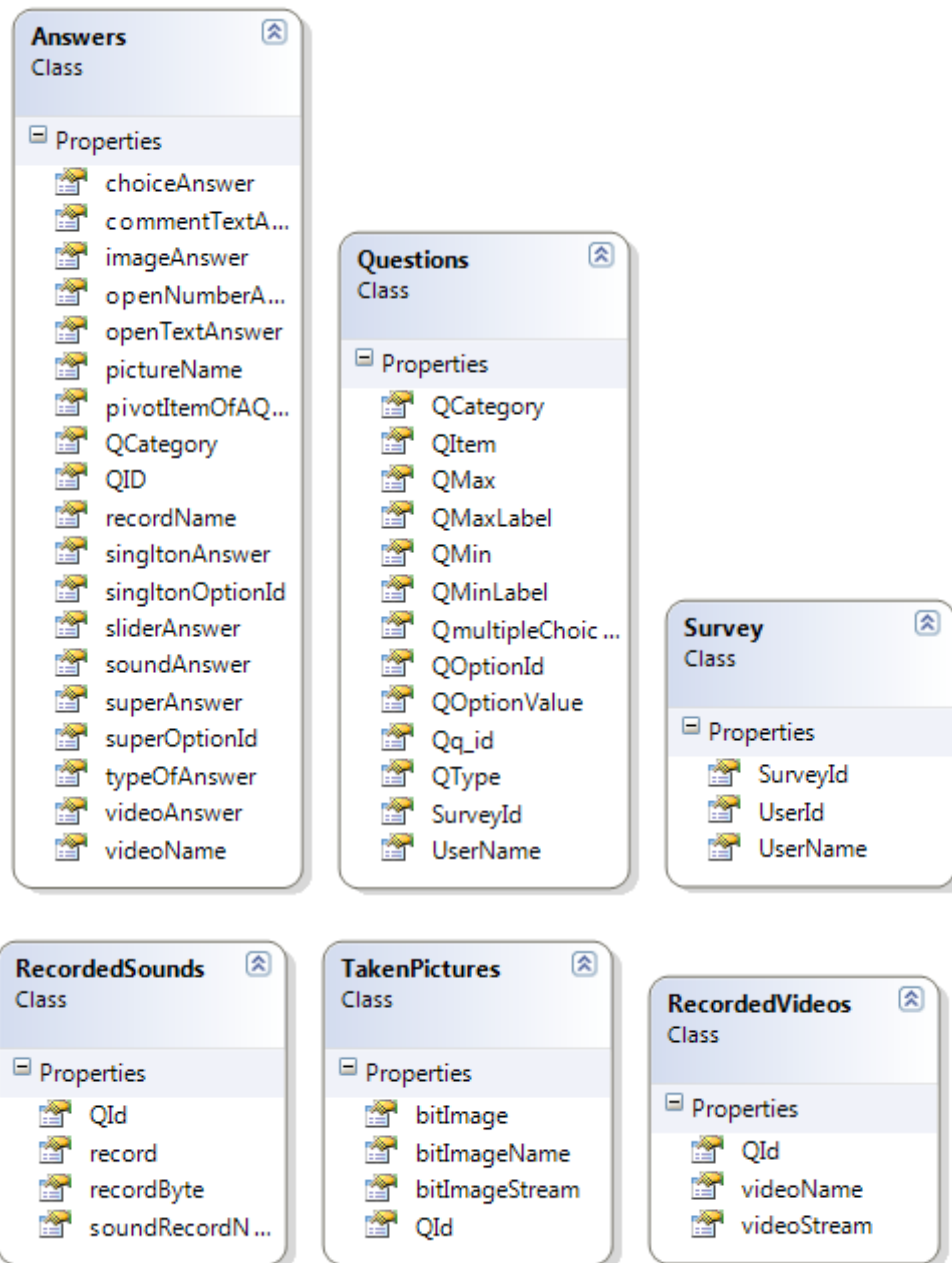


Figure 18. Models of CASS-Q client

As shown in *figure 18*, the Answer Model is the model that is involved with user answers which later helps in the submission of the whole survey data to the server. The Question Model on the hand is the model which defines the structure in which the questions downloaded from the server are temporarily stored in a phone. The Survey Model defines the structure of data to be temporarily stored on the phone concerning a specific survey.

7 Implementation of the CASS-Q client graphical design

A Graphical User Interface (GUI) is a human-computer interface which allows users to interact with an application giving a high level of abstraction to the commands carried out by the application. GUIs stand in sharp contrast to command line interface (GLIs) which is only gives a text-based interaction with an application.

In the following section, the different graphical elements included in the UI design of the CASS-Q client and the snippet code used are discussed.

7.1 Application banner

If an application should have a title that stays visible throughout its life cycle, then a proper banner which could give identity to the application to be designed. Below is a banner that is used in the CASS-Q client. Some looks are adopted from the UIs of the CASS-Q server application. *Figure 19* shows that banner used in CASS-Q client.



Figure 19. Banner of CASS-Q client

As mentioned in section 6.1.1, the UI definition in Windows phone is stored in the XAML file and below a segment which is used to add the banner to each page. *Listing 1* below illustrates how a banner is added using the XAML file of each view.

```
<StackPanel Margin="-13,-10,12,12">
    <Image Source="/cassq;component/Images/
        /cassq_banner.png" Margin="12,10,-331,17"
        Height="100" Width="auto"/>
</StackPanel>
```

Listing 1. Banner definition in XAML file

7.2 Launcher and application bar icons

First of all, every Windows phone application needs a launcher icon that represents the application on the home as well as on the applications screen. In order to create a unique and suitable icon, it is important to consider what the application is about. The CASS-Q client helps to analyze feelings and behaviors of certain people while learning or working. An icon that contains a head graphics and an orange surrounding was created taking the criteria mention prior into consideration. The background is made to be transparent so that it takes the background of the chosen theme in a Windows phone. *Figure 20* below shows the icon used.



Figure 20. CASS-Q logo/icon

After the application is launched, a user is requested to enter the token and log into the application by pressing an application bar icon. Then after logging in, based on the type of question in a detail view, the icons that are listed on the application bar are changed.

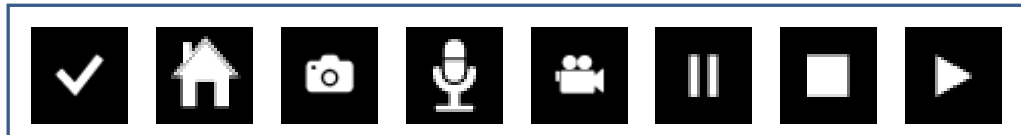


Figure 21. Application bar icons of CASS-Q client. Reprinted from Visual Studio 2010
All the application bar icons used in CASS-Q client are listed in *figure 21*.

7.3 Input controls

The controls in the CASS-Q client are added to the UI dynamically based on the type of questions included in a survey. The input controls are those in the UI which allow user to enter values as answer to a question. In this section, each control that is used in CASS-Q client application and mechanism used to include the element in the UI is discussed. Below are the input controls that are used and the code snippet related with each is given accordingly.

Text boxes are generated/declared and included in the detail view of an open text type of question. The text box allows user to input answer using the built-in keyboard of the phone. The text boxes in the application are created and included using C# code rather than by editing the XAML file of a corresponding view. This mechanism was followed as each UI is generated dynamically based on the download data. *Listing 2* illustrates how a text box is defined and added to CASS-Q UI.

```

TextBox[] openTextAnswer = new Text-
Box[gv.openTextTypeCount];
openTextAnswer[openTQ] = new TextBox();
openTextAnswer[openTQ].Focus();
stackPanelItemTQ[openTQ].Children.Add (openTex
tAnswer[openTQ]);

```

Listing 2. Definition and inclusion of a textbox

List picker is a new control that was introduced to the Windows platform which shows the selected item from a list and also allows the user to pick from a list if they want to change it. The open number question type can get input from the user using this control. As written in *listing 3*, a list picker array is defined based on the number of open number type questions.

```

ListPicker[] listPicker=new ListPicker[gv.opeCount];
listPicker[openNQ] = new ListPicker();
stackPanelItemON[openNQ].Children.Add(listPicker[openNQ]);

```

Listing 3. Definition and inclusion of a list picker

For slider type questions, the *slider* control is used to take user answer. A slider is a control that allows a user to slide over it to give some value to the slider. The filled part of the slider gets the color from the chosen theme of the phone. In *listing 4*, the code that is used to declare a slider and include it to the UI is shown.

```

Slider[] slideControl = new Slider[gv.sliderTypeCount];
slideControl[sliderQ] = new Slider();
stackPanelItemSQ[sliderQ].Children.Add(slideControl [slid-
erQ]);

```

Listing 4. Definition and inclusion of a list picker

The CASS-Q client application requires *radio buttons* for the single choice question type. The user is able to choose his/her answer by activating the desired radio button. Check boxes, however, are used for multiple choice questions and therefore enable the user to select more than one option.

```
RadioButton[] rb = new RadioButton[optionCount];
rb[i] = new RadioButton();
stackPanelItemSingletonC[singletonCQ].Children.Add(rb[i]);
```

Listing 5. Definition and inclusion of a list picker

Listing 5 shows how an array of radio buttons are declared, initiated and added to the UI stack panel. The same procedure is followed to define and include check boxes into the UI of CASS-Q client application.

7.4 Layouts

After the design decision was made, as mentioned in section 6.1, two layouts were suggested, a list view and detail view (master-detail) layouts. In order to implement this list-detail view relation, the *pivot item* control was selected. The pivot item in Windows phone is organized in a tabbed format. (Refer section 3.4.1)

The first pivot item contains all the questions from the survey as a clickable list. Though all the questions are listed, no controls are included to take user input from the first pivot item. The user is able to click on each question list in order to navigate to the detailed view of the clicked item list. The user can also swipe from one pivot item to another, back and forth, to traverse though all the detail views and the main question list. The screen shot is shown in *appendix 1*.

After a user logs in, a list view is appears displaying all the questions as clickable list items. The first pivot item is always reserved for these list items. Based on the number of questions, the number of extra pivot items to be generated is determined. For example, if there are twenty questions in a survey, the application generates twenty one pivot items. The first pivot item will contain all the questions as clickable links to the detail views of each question which will be generated and put in the rest twenty pivot items.

When a user clicks on an open text list from the list view, the pivot item with the detail view of the question is made visible. The detail view for open text question type contains a *TextBox* and *TextBlock*. The text block is the control used display the question itself and the text box is the element used for accepting user input. In *appendix 1*, the detail view for open text question type is shown.

The detail view that appears after a user clicks open number type contains a *TextBlock* and *ListPicker* controls. The *ListPicker* always attains the least number as its default value and when a user clicks it, all the numbers get visible and a user can select one. The *ListPicker* is an expandable control that can accommodate five number options without opening a bigger list in another window. The view is shown in *appendix 1*. A slider question detail view consists of a slider, start and end labels. The colour of the slider takes after the colour of the chosen theme of the phone. A Slider control is used added to the *TextBlock* used to display the question and labels. The picture included in *Appendix 1* show how the slider detail view looks.

The single choice question's detail view contains a *TextBlock* that displays the question itself and *RadioButtons* with labels specifying the option. The radio buttons allow user to select only a single option. The screen shoot depicting the view is shown in *appendix 1*. To let the user choose multiple answers for a question, the detailed view uses *CheckBox* and *TextBlock* controls. The text block is used to display the question and the check box is used to show the options available for choice. *Appendix 1* includes the screen shoot of the detail view of multiple choice type questions.

For questions that involve taking a picture, an *Image* control to the body and an icon is to the application bar are added to generate the detail view. The icon is used to launch the camera task so as to take a picture. The *Image* control is empty before taking any picture. After clicking the icon, the phone's camera interface is displayed and after accepting the taken picture, the view displays it in the *Image* control. The photo detail view is shown in *Appendix 1*.

In the detail view of a video type question, other than the *TextBlock* that displays the question, a video record, stop, play icons, *MediaElement* and *Rectangle* controls are added. The media element is used to play back a recorded video. The rectangle control is used to stream the live image that comes from the camera when a user starts re-

ording. Unless the record icon on the application bar is clicked, nothing will show up in the body of the detail view except the question itself. The view is given in *appendix 1*.

The detail view of a sound type question contains icons for recording, stopping and playing back a record are included into the application bar. Other than these, there are two *TextBlocks*, one for displaying the question and the other for telling the status of the recording. The view is shown in *appendix 1*.

8 Implementation of the CASS-Q business logic

In this section all the steps followed to implement the business logic of the CASS-Q Windows phone client application are discussed.

8.1 Making XML web request

The CASS-Q client has to make a web request to get survey data from CASS-Q server application. To make a web request, the *WebClient* is used and as its parameter, it takes the request URL that includes the token of a user.

```
//Function that downloads queries.
public void downloadQuery()
{
    //Init a web client that makes a download request
    client = new WebClient();
    request_uri = gv.uri;
    token = gv.token;
    string baseUrl = gv.settings["url"] as string + token;
    logIn.IsEnabled = false;
    try
    {
        client.OpenReadAsync(new Uri(baseUrl,
        UriKind.Absolute));
        //To be fired as survey is downloaded.
        client.DownloadProgressChanged += new DownloadProgressChangedEventHandler(client_DownloadProgressChanged);
        //To be fired when the download is complete.
        client.OpenReadCompleted += new OpenReadCompletedEventHandler(client_OpenReadCompleted);
    }
    catch (Exception Ex)
    {
        MessageBox.Show(Ex.Message);
    }
}
```

Listing 6. Code that makes web request to download survey

As shown in *listing 6*, with the *WebClient*, the *OpenReadAsync* method is used to give absolute URL. When the download starts, the *DownloadProgressChanged* and *Down-*

loadProgressChanged events are fired and operations that are needed to get performed are executed based on these events.

8.2 Parsing XML

the

In *listing 7*, part of the code that parses the XML file is shown. Information about the survey is needed so as to know the id of the survey. Knowing the id of a survey helps in relating user answers to the survey before creating and send XML file back to the CASS-Q server application. After saving important survey properties into a list, the questions are parsed and the data is saved in a list that is defined according to a model definition made for data storage.

```
var surveyDetail = (from surv in xDoc.Root.Attributes()
select new Survey
{
    UserName = surv.Document.Root.Attribute("username").Value,
    UserId = surv.Document.Root.Attribute("uid").Value,
    SurveyId = surv.Document.Root.Attribute("surveyId").Value,
}) .ToList();

foreach (var surveryInfo in surveyDetail)
{
    gv.UserName = surveryInfo.UserName;
    gv.UserId = surveryInfo.UserId;
    gv.SurveyId = surveryInfo.SurveyId;
    break;
}
//get each queation item
var ResearchQuestions = from item in
xDoc.Root.Descendants("item")
select new Questions()
{
    QItem =item.Value,
    QType=item.Attribute("type").Value,
    Qq_id=item.Attribute("q_id").Value,
    QMin=(string) item.Attribute("min"),
    QMax=(string) item.Attribute("max"),
    QMaxLabel=(string) item.Attribute("maxlabel"),
    QMinLabel=(string) item.Attribute("minlabel"),
    QmultipleChoices=(from child in item.Descendants("option")
select new QuestionOptions()
{
    QOptionValue=(string) child.Attribute("value"),
    QOptionId = (string) child.Attribute("o_id"),
}).ToList()
}).ToList();
```

Listing 7. Parsing data about survey and questions in the survey

The full code that performs the parsing and storage of data is given in *appendix 1*.

Added to parsing and storing survey data, the full code shown in *appendix 1*, sends the token to the server and displays any kind of information that the server could send.

8.3 Building UI

After downloaded XML is parsed, the question type is identified and the UI is developed accordingly. The controller takes data from the models defined and the views are generated dynamically. For different types of questions different types of controls are defined and included in the UI *stackpanel*. *loadUI(List<Questions> questionList)* is the function that takes care of UI generation. It accepts list of questions as its parameter. The list is a collection of objects which is defined according to the model definition for questions. *Listing 8* below shows the basic steps followed to generate dynamic UI for CASS-Q client.

```
public void loadUI(List<Questions> questionList)
{
    //Question type-1 (Open Text Question)
    //Creat array of Elements based on the count on the
    question type
    PivotItem[] pivotItemTQ = new Piv-
otItem[gv.openTextTypeCount];
    ListBox[] listBoxItemTQ = new
    ListBox[gv.openTextTypeCount];
    StackPanel[] stackPanelItemTQ = new StackPan-
el[gv.openTextTypeCount];
    TextBlock[] openTextQuestion = new Text-
Block[gv.openTextTypeCount];
    TextBox[] openTextAnswer = new Text-
Box[gv.openTextTypeCount];

    //Question type-2 (Open Number Question)
    //Creat array of Elements based on the count on the
    question type
    PivotItem[] pivotItemON = new Piv-
otItem[gv.openNumberTypeCount];
    ListBox[] listBoxItemON = new
    ListBox[gv.openNumberTypeCount];
    StackPanel[] stackPanelItemON = new StackPan-
el[gv.openNumberTypeCount];
    TextBlock[] openNumberQuestion = new Text-
Block[gv.openNumberTypeCount];
}
```



```

TextBox[] openNumberAnswer = new Text-
Box[gv.openNumberTypeCount];
ListPicker[] listPicker = new ListPick-
er[gv.openNumberTypeCount];
List<int>[] listOfNumbers = new
List<int>[gv.openNumberTypeCount];

foreach (var questionItem in gv.questionList)
{
    switch (questionItem.QType)
    {
        case "1":
        {
            openTextQuestion[openTQ] = new Text-
            Block();
            openTextAnswer[openTQ] = new TextBox();
            stackPanelItemTQ[openTQ] = new StackPanel();
            listBoxItemTQ[openTQ] = new ListBox();
            pivotItemTQ[openTQ] = new PivotItem();
            //add the answer control into the stackpanel
            stackPanelI-
            temTQ[openTQ].Children.Add(openTextAnswer[ope-
            nTQ]);
            listBox-
            ItemTQ[openTQ].Items.Add(stackPanelItemTQ[ope-
            nTQ]);
            pivotItemTQ[openTQ].Content = listBox
            ItemTQ[openTQ];
            cassqPivot.Items.Add(pivotItemTQ[openTQ]);

            if (openTQ < gv.openTextTypeCount)
                openTQ++;
            break;
        }
    }
}

```

Listing 8. Generating and including UI controls based on question type

By iterating though each question object, the function identifies the *question type* property and knows what type and how many controls to define and include in the application.

8.4 Accepting user answer

After all questions are displayed, the UI provides ways for a user to provide answers to the questions. When a user interacts and gives answers, the values of the controls are retrieved and saved in a structure temporarily.

```

/Add answer to the anwer list with
//the question id and type
this.cassAnswers.Add(new Answers()
{

```

```

        openNumberAnswer = listPicker[openNQ],
        QID = questionItem.Qq_id,
        typeOfAnswer = questionItem.QType,
        openTextAnswer = null,
    });

```

Listing 9. Fragment that accepts user input

A data model that defines the structure of an answer is created and user answers are added to an object of an answer model. In *listing 9* shows, the fragment that accepts user value from a list picker used for number type question.

8.5 Building XML and uploading user answer

After the user finishes answering, he/she presses the *submit* application bar menu item. The *Submit.cs* class checks for the completeness of answers and build XML from the answers to upload to the CASS-Q server.

```

using (XmlWriter writer = XmlWriter.Create(isoStorage, settings))
{
    writer.WriteAttributeString("answer", answerItem.recordName.ToString() + ".amr" as string);
}
if (answerItem.typeOfAnswer == "4") {
    foreach (var rb in answerItem.singletonAnswer) {
        if (Convert.ToBoolean(rb.IsChecked)) {
            singletonSelectedOption=rb.Content.ToString();
        }
    }
    writer.WriteAttributeString("answer", singletonSelectedOption);
}
if (answerItem.typeOfAnswer == "7") {
    writer.WriteAttributeString("answer", answerItem.pictureName.ToString() + ".jpg" as string);
}
if (answerItem.typeOfAnswer == "9") {
    writer.WriteAttributeString("answer", answerItem.sliderAnswer.Text.ToString());
}
if (answerItem.typeOfAnswer == "10") {
    foreach (var cb in answerItem.choiceAnswer) {
        if (Convert.ToBoolean(cb.IsChecked)) {
            selectedMultipleOptions+=cb.Content.ToString()+ ",";
        }
    }
}
}

```

```

        writer.WriteString("answer", selectedMultipleOptions);
    }
        writer.WriteEndElement();
        selectedMultipleOptions= "";
    }
    writer.WriteFullEndElement();
    writer.WriteEndDocument();
    writer.Flush();
    writer.Dispose();
}

```

Listing 10. Building XML

8.6 Settings

The settings window and the functionality of the window is done by the *Setting.cs* class of the project. A pop up window is generated and the user is able to save the token if there is a need.

```

const string URL =
"http://54.247.115.29/cass/MobileIO/XMLGen.php?uid=";
const string XMLUploadURL =
"http://54.247.115.29/cass/MobileIO/mobileanswer.php";
const string MediaUploadURL =
"http://54.247.115.29/cass/MobileIO/mobileMediaFiles.php";

IsolatedStorageSettings settings = IsolatedStorageSettings.ApplicationSettings;
public Setting(Action close) {

    if (gv.settings.Contains("token").ToString() == ""){
        gv.settings.Add("token", gv.token);
    }
    else
        token = gv.settings["token"] as string;
    if (!gv.settings.Contains("url")){
        url = URL;
        settings.Add("url", url);
    }
    else
        url = gv.settings["url"] as string;
    openPopUp();
    t = close;
}

```

Listing 11. Saving token

The request and upload URLs are displayed but are not editable. In *listing 11*, part of the code that stores token is given.

9 Testing and publishing

After development come testing the application based, on different criteria that were meant to be met when the project was started. Though there are different procedures that should and/or could be followed while testing software, usability testing was the one that was given attention and performed.

9.1 Usability testing

As the term suggests, usability means how much a software can be better than the purpose it was created for. It is a way to measure how the intended end users find it easy, moderate or hard to interact with and use the system keeping its purpose in mind. A usability test was performed because it can be modified to cover many other types of testing such as functional testing, system integration testing, unit testing and smoke testing. It helps to develop software with better quality and shorter learning curve for new users. [20;21]

As mentioned in *section 5.5*, the log in, navigation between questions, selection and answering a question, sending survey answers and saving tokens based on the need are the global test scenarios that were considered while performing the usability testing. Based on the uses made by end users at Helsinki University, the layout of the UI and navigation between different pages were not found to be difficult. Users were able to log in to the application easily. The users were able to select specific question to get the detail view and give answer (s) according, edit answers any time the need arises, send the answer to the server after finishing answering all the questions and save the token after getting to the *settings* page.

Even though the software developed was providing the purpose it was developed for the following are some of the constraints that it needs to address before being released as a final version in the Windows Phone Marketplace:

- It does not inform the end user while uploading the media files to the server.
- It only operates in portrait orientation.
- It allows only one question to be included in a subcategory of a super question.
- It allows recording only one video in a survey.

9.2 Submitting CASS-Q to Windows Phone Marketplace

After performing the usability testing that led to the discovery of the aforementioned constraints, the application was published on the Windows Phone Marketplace as a beta version. While submitting CASS-Q application to the marketplace certain requirements were needed to be fulfilled beforehand:

- A valid, active App Hub membership
- Final, release build of the CASS-Q application package (the .xap file)
- CASS-Q application artwork
- Category and subcategory selection, which was education for the CASS-Q application
- The price of the application, which was free for the CASS-Q application
- Various metadata about CASS-Q application including unique name, display title, version number, detailed description, keywords, website and copyright URLs, and support email address.

After making all the above requirements ready, the CASS-Q Windows phone application was released as a beta version and end users are now able to download and use the application.

10 Conclusion

The project was carried out to develop a Windows phone application for CASS-Query tool. The Contextual Activity Sampling System is a research methodology for the contextual tracking of activities. The main purpose of the project was to make the query tool available on a Windows platform as requested by Helsinki University. The goal of the project was to develop the application in a way that users could smoothly interact with it giving them full functionality, making the whole query tool more reliable.

Following the proper software development procedures, the CASS-Q application was realized. The realization was separated into two major parts. A suitable graphical user interface type was determined and subsequently implemented with an attractive design following the Windows phone design guidelines. A UI which is intuitive, attractive and easy to use was developed.

All mandatory functionalities were implemented and, as a result, a fully functional Windows phone client application was developed. The development of the client application would widen the platforms through which the CASS-Q tool could be used. The application is able to receive surveys, display questions to the user using easily usable UI controls, enable him/her answer the questions and send the completed survey back to the server. Thereby, it is possible to use all question types including photo, video and audio, the only limitation on the media files being the ability to record only one video for a survey.

In the future, the client application could have more features added to it in order to make it more functional and appealing to the end user. One of such features could be the ability for the application to let respondents answer questions even if the mobile phone was not online and later let the respondent upload the answers to the server. The other feature could be that the visualization about the respondent could be made available on the client application itself. This would give an immediate insight to the respondent about the feelings and thoughts of him/herself in certain situations and activities at a certain time.

References

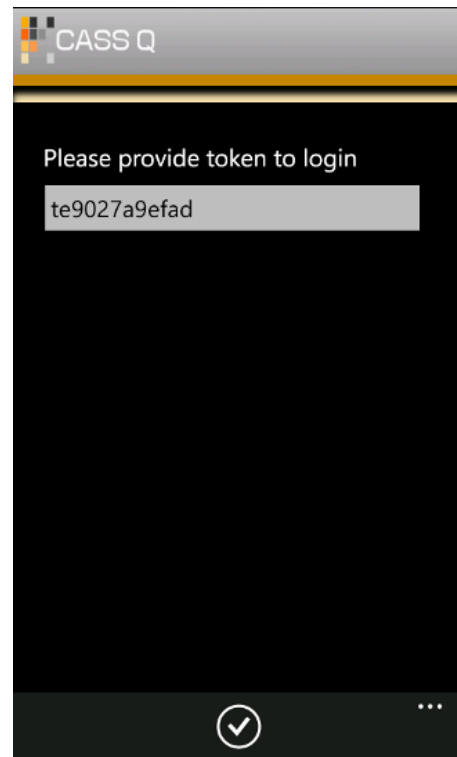
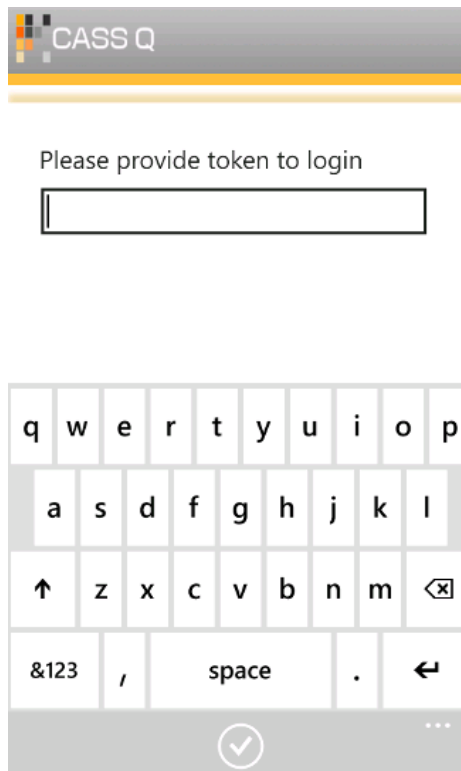
- 1 Muukkonen, H., Inkinen, M., Kosonen, K., Hakkarainen, K., Vesikivi, P., Lachmann, H., & Karlgren, K. (2009, June). Research on knowledge practices with the Contextual Activity Sampling System. In Proceedings of the 9th international conference on Computer supported collaborative learning-Volume 1 (pp. 385-394). International Society of the Learning Sciences.
(http://www.helsinki.fi/science/networkedlearning/texts/Muukkonen_etal_CSCL2009_CASS.pdf)
- 2 Microsoft. Application Platform Overview for Windows Phone [online]. Microsoft; March 22, 2012.
URL: <http://msdn.microsoft.com/enus/library/ff402531%28v=VS.92%29.aspx>
Accessed November 10, 2012.
- 3 WindowsPhoneGeek. Intro to WP7 Development: #2 choosing the right Framework [online]. WindowsPhoneGeek; January 20, 2011.
URL: <http://www.windowsphonegeek.com/tips/intro-to-wp7-development-2-choosing-the-right-framework>.
Accessed November 10, 2012.
- 4 Microsoft. Application Platform Overview for Windows Phone [online]. Microsoft; March 22, 2012.
URL: [http://msdn.microsoft.com/en-us/library/bb404713\(v=vs.95\).aspx](http://msdn.microsoft.com/en-us/library/bb404713(v=vs.95).aspx)
Accessed: November 18, 2012.
- 5 Microsoft. Serialization (C# and Visual Basic) [online]. Microsoft; March 22, 2012.
URL: <http://msdn.microsoft.com/en-us/library/ms233843.aspx>
Accessed November 18, 2012
- 6 Nielsen, J., & Molich, R. (1990, March). Heuristic evaluation of user interfaces. In Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people (pp. 249-256). ACM.
- 7 Jakob Nelson. 10 Usability Heuristics [online]. Nielsen Norman Group; January 1, 1995.
URL: <http://www.nngroup.com/articles/ten-usability-heuristics/>
Accessed February 16 2013.
- 8 Microsoft. User experience design guidelines for Windows Phone [online]. Microsoft; September 25, 2012.
URL: [http://msdn.microsoft.com/en-us/library/windowsphone/design/hh202915\(v=vs.92\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/design/hh202915(v=vs.92).aspx) Accessed November 22, 2012.

- 9 Microsoft. Getting applications from Marketplace [online]. Microsoft; 2012.
URL: <http://www.windowsphone.com/en-au/how-to/wp7/apps/find-and-buy-apps-in-marketplace>
Accessed November 22, 2012.
- 10 Dr. Dobb's Journal. Parsing XML Files in .NET Using C# [online]. South Tower, Suite 900, San Francisco; July 01, 2003.
URL: <http://www.drdoobbs.com/windows/parsing-xml-files-in-net-using-c/184416669?pgno=1>
Accessed December 1, 2012.
- 11 Microsoft. What is Namespace? [online]. United States; July 16, 2001
URL: <http://msdn.microsoft.com/en-us/magazine/cc302166.aspx>
Accessed December 15, 2012.
- 12 Bell. The sequence diagram [online]. IBM Corporation; February 16, 2004.
URL: <http://www.ibm.com/developerworks/rational/library/3101.html>
Accessed November 29, 2012.
- 13 Microsoft. Hardware Specifications for Windows Phone [online]. Microsoft; February 16, 2004.
URL: [http://msdn.microsoft.com/en-us/library/ff637514\(v=vs.92\).aspx](http://msdn.microsoft.com/en-us/library/ff637514(v=vs.92).aspx)
Accessed December 5, 2012.
- 14 Microsoft. What is XAML? [online]. Microsoft 2013.
URL: <http://msdn.microsoft.com/en-us/library/cc295302.aspx>
Accessed December 5, 2012.
- 15 Microsoft. Metro Design Language For Windows Phone [online]. Microsoft; 2013.
URL: <http://www.microsoft.com/design/toolbox/tutorials/windows-phone-7/metro/>
Accessed January 1, 2013.
- 16 Microsoft. What is software architecture? [online]. Microsoft 2013.
URL: <http://msdn.microsoft.com/en-us/library/ee658098.aspx>
Accessed January 4, 2013.
- 17 Microsoft. ASP.NET MVC Overview [online]. Microsoft; 2013.
URL: <http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview>
Accessed January 7, 2013.
- 18 Gielens. Model View Controller (MVC) Using C#, Delegates and Events in .NET [online]. Netherlands; May 12, 2002.
URL: <http://www.codeproject.com/Articles/2353/Model-View-Controller-MVC-Using-C-Delegates-and-Ev>
Accessed January 10, 2013.
- 19 Code project. Simple Example of MVC (Model View Controller) Design Pattern for Abstraction [online]. Canada; April 8, 2008.
URL: <http://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design>
Accessed February 3, 2013.

- 20 Buzzle. Software Testing-An introduction to usability testing [online].
URL: <http://www.buzzle.com/articles/software-testing-introduction-usability-testing.html>
Accessed Feb 18, 2013
- 21 Dumas, J. S., & Redish, J. C. (1999). *A practical guide to usability testing*. Intellect Limited.

Screen shots of CASS-Q client

Log in page



List view of all questions:



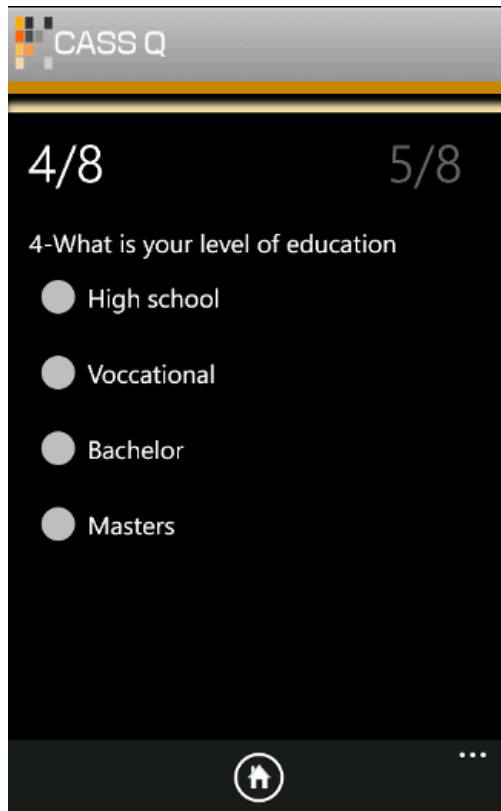
Detail view of an open text question type:



Detail view of an open number question type:



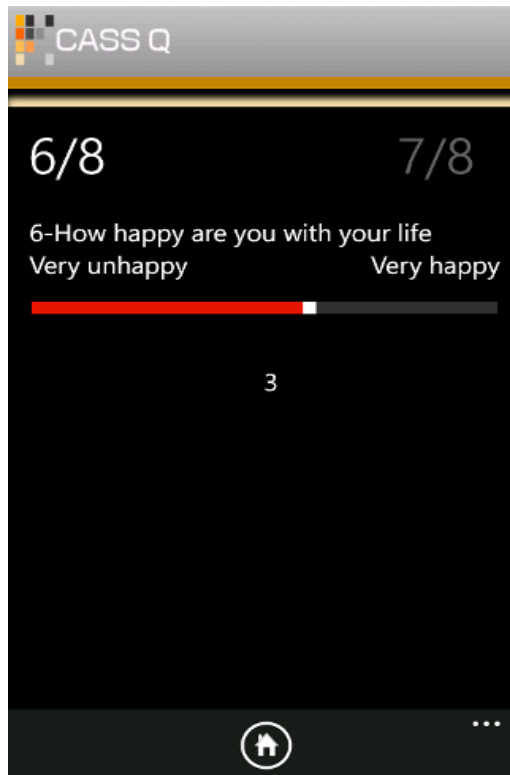
Detail view of a single choice question type:



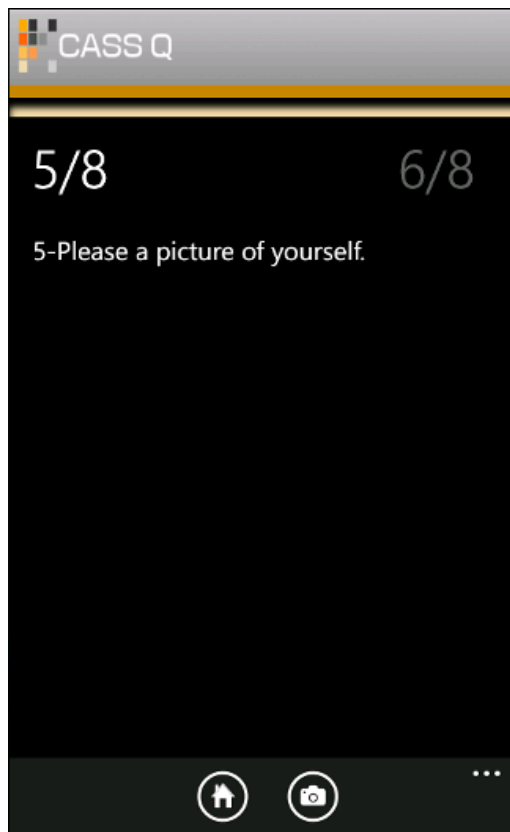
Detail view of a multiple choice question type:



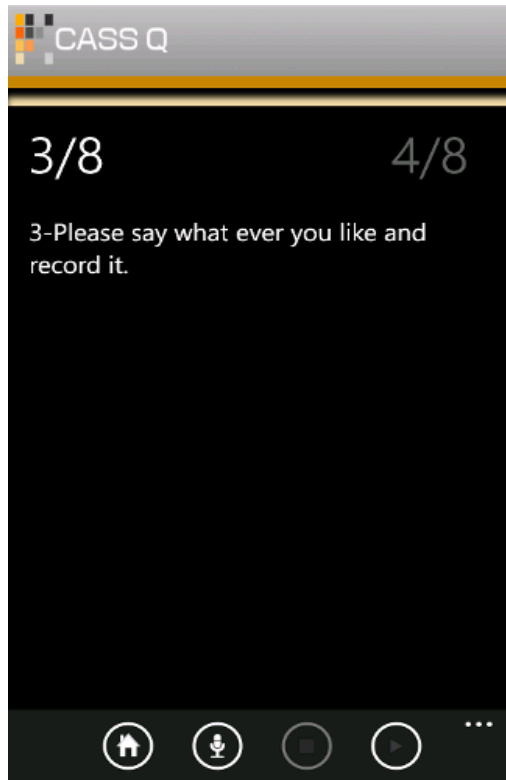
Detail view of a slider type question:



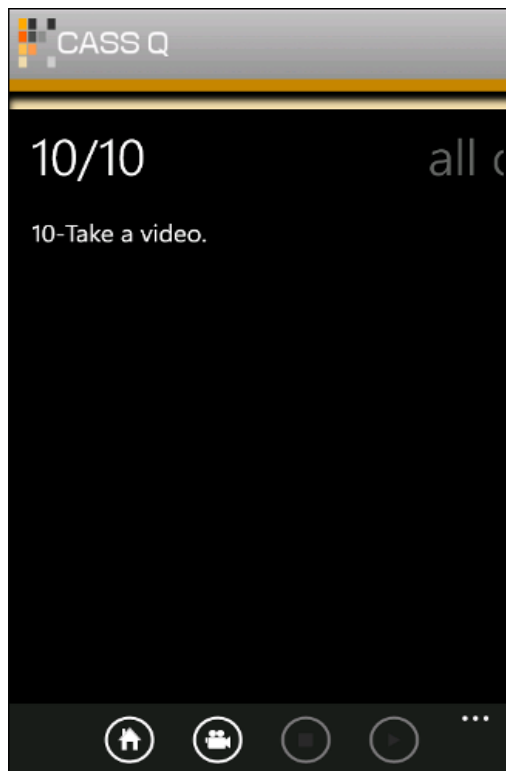
Detail view of a photo type question:



Detail view of a sound type question:



Detail view of a photo type question:



Setting view

The image shows a mobile application interface for 'CASS Q'. At the top left is the logo, which consists of a 3x3 grid of squares in various shades of gray and orange, followed by the text 'CASS Q'. Below the logo, there are four settings sections, each with a label and a text input field:

- Token**: The input field contains the text 'te9027a9efad'.
- Survey Request URL**: The input field contains the text 'http://54.247.115.29/cass/MobileIO/XMLGen.php?uid='.
- URL to send answer**: The input field contains the text 'http://54.247.115.29/cass/MobileIO/mobileanswer.php'.
- URL to upload media files**: The input field contains the text 'http://54.247.115.29/cass/MobileIO/mobileMediaFiles.php'.

At the bottom of the screen is a large white button with the text 'Save' in black.