ONLINE SURVEY SYSTEM
A WEB-BASED TOOL FOR CREATING AND ADMINISTERING STUDENT EVALUATIONS ONLINE

Fatima Marie Idowu
Online Survey System

A Web-Based Tool for Creating and Administering Student Evaluations Online

A Thesis in
Applied Computer Science

by
Fatima Marie Idowu

Submitted in Partial Fulfillment
of the Requirements
for the Degree of
Masters of Science

June 2006

© 2006 by Fatima Marie Idowu
I have submitted this thesis in partial fulfillment of the requirements for the degree of Master of Science.

8th June 2006

Date

Fatima Marie Idowu

We approve the thesis of Fatima Marie Idowu as presented here.

June 8, 2006

Date

Christopher Whitehead, Assistant Professor of Computer Science, Thesis Advisor

June 8, 2006

Date

Bhagyavati, Assistant Professor of Computer Science

June 8, 2006

Date

Eugen Ionascu, Associate Professor of Mathematics

June 11, 2006

Date

Paulina Kuforiji, Associate Professor of Education
Abstract

With the advancement in technology over the years, the administering of online surveys has expanded. In particular, universities are using the online medium to administer surveys to students, in order to evaluate faculty performances. The move of surveys to the online realm has meant a reduction in cost, time and efforts, of survey administrators, and the increase in use of technology within universities. With the use of online surveys the challenges of confidentiality, anonymity and response rates are as prominent as they are with paper-based surveys.

This study researched the use of online surveys in education; detailing systems currently used by many universities to facilitate the creation of an Online Survey System that would solve the latter challenges. The principal idea was to create a system that would provide the Columbus State University Computer Science department with a web-based tool for creating surveys and administering them online. The surveys will be created by faculty members; Survey Administrators, and administered to students; Survey Takers. With the system, faculty members are able to create surveys for classes taught, providing questions and responses deemed suitable. Once a survey is administered, students can access the survey by logging onto the system. Upon authentication students are able to complete surveys online.

Surveys are generated dynamically, depending on the survey criteria supplied by the survey administrator. The system stores survey criterion in a database, which is retrieved when generating a survey for a survey taker to complete.
The system developed focused on providing functionality that would increase the convenience of completing evaluation surveys and protect anonymity of students. The created system was successful in its aim to provide these features, also providing ease for faculty to create online surveys without having the extensive technical expertise that is required to do so.
Table of Contents

Abstract ........................................................................................................................................ iii
Table of Contents .......................................................................................................................... v
List of Figures ............................................................................................................................... viii
List of Tables ................................................................................................................................ ix
List of Figures ............................................................................................................................... ix
Acknowledgements ................................................................................................................... x

1. Introduction ............................................................................................................................... 1

2. Purpose of the System/Motivation/Related Work ................................................................... 3
   2.1 Online versus Paper Surveys ............................................................................................... 6
   2.2 Advantages and Disadvantages of Online Surveys ............................................................ 11
   2.3 Constructing Surveys ........................................................................................................... 12
   2.4 Issues with Response Rates ............................................................................................... 15
   2.5 Overview of related work / Comparable Survey Systems in Current Use ................. 17

3. Requirements Analysis ............................................................................................................. 22
   3.1 Introduction ......................................................................................................................... 22
      3.1.1 Scope of the system ...................................................................................................... 22
      3.1.2 Objectives and success criteria of the project ........................................................... 23
   3.2 Proposed System .................................................................................................................. 24
      3.2.1 Overview ..................................................................................................................... 24
      3.2.2 Functional Requirements ............................................................................................ 25
         3.2.2.1 User Authentication ............................................................................................... 25
         3.2.2.2 Querying Database for previous survey questions .............................................. 25
         3.2.2.3 Dynamic Survey Creation ................................................................................. 25
         3.2.2.4 Survey Start and Cut-Off Dates .......................................................................... 26
         3.2.2.5 Email Notification ................................................................................................. 26
         3.2.2.6 Generation of statistical survey result report ....................................................... 27
         3.2.2.7 Allocation of Survey Administrators and user groups ...................................... 27
         3.2.2.8 Viewing completed surveys ............................................................................... 27
         3.2.2.9 Other aspects of the system ................................................................................ 27
      3.2.3 Justification of functional requirements ......................................................................... 28
         3.2.3.1 User Authentication ............................................................................................... 28
         3.2.3.2 Querying Database for previous survey questions .............................................. 28
         3.2.3.3 Dynamic Survey Creation ................................................................................. 28
         3.2.3.4 Email Notification ................................................................................................. 28
         3.2.3.5 Generation of statistical survey result report ....................................................... 29
         3.2.3.6 Protection of anonymity ....................................................................................... 29
      3.2.4 Non-functional requirements ......................................................................................... 29
         3.2.4.1 User interface and human factors ......................................................................... 29
         3.2.4.2 Hardware and software considerations ............................................................... 30
         3.2.4.3 Performance characteristics ............................................................................... 30
         3.2.4.4 Error Handling and Extreme Conditions ............................................................. 30
3.2.4.5 System Modifications ................................................................. 31
3.2.4.6 Operating Environment ............................................................... 31
3.2.4.7 Security Requirements ................................................................. 31
3.3 System models ................................................................................. 31
  3.3.1 Scenarios ..................................................................................... 31
  3.3.2 Use Case Model ........................................................................... 34
  3.3.3 Use Case model Descriptions ......................................................... 36
    3.3.3.1 The ‘Log On’ Use Case .............................................................. 36
    3.3.3.2 The ‘Create Survey’ Use Case .................................................. 37
    3.3.3.3 The ‘View Survey’ Use Case ...................................................... 39
    3.3.3.4 The ‘Notify Survey Takers’ Use Case ......................................... 40
    3.3.3.5 The ‘Complete Survey’ Use Case .............................................. 41
    3.3.3.6 The ‘View Completed Survey’ Use Case ..................................... 42
  3.3.4 Object Models .............................................................................. 43
    3.3.4.1 Class Diagram ......................................................................... 44
3.5 Functional Requirements Cross-referenced and prioritized ............... 46
Summary ............................................................................................... 48
4. Design and Implementation ................................................................. 49
  4.1 Introduction ...................................................................................... 49
  4.2 Design decisions .............................................................................. 49
  4.3 Proposed software architecture ......................................................... 52
    Overview ............................................................................................ 52
  4.4 Subsystem decomposition ................................................................. 53
  4.5 Subsystem services .......................................................................... 55
  4.6 Survey Application Subsystem ........................................................... 56
    4.6.1 User Interface ............................................................................. 58
      4.6.1.1 Login .................................................................................. 58
      4.6.1.2 Default ............................................................................... 58
      4.6.1.3 AdminMain ........................................................................ 59
      4.6.1.4 Create .............................................................................. 61
      4.6.1.5 Answer2 ............................................................................ 63
      4.6.1.6 Survey ............................................................................... 66
      4.6.1.7 SendMail ........................................................................... 68
      4.6.1.8 Message Sent .................................................................... 70
      4.6.1.9 Results .............................................................................. 71
      4.6.1.10 Main ............................................................................... 72
      4.6.1.11 Page ................................................................................. 74
      4.6.1.12 Survey Error ..................................................................... 76
  4.7 The Survey Application Object Subsystem ......................................... 77
  4.8 The Database Subsystem ................................................................. 79
Summary ............................................................................................... 81
5. System Evaluation ................................................................................ 82
  5.1 Functionality Testing ....................................................................... 82
    5.1.1 Choosing Test Cases ................................................................... 82
      5.1.1.1 The ‘Log On’ Test Case ...................................................... 82
List of Figures

Figure 1: The use case model for the propose system ........................................... 36
Figure 2: Abstract object model for proposed system ............................................ 44
Figure 3: Login Interface of Proposed system ....................................................... 45
Figure 4: Create Survey Interface of Proposed system .......................................... 46
Figure 5: Three Tier architecture for web applications .......................................... 53
Figure 6: The subsystems for the Online Survey System ....................................... 54
Figure 7: Object Model (class diagram) of the Survey Application subsystem ......... 57
Figure 8: System default page .............................................................................. 59
Figure 9: Survey Administrator Main page ......................................................... 61
Figure 10: Add New Question panel on the create page ....................................... 63
Figure 11: Configuration of survey question responses ......................................... 65
Figure 12: Creating a new answer for a survey question ........................................ 66
Figure 13: Survey created by survey administrator on survey page ....................... 67
Figure 14: Email notification on sendmail page .................................................... 69
Figure 15: Reminders notification on sendmail page ............................................. 70
Figure 16: Email confirmation ............................................................................. 71
Figure 17: Survey response rate graph ................................................................. 72
Figure 18: Email notification received by survey taker ......................................... 73
Figure 19: Survey Taker Main page ..................................................................... 74
Figure 20: Survey completion page ..................................................................... 75
Figure 21: Previously completed survey .............................................................. 76
Figure 22: Page access error page ....................................................................... 77
Figure 23: Object Model (class diagram) of the Survey Application Object subsystem .......... 78
Figure 24: Entity Relationship diagram of the system database ........................... 80
List of Tables

Table 1: Actors for the system and their roles ................................................................. 35
Table 2: Initial objects Descriptions for the Online Survey System .................................. 44
Table 3: Functional requirements cross-referenced with use cases .................................... 48
Acknowledgements

Firstly I would like to give glory and praise to my Heavenly Father for giving me the peace, wisdom and strength I needed to complete this thesis. Thank you Lord.

I want to express my deepest gratitude to Professor Whitehead for being so patient with me and always being available when I needed help throughout the course of this project. For your guidance and support, I am truly grateful.

I also want to thank the committee members, Bhagyavati, Eugen Ionascu, and Paulina Kuforiji, for their comments and suggestion throughout the process of documenting this thesis and for their approval of my work.

Finally I want to thank my parents and sisters for their love, support and prayers for me during the course of this project. Thank you to Andrew Smith for guidance, suggestions and constructive criticism of my work and Amir Lynton for encouraging and constantly assuring me that I was able to complete this project.

God Bless You All and Thank You.
1. Introduction

For many decades, surveys have been used for research and as a means of obtaining feedback from consumers. In fact, surveys are probably the most commonly used research method world-wide [Pfleeger & Kitchenham, 2001]. Surveys can be administered in two different forms: supervised and unsupervised. The means by which a survey is administered is dependent upon the administrator's objectives and the resources available. Supervised surveys are those where a survey researcher is assigned a respondent, or survey taker; for example, telephone surveys require a researcher to ask the respondent a series of questions and then to record the answers. Unsupervised surveys take the form of automated voice telephone calls or mailed questionnaires.

In light of the advancement in technology over the years, the medium used to administer surveys has expanded to the online realm. Companies have been able to administer their surveys online in order to obtain feedback on their products and services. Due to the nature of the Internet, respondents' answers are recorded, totaled, and ready for analysis immediately, thus eliminating the need for the tallying of surveys' results by hand; thereby saving time and money. In the past few years, this trend has moved into the education system.

It is common practice among universities to give end-of-unit surveys in order to gain feedback on the units offered at the institute. Over the years, the use of communication has grown extensively in university teaching and learning. Many universities have expanded to online curriculum and with this expansion have adopted online surveys as a method of
collecting student feedback. However, the use of online surveys for feedback on the university’s curriculum and performance of its professors is not limited to those taking online classes. Universities are adopting online surveys as a mechanism to replace “pen and paper” surveys across the board; online and face-to-face classes alike. Research done at Murdoch University found that the growth in the use of online surveys developed from the desire to align the surveys with the use of technology for classes, “to increase access to external students, and to improve the efficiency of the survey process.” [Cummings & Ballantyne, 2000]

While online surveys are used in numerous universities all around the world, there is little research literature investigating their use in education [Cummings & Ballantyne, 2000]. Few researchers have investigated the use and creation of online surveys for the educational field [Cummings & Ballantyne, 2000; Pfleeger & Kitchenham, 2002b; Ha & Mars, 1998].

The University of Leicester has a research project entitled ‘Exploring Online Research Methods.’ The research resulted in the development of a website that aims to provide an “online resource which provides training for researchers who are interested in using online research methods such as online questionnaires and online interviews.” The website targets researchers and postgraduates in the higher education, and “researchers working for other organizations such as those involved with public policy and market research,” and is a great resource providing online methodologies to those wishing to explore the online research realm.
2. Purpose of the System/Motivation/Related Work

The common practice at Columbus State University (CSU) is to administer paper-based end of unit surveys. This allows departments to offer students the opportunity to evaluate their classes and the faculty members who conducted these classes on the basis of their teaching skills and the way the class or module was delivered. The main objectives of these evaluations are to assess the instructor's performance, gain insight into student attitudes about course content and assignments, and student satisfaction levels with quizzes, exams, and the course in general. Evaluations are also used to assess changes in instructional practices or new courses; specifically when this practice is used or the course is offered as part of an experiment or a trial. The results of these surveys are generally used to determine whether a class will be offered again, if the class was offered on a trial basis, or whether the professor's teaching style needs to be adapted in anyway.

The Department of Computer Science at CSU currently uses both paper-based and online evaluations. Some faculty members prefer the use of paper-based surveys while others prefer their online counterparts. However, these ultimately contain the same information. The Department of Computer Science has a designated individual who creates the online surveys for the department and faculty are then responsible for pointing students to the appropriate Internet address to take the survey.

The online surveys currently administered by the Department of Computer Science are static in nature, meaning that the questions are predefined and each survey has the same questions on them, these questions are the same as those used in the paper-based surveys; there is no opportunity provided for faculty to pose their own questions regarding classes that they
teach. This is somewhat unfortunate as a faculty member teaching a class will more than likely have questions specific to the class that he/she would like to ask the students.

The intent of this project, therefore, is to produce a system that is aimed at providing CSU Computer Science faculty members with a tool which can be used to develop online evaluation surveys for the courses taught and to administer these surveys to students when they deem fit; most likely near the end of the semester or half semester, for those classes offered for only half a semester. The system will allow faculty to supply questions that they feel are relevant to their course rather than administering a static survey which is used college wide. The system will be web based allowing faculty to create surveys when they wish and to make them available to students. This system will be beneficial for both students and faculty, as students will have the opportunity to take surveys in the convenience of their own time. It is anticipated that this convenience will allow students to think about their responses; and thus increase the quality of their responses. Feedback from students revealed that completing paper-based surveys in a class room setting leads to the selection responses that would not normally be chosen. The reason for this is that students feel rushed to complete surveys in the time given. Whilst this notion is plausible, it has not been validated by research.

The system currently in place has no way of restricting students from completing a class survey more than once or even restricting students who did not take a particular class to have access to complete the survey. The proposed system will utilize user authentication in order to restrict users from completing a survey more that once and makes surveys available only to those students specified by faculty.
There are three groups of individuals who are involved in the process of student evaluations [Gaide, 2005]. These are system administrators, survey administrators and students. A survey administrator is responsible for the creation of online surveys. In the proposed system the faculty members will take on the role of survey administrators and students will take the role of survey takers. The system will have a system administrator who will be responsible for maintaining the system.

The success of student evaluations online is wholly dependent upon the faculty’s support. It is the faculty’s responsibility to make students aware of the surveys they need to complete. Faculty are, however, in a position where they can either undermine the survey by lack of attention and despondent comments or promote the survey through supportive comments, reminders and providing feedback on how past survey results were used to improve the unit [Cummings & Ballantyne, 2000]. It is, therefore, anticipated that the system will encourage faculty to promote surveys as it will be surveys that they created themselves and feedback from those surveys will be more directed towards information they require about the particular class rather than simply generic feedback about the teaching process.

Response rate is a major issue when administering surveys. Any reliable survey system should measure and report its response rate and responses. Using the system presented in this thesis, faculty will be able to view statistical results of completed surveys online, allowing them to graphically see responses to the surveys instantly instead of having to wait for all surveys to be taken and then tallied; as is the case with paper-based surveys.
The proposed system therefore aims to improve the process of administering surveys by using the Internet as a medium and making the process more convenient for faculty and students alike.

2.1 Online versus Paper Surveys

Two common types of surveys are paper-based and telephone surveys. However, with the recent move made by higher education institutions to make their surveys available online, the question arises as to the effectiveness of online surveys versus paper-based surveys. It has been argued that comparison of responses between online and paper surveys shows no difference [Kelly & Marsh, 1999].

There are various advantages when using an online surveying system for those involved in the process. The common advantage to all involved is the flexibility and the “potential to improve the efficiency of the survey process.” Alternatively, low response rates and “increased reliance on access to technology” are identified as significant disadvantages [Cummings & Ballantyne, 2000].

The creation of surveys and the surveying process highlights many challenges. These challenges range from confidentiality and anonymity to response rates. These challenges are common, however, to both paper-based and online surveys.

The observed challenges that face an online surveying system are those of computer access and literacy, security and confidentiality and response abuse, such as multiple submissions.
In asking the questions whether student feedback should be moved completely online, four essential criteria were found to be influential [Cummings & Ballantyne, 2000]. These are “support of staff members involved, level of student access to information technology, the lowest level of computer literacy in the student target group and the level of student acceptance of undertaking tasks online.” If surveys are to be fully administered online, they must be promoted by staff or students will be unaware of their availability. Staff members may be reluctant to promote evaluation surveys because they do not want to change their teaching style or be critiqued by students. This also holds true with paper-based surveys.

The access to hardware is a limitation to consider when thinking about online surveys. The level of access that students across a campus or class have may vary considerably. The physical access to appropriate hardware and software for students is not the only factor to be considered when talking about access to information technology, but also access where and when it is convenient for the student. With regard to access to hardware, class time could be provided where students have access to the appropriate hardware and software needed to complete a survey. Paper-based surveys are usually administered during class time. Nevertheless, with both online and paper surveys, the issue of convenience is still at hand. Are a couple of minutes before the end of class necessarily a convenient time for a student to complete a survey? As a student myself, and this is true for many students that this issue has been discussed with, when asked to complete a course evaluation in class, one would prefer to complete it in his or her own time, as the student tends to feel rushed to complete the survey within the few minutes at the end of a class or within the time frame given. The
given responses therefore may not necessarily be the same as those students would have
given if given the opportunity to complete surveys in their own time. This is especially true
when one is required to give comments, as a student will not have time to ponder the
response they want to give.

The level of computer literacy among students and also among staff is another factor when
considering transferring surveys online. Unless their particular area of expertise affords
them to deal with information technology regularly, staff members, especially the ones less
familiar or comfortable with using technology, will have more problems administering online
surveys. The level of competence with computers is not necessarily evenly distributed among
the student population and also the staff population. However, many high school graduates
are fairly familiar with computer technology, as nowadays it is a requirement for high school
students to take computer literacy classes. There are however, some parts of the world where
it may not be necessary depending on the field the student wants to embark upon, i.e.,
mature students also may lack experience or confidence with computers

If CSU is to administer all its surveys online, faculty need to be comfortable with using an
online surveying system, appropriate access to technology needs to be available to all
students asked to complete surveys. Students need a reasonable level of competency to
complete the survey and a willingness to do so. With regards to staff comfort in
administering online surveys, it is not necessary that all staff members become survey
administrators. Those comfortable with the task can be given the responsibility of
administering surveys for themselves and other staff members. With reference to student
access to technology, most university campuses are equipped with computers available for student use; the only issue here is the convenience to the student.

As the primary focus for this system is the Department of Computer Science, the issue of computer literacy among staff members should not be an issue. With students there is a chance that computer literacy may be low, but as they are completing a computer science degree this level will increase as they progress with their degree and there are tutors who are available to help with any computer literacy issues.

When considering taking surveys online, there must be a means of informing the respondents, students in this case, that they are required to take the survey. If a student has the liberty to complete a survey in their own time, a mechanism should be in place to inform the student of the requirement to complete the survey; this should be something other than the instruction from the teacher. With existing systems, students are usually notified via school email.

An effective survey system for student feedback should be password protected; there should be some way of authenticating users to ensure that the intended survey taker is the one actually taking the survey. With paper surveys, particularly those that are mailed and in the case of students, those which students are given the opportunity to take away with them and return at a later date, it is somewhat difficult to ensure that the correct person has completed the survey. A student might have a friend or a sibling complete a survey for them. With the means of authentication, a majority of students are aware of security issues and will be more
reluctant to share their login credentials with others. A password system will therefore allow the intended survey audience to respond and no one else.

With paper-based surveys administered in class, instructors usually tell students to drop the completed surveys off when the students are leaving or return the completed surveys to the faculty member's mailbox at a later date. These surveys are somewhat anonymous, so there is no way to remind those who have yet to submit them to do so. With an online system such as the one proposed, there will be a means of identifying those students who have yet to complete survey for the purpose of reminding them to do so. The issue of confidentiality and anonymity will arise however if survey administrators have access to the identities of students who have yet to or did not complete their surveys. For this reason the proposed system will protect anonymity by having a mechanism to track those who have not completed surveys solely for the purpose of sending reminders. This information will be encapsulated so that the instructor or persons administering the survey have no information as to who reminders are sent to. The system will also have a mechanism of ensuring that surveys are only responded to once by a particular respondent.
2.2 Advantages and Disadvantages of Online Surveys

Advantages of online surveys include access to individuals in distant locations; this is beneficial for those taking online classes. Automated data collection is another advantage which as a result reduces researcher time and effort. Online surveys save time by allowing large volumes of data to be collected for the given survey continuously and imported into statistical tools and databases, increasing the speed and accuracy of analysis. In the case of the Department of Computer Science, the resource which is normally used to collect and analyze the data for surveys will no longer be required, as survey responses will be automatically stored on a database system. As a result, there will be a reduction in the cost of administering survey, “from less staff time required to handle forms and enter data” [Cummings & Ballantyne, 2000] to saving cost of printed forms. Data can also be automatically validated for online surveys; that is, the system can return error messages requesting the correct format of data entry, resulting in low data entry errors. In addition, online surveys eliminate handwriting interpretation problems.

Disadvantages of online surveys include uncertainties over the validity of the data and sampling issues. Here validity refers to the accuracy of the specific conclusions and inferences drawn from non-experimental data [Gunn]. For online researchers, sampling is an issue as there is no access to a central registry or database where an accurate sampling frame can be gathered, neither is there any way of discerning how many users are logging on from a particular machine. For the proposed system, this is not an issue of concern, as the sample will be the students registered for a particular class or those that meet the criteria set out by the survey administrator. Another disadvantage is a concern surrounding the design, implementation and evaluation of an online survey [Wright, 2005]. Administering an online
survey requires a certain level of technical expertise. In addition, the time taken to prepare an online questionnaire can be substantial and may outweigh some of the time savings noted in the advantages. With the proposed system however, a substantial part of the designing will be automated. The only concern for the survey administrator will be to provide the survey questions and administer it to students by sending them email notifications and reminders, which will be done through the system.

Another disadvantage to consider is that online surveys may need to be shorter than paper-based surveys. Response rates for online surveys drop after 10-15 questions and are directly and negatively correlated with questionnaire length [Madge, 2006].

Technical issues can also occur with online surveys. A server or computer can crash. There are technical variances in computers, monitors, browsers and internet connections which may have design implications.

2.3 Constructing Surveys

The construction of a survey goes far beyond the development of a questionnaire and asking the intended audience to complete the survey. A survey should be seen as more than just an instrument for gathering information. Surveys should be viewed as comprehensive systems for “collecting information to describe, compare or explain knowledge, attitudes and behavior” [Pfleeger & Kitchenham, 2001].

The “survey instrument” is part of a large survey process defined by the following ten activities [Pfleeger & Kitchenham, 2001]:

1. Setting specific, measurable objectives
2. Planning and scheduling the survey
3. Ensuring that appropriate resources are available
4. Designing the survey
5. Preparing the data collection
6. Validating the instrument
7. Selecting participants
8. Administering and scoring the survey instrument
9. Analyzing the data retrieved
10. Reporting the results, statistical analysis and inference of survey results

It is necessary that a survey be designed to provide the most effective means of obtaining information needed to address the objectives of the survey [Kitchenham and Pfleeger, 2002b]. For a survey to provide the most effective means, it should be designed in a way that it will not be swayed by a particular faction, aspect or opinion. The survey should make sense in the context of the population, and the administration and analysis should be within the resources allocated to the survey.

Those conducting surveys often have some idea of what they are seeking. As a result, the way they build the survey instrument can inadvertently reveal their biases [Kitchenham and Pfleeger, 2002a]. Replies to survey questions can be influenced by

1. The way the question is asked
2. The number of questions asked
3. The range and type of response categories
4. The instructions to respondents
5. The order in which the questions are asked.

6. The language used in the questions.

To avoid bias, survey construction must be done in a way that questions are neutral, the use of words should not influence the respondent’s thoughts, enough questions should be asked to adequately cover the survey topic, attention should be paid to the order of questions (so that answers to one do not influence responses to the next), provision should be made for exhaustive, unbiased, mutually exclusive response categories, and instructions should be clear and unbiased.

When constructing a survey, care must be taken as to how the questions are formulated and structured. Questions must be formulated in a way that respondents can answer easily and accurately. Questions should, therefore, be worded so that a respondent can see relationships between the intention of the question and survey objectives [Kitchenham & Pfleeger, 2002b]. That is, the purpose of the question should be clear, or the question is likely to go unanswered or ‘thought-less’ responses provided. It is important to keep in mind that the number of questions that can be realistically asked in a survey depends on the amount of time respondents are willing to spend completing it [Kitchenham & Pfleeger, 2002c].

The time it takes to complete a survey can be dramatically reduced by having standardized answer formats, for example “strongly agree, agree, disagree, strongly disagree” [Kitchenham & Pfleeger, 2002c]. Standardized answer formats save time, as the respondent can anticipate that the same choices are available for each question and do not have to take the time to
read new answer choices for every question within the survey. Questions in a survey can be either open or closed. **Open questions** avoid imposing restrictions on the respondent, ultimately leaving room for misinterpretation and provision of irrelevant or confusing answers. Open questions are difficult to code and analyze [Kitchenham & Pfleeger, 2002c]. **Closed questions** restrict responses but are easier to analyze. However, on the subject of standardized responses for closed questions, each question may require responses that may not be necessarily aligned with the standard answer set, whatever it may be.

Questions can usually be grouped into topics where each topic addresses specific objectives for the survey. It is important not to have too many questions on a survey as this can make the respondent lose interest [Kitchenham & Pfleeger, 2002c]. Kitchenham and Pfleeger further suggest identifying a topic that is addressed by many questions and removing some of the less vital ones as a way of reducing question size. However, a balance should be maintained between what one wants to achieve with the survey and the willingness of the respondent to provide the required information.

### 2.4 Issues with Response Rates

Some researchers oppose the view that there is little difference between the response rates of online and paper-based surveys [Nulty, 2000]. Response rate is a major issue when administering surveys. Any reliable survey should measure and report its response rate. The validity of a survey is severely compromised if there is a significant level of non-response [Kitchenham & Pfleeger, 2002b].
[Cummings and Ballantyne, 2000] found that when they established an online system at Murdoch University, the response rate from students was lower than that in paper-based surveys administered at the university. Despite strategies put in place to make the system effective, overall response rates for each semester were 30% in comparison with 65% achieved for paper forms.

Many factors contribute to low response rates. For example, research suggests that people do not like to participate in surveys unless they feel it is beneficial to them in some way [Kitchenham & Pfleeger, 2001]. Incentives are usually introduced to increase response rates. Cummings and Ballantyne used a “cash prize draw as an incentive for students who completed surveys for all their units online.” This improved the response rate from 30% to 54%.

Due to the “nature of online survey process, response rates need particular attention” [Cummings & Ballantyne, 2000]. Cummings and Ballantyne demonstrate that there are a “number of useful strategies to improve response rates.” These include but are not limited to encouraging staff to promote the survey, discussing feedback and any consequential course changes with students, and offering incentives, particularly cash. These methods have been shown to contribute to improved response rates.

The following steps were highlighted by Kitchenham and Pfleeger [2002b] to improve response rates:

1. Ensure that respondents are able to answer the survey questions (questions should be unambiguous and answerable)
2. Ensure that questions cannot be perceived by respondents as intrusive or impertinent

3. Ensure that respondents are motivated to answer the questions (show clearly that there is some benefit to them in answering the survey questions)

Kitchenham and Pfleeger [2001] also noted that people are more motivated to complete surveys if they can see that the results are going to be useful to them within the education environment, students need to be motivated that it is worthwhile to complete the evaluation for a class they have taken. This can be done by informing them of changes or things that have been put in place as a result of previous survey responses.

2.5 Overview of related work / Comparable Survey Systems in Current Use

There are few systems already in existence implementing the same basic goals as the system that will be produced as a result of this thesis. The first is the Web-Onlne Feedback system (WOLF). This was implemented as part of research at Queensland University of Technology [Nulty, 2000]. The research aimed to “overcome the common problem with student evaluation of teaching” [Nulty, 2000]. The research found that “qualitative components of student evaluations (in particular student comments) were not being made available to teaching staff concerned until sometime after the teaching has occurred.” The system developed offered a user-friendly, web-based interface that allowed users to generate web-based questionnaires. Access to the system was 24 hours a day. The main goals for the system were:

1. Reduce/remove delays in receiving feedback from paper-based surveys.
2. Improve on the range of different ways that people have to obtain feedback on conceivably any aspect of their work or service.

3. Support the gradually changing ethos surrounding evaluation of teaching and units from one-off sporadic evaluations to that which evaluation is seen as an integral part of the daily practice.

4. Enable people to obtain context-specific feedback as and when it is needed.

5. Ability to review pre-existing questions available in item banks (for paper and pencil questions) to ensure that issues explored questions posed are adequately represented in those item banks.

It was found that academic staff opposed questionnaires which made use of a “fixed bank of optional items or consisted of a fixed set of compulsory items limiting them from conducting meaningful evaluations in relation to their context” [Nulty, 2000]. Hence, as part of the fourth goal of the system, WOLF allows people to specify their own questions.

The system was not used as expected, so the success rate was low. However, there was success with the response rate of questionnaires created on the system “by lecturers in the faculty of Information Technology and Brisbane Graduate School of Business” [Nulty, 2000].

The second system was a pilot program for an evaluation system implemented by the College of Computer Science, Mathematics and Physical Science at the University of Maryland. Instead of building a system from the ground up, the researchers made use of the features available in WebCT [Denman, Robinson & White, 2004]. The feasibility of moving
from classroom paper-based evaluation to web-based evaluation forms was investigated by the Physics department. The paper-based forms were mimicked within a web interface to a database. The success of the pilot was measured by the amount saved on paper and scanning cost (scan of paper forms). The pilot also eliminated sources of potential error (damaged or misplaced scan sheets). This also reduced the amount of time required to produce the final reports.

The University made use of the survey tool available on WebCT. The tool offered a confidential electronic means of collecting evaluation data from students on courses at the university. This information could then be transferred to the statistics lab for processing. It was the university’s desire to have a system that would send introductory emails and reminder emails to students. The frequency of the reminders would be controlled by each college. Staff desired a mechanism where a list was provided at the end of the evaluation period of students who had completed evaluations; this would then be used to allocate extra-credit to those students.

The system was set up in a way that one WebCT “course” was created per department, this was known as the evaluation space, and one ‘survey’ (a tool provided by WebCT) was created for each university course. Department Representatives (DRs) were set up, assigned and given the responsibility for very basic troubleshooting [Denman, Robinson & White, 2004]. DRs also had a tool available to them which allowed them to receive student response rates upon request for each evaluation and overall for the department. Students had access to surveys for the courses they were enrolled in for the semester. Access to surveys was given to students based on criteria of the course number and section of the course(s)
enrolled in, in a particular department. Dates where also set for the release of the survey ending the last day of the term [Denman, Robinson & White, 2004]. Icons that linked to surveys were provided on the WebCT homepage. Students were notified of surveys via emails which were sent from a website set up for DRs. Reminder messages were placed in queues with information regarding the beginning and end dates and frequency set by the DRs. Reminders were processed once a day and any messages which matched the criteria for the specific day were sent to students who had not yet responded to all their surveys in the different departments.

Denman, Robinson & White [2004] focused on student response rate as the dominant measure the college used to determine the success in the transfer from paper to online course evaluation. The response rate for the student population for the summer, fall, and spring semesters were 38%, 44%, and 31% respectively, the research did not provide response rates for previous paper-based surveys administered. The disparity of the response rates in the spring as well as the lower percentages may have been related to the fact that reminder emails sent during the fall semester were limited and dependency was placed on introductory emails and a mid-way email. Certain departments had higher response rates due to incentives such as extra credit and $50 gift certificate being offered.

The last system reviewed was developed as “part of a joint venture between Hong Kong University of Science and Technology, University of Hong Kong and Hong Kong Polytechnic University” [Ha & Mars, 1998]. The project saw the development and implementation of two web-based systems, COSSET and OSTEI. These aimed to support “student evaluation of teaching in local higher education institutes” [Ha & Mars, 1998].
COSSET is a centrally controlled system designed for collecting and processing data for summative evaluation of teaching. It is a system for formative evaluation of teaching, allowing instructors to construct their own questionnaires, gather student responses and view the evaluation results online. During the project, the team also focused attention on “evaluating the viability of the web as a valid and reliable medium for student evaluation of teaching” [Ha & Mars, 1998].

OSTEI allows instructors to access its web site and set the necessary configurations to conduct a questionnaire survey on the web. Once an instructor creates a questionnaire, with the help of a question bank, students are able to access the OSTEI student site and complete questionnaires.

The OSTEI system uses a “registration system to control access by instructors” [Ha & Mars, 1998]. When an instructor builds a questionnaire, he or she is assigned a questionnaire ID. The ID along with the instructor’s username uniquely identifies the questionnaire. For the system to retrieve the correct questionnaire from the database, a student must supply the questionnaire ID and instructor’s username. No personal information from a student is required to complete a questionnaire. This is limiting to the system as it allows students to complete a given questionnaire more than once. “It is also possible for anyone aware of a questionnaire ID and instructor username to complete a questionnaire” [Ha & Mars, 1998], even if it was not intended for them to complete it. For this reason, OSTEI is not recommended for formal teaching evaluations [Ha & Mars, 1998].
3. Requirements Analysis

3.1 Introduction

This chapter describes the system in terms of its functional and non-functional requirements. It also presents the functional model of the system. The functional model represents the use cases, which elaborate on the requirements of the system by describing the functionality of the system from an actor’s point of view. An actor is an external entity that needs to exchange information with the system. An actor can represent either a user role or another system. The analysis object model is presented in this chapter and is represented by class diagrams for the proposed system. The diagrams describe the entities manipulated by the system.

This chapter begins by introducing the scope of the system and goes on to discuss the objectives and success criteria for the proposed system. The proposed system is then detailed along with the functional and non-functional requirements. The functional model and an abstraction of the object model are also provided.

3.1.1 Scope of the system

The Online Survey System is an application that is aimed at improving the convenience of the student evaluation process, both for students and for faculty. It aims to alleviate the time and cost incurred when collecting data from course evaluations and to provide staff with the opportunity of constructing evaluations themselves. The system will be a web-based application which will be hosted in the CSU Studentwebs server. The system will allow faculty, survey administrators, to create evaluation surveys by specifying the questions they
want on the survey and the answer formats for those questions. These criterions will be stored on a database along with the intended audience as selected by the survey administrator. The survey administrator is able to create as many surveys as he or she requires and the selected target audiences for these surveys to be sent to. Once the surveys have been created, the survey administrator will need to indicate to the system that the surveys should be sent to the intended audience. The system will provide a list of all surveys that the survey administrator has created along with the title they provide for the survey, the survey creation date, the survey expiration date, which will also be provided by the survey administrator, and the date the survey was sent to the intended audience. On selecting a created survey, the survey administrator will be able to choose to send the survey to the intended audience, send a reminder email or update questions or answers on the survey provided that a notification email has not been sent out. On receipt of a survey notification, a survey taker can click the web link contained in the email. This will bring the survey taker to the system login and on authentication the survey taker will be provided with a list of surveys he or she needs to complete or have already completed. When a survey is selected, the survey will be dynamically created and displayed on the survey taker's screen. Survey takers will be able to view responses of surveys they have previously completed. A survey administrator will not be able to access individual student responses but can access statistical information on the overall responses for each survey question on selection of a survey.

3.1.2 Objectives and success criteria of the project

Online surveys provide a faster and more cost effective way of obtaining feedback from audiences [Madge, 2006]. The objective of the application is to create a system that will allow
a certain group of users (survey administrators) to create surveys that other users can complete online. Web-based surveys will be generated dynamically from the retrieval of questions stored on a database. Once a survey taker selects a survey to complete, the questions for the survey will be retrieved from the database and the survey will be dynamically generated. A survey taker needs to be authenticated by the system before he or she can complete a survey.

The system will therefore allow for the following:

- Create user login functionality
- Create a survey display page that displays “user specific” survey questions based on the group the users has been allocated to
- Create a mechanism for certain users (survey administrators) to create surveys, by providing questions and the user groups to whom the survey should be assigned
- Create a set of user groups, allowing users to fall into more than one group

3.2 Proposed System

3.2.1 Overview

The system will be a web-based application that will allow students and staff members of CSU to take surveys online. The proposed system has features from each of the systems described in section 2.5. This system will be similar to the OSTEI, but will implement a few additional features that will eliminate the limitations identified with the OSTEI system. These additional features are user authentication, which will restrict survey takers from completing a survey more than once and make surveys available to only those that they are
intended for, notification and reminder email, which will give the survey takers the notification and reminders they need of the availability of the surveys, and provision of statistical results of the responses of survey.

3.2.2 Functional Requirements

The following section provides an overview of the functional requirements of the system. Functional requirements deal with what the system should do or provide for the users. The functional requirements will detail what facilities are required and what activities the system should carry out. In other words, functional requirements define the required functional support or functionality that the system to be created should provide.

3.2.2.1 User Authentication

1. Authentication of all users should be provided by the system.

3.2.2.2 Querying Database for previous survey questions

1. Allow survey administrators to retrieve questions used in previous surveys, which are stored in a question pool on the database. The question pool should contain all previous survey questions from all surveys stored on the database.

3.2.2.3 Dynamic Survey Creation

1. The system should build surveys dynamically at run-time. That is, the survey is derived from questions stored in a database and the question response format and question numbers as specified by the survey administrator.
3.2.2.6 Generation of statistical survey result report

1. Survey administrators should be able to view statistical results of the responses to the survey.

2. Qualitative results should be made available for surveys before and after the cut-off date, depending on the responses at the time of query.

3.2.2.7 Allocation of Survey Administrators and user groups

1. A system administrator should be able to provide access to survey administrators. For example, if the department secretary wanted to create a survey, the system should have a mechanism for allowing him or her to do that.

2. System administrator should be able to create user groups as requested by survey administrators.

3.2.2.8 Viewing completed surveys

1. Survey takers should be able to view surveys that they have previously completed.

3.2.2.9 Other aspects of the system

1. Provide Internet client/server application that will allow users to connect via a login system in order to complete surveys and create surveys.

2. Inform users of any errors detected while using the system or writing to the database, loss of connection to the server, database, etc.

3. Survey Administrators should be able to modify survey questions and answers before it has been sent to users.
4. Survey Administrators should have the ability to delete created surveys.

5. System should protect user anonymity.

3.2.3 Justification of functional requirements

3.2.3.1 User Authentication

Users will be authenticated to avoid multiple submissions from the same user. The user authentication will be the same as their Novell authentication. This will enable survey takers to use the same authentication they use for logging onto the school network. Survey takers' email addresses will also be retrieved from the information held on the Novell database.

3.2.3.2 Querying Database for previous survey questions

Questions used in previous surveys should be made available for all survey administrators to save time in typing and avoid duplication of data in the database.

3.2.3.3 Dynamic Survey Creation

The system should be able to generate surveys dynamically so that survey administrators are not restricted in the number of questions they can have on a survey.

3.2.3.4 Email Notification

Survey introduction and reminder notifications should be sent to inform users that the survey is available and remind those who have yet to take the survey to do so.
3.2.3.5 Generation of statistical survey result report

The system will aggregate survey responses and present that aggregate information in the form of charts.

3.2.3.6 Protection of anonymity

To protect anonymity of the student taking the survey, the system must encapsulate all information regarding the students and surveys that they have completed. Survey takers' email addresses will be encapsulated in group names so when reminders are sent, the survey administrator is only able to view the group name of the survey takers who are being sent a reminder. Survey takers' completed surveys are only available to the survey takers themselves upon logging in the system.

3.2.4 Non-functional requirements

The Online Survey System must fulfill the following non-functional requirements. Non-functional requirements describe the user-visible aspects of the system that are not directly related with the functional behavior of the system [Bruegge & Dutoit, 2000].

3.2.4.1 User interface and human factors

The user interface should be similar to that of a standard online survey application and satisfy the ten usability heuristics¹ (see Appendix A).

¹ Taken from http://www.useit.com/papers/heuristic/heuristic_list.html
3.2.4.2 Hardware and software considerations

The required hardware for the system is as follows:

- The system should operate on the studentwebs server in the Department of Computer Science, CSU.
- The time taken for the system to load and retrieve database data will depend on the network connection over the Internet and the performance of the server.
- The user interfaces with the system via a web-based interface in the user's browser.

3.2.4.3 Performance characteristics

The system should be able to retrieve data from, and write data to, the database, and generate and display surveys for completion with minimum delay. The fact that a database connection is made on creation of surveys or retrieval of surveys should be transparent to all users. The response time between a request being made by the user and the response displayed on the user's screen/browser should be a factor in this transparency.

3.2.4.4 Error Handling and Extreme Conditions

Two possible errors could occur on the system.

1. Error in writing to the database. In the case of this error, the user will be directed to an error page and instructed on steps they need to take.

2. The client may not successfully connect to the server. The client will be informed that connection to the server was refused and advised to try again.
3.2.4.5 System Modifications

The bare minimum of the system is to allow survey administrators to create surveys and administer them through email notification to the intended survey takers. The system could be modified in the future by adding a 'find' functionality and also making all the surveys created by other survey administrators available to all survey administrators.

3.2.4.6 Operating Environment

The system will be implemented using ASP.NET C# and an SQL Server 2000 database. The system will perform best using an Internet Explorer browser. Mozilla Firefox browser can also be used.

3.2.4.7 Security Requirements

The use of the system will be user authenticated. Pages within the system will also be authenticated causing restrictions to certain pages for certain users. The Survey Taker will only be able to access the page where they can complete a survey and survey administrators will be able to access pages for survey creation purposes. The system administrator will have access to all pages.

3.3 System models

3.3.1 Scenarios

This section highlights the scenarios identified for the proposed system. A scenario is an informal, concrete, focused description of a single feature of the system from the viewpoint of a single actor [Bruegge & Dutoit, 2000].
**Scenario name:** Creation of Survey

**Participating actors:** Survey Administrator: User

**Flow of Events**

1. The user accesses the main page of the System.

2. The user is redirected to the system login page, if they are not authenticated.
   
   Once the user logs in successfully, they are redirected to the main page, where they are presented with surveys they have previously created, with the options of editing, deleting or creating a new survey.

3. The user selects creation of a new survey. The user is forwarded to a page where he/she can create a new survey.

4. The user goes through the steps of defining the survey criteria. A textbox is available to input new questions, a button to add the new question to the survey being created is next to the textbox.

5. The user inputs a question, selects a question category (if none selected, question is given the default ‘General’ category will be used) and presses the add button, this displays the question in a table on the page labeled ‘Survey Questions.’

6. The user selects a question category from the table which is used to display existing database questions. This lists questions from the question pool in the database from the selected category in the table labeled ‘Existing Questions.’ From this the user checks all questions required from the chosen category. The user then clicks the ‘Add’ button, which adds it to the list of questions for the survey being created. These questions are displayed in the table labeled ‘Survey Questions.’
7. The user clicks the ‘Continue’ button and is brought to a page where he/she can configure question answers. Questions already in the database are listed with their answers displayed below them in a table (future system, user can modify answers). New questions (questions not already in the database) do not already have a selection of answers attached to them. A user selects the question from the table of questions to provide an answer for and selects the format of the answer(s), then enters the relevant answers for that particular question (user can go back and add or delete questions from the survey).

8. User then clicks the ‘Create Survey’ button and is directed to a page that displays the created survey. If the user is happy with the survey, he/she can click the finish button to create the survey permanently on the database. The user must provide a start and expiration date for the survey to be taken by and a survey title in the textbox provided. Once the ‘Submit’ button is clicked, the user is brought back to their ‘Main’ page where they can select a survey to send to the specified users group (future system, can modify user group and expiration date, select multiple user groups for survey, this should send duplicated notifications to users) by clicking the ‘Distribute’ for a selected.

9. User logs out of the system.

**Scenario name:** Completing a Survey

**Participating actors:** Survey Taker: User

**Flow of Events**

1. User receives an email requesting he or she complete a survey.
2. The email details the survey to be taken and provides a link to the page where the survey can be accessed.

3. User accesses the main page of the system via the link in the email notification and is redirected to the login page for authentication.

4. Once authenticated, the user is sent to the main page of the system for the user with a survey taker role. Here the user is presented with a list of surveys they have previously taken and those they are required to take, with dates taken and expiration dates displayed. Expired surveys and surveys already taken are inactive.

5. The user selects a survey to complete via the survey title and is taken to a page that displays the survey. Once completed, the user clicks the ‘Submit’ button; and is redirected to the main page.

6. User logs out of the system.

7.

3.3.2 Use Case Model

This section establishes the use cases for the system and goes on to describe each use case.

“A use case represents a complete flow of events through the system in the sense that it describes a series of related interactions that result from the initiation of the use case” [Bruegge & Dutoit, 2000].

The use case diagram for the Online Survey System is shown below in Figure 1. There are three actors for the system, Table 1; an actor is an external entity, which interacts with the system. In this case the actor is the user.
<table>
<thead>
<tr>
<th><strong>System Administrator</strong></th>
<th>User is responsible for setting-up and configuring (allocating survey administrators) and for the maintenance of the system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Survey Administrator</strong></td>
<td>One of the main users of the system. These users are able to logon to the system, create surveys and administer them to survey takers, create survey taker user groups and allocate users to those groups (specify what type of users will take the survey).</td>
</tr>
<tr>
<td><strong>Survey Taker</strong></td>
<td>These are one of the main users of the system. These users use the system to complete surveys. These users should be able to log into the system and take a given survey within the time allocated.</td>
</tr>
</tbody>
</table>

**Table 1:** *Actors for the system and their roles*
Figure 1: The use case model for the propose system

3.3.3 Use Case model Descriptions

In the sections following are detailed descriptions of the use cases displayed in Figure 1.

These use cases were derived from the functional requirements listed in section 3.2.2.

3.3.3.1 The ‘Log On’ Use Case

Name: Log on
Description

A User (a System Administrator, Survey Administrator or Survey Taker) logs onto the Survey System by entering the username and password. The system identifies the user as a system administrator, survey administrator or a survey taker from login credentials.

Entry Condition:

The user accesses the main web page, which checks user authentication before entering the system.

Normal Flow

1. User directed to login page.
2. Enters user id and password.
3. Successful login.
4. The system displays the main page for the appropriate user.

Alternative Flow

The ‘Unsuccessful Login’ Alternative

1. At step 2, if the user is not authenticated, the logon screen says displayed.

The ‘Connection Refused’ Alternative

1. At step1, if the system fails to connect to the database to pull the user information for any reason, e.g. the server is not running, the system informs the user that his or her connection to the server was refused.

3.3.3.2 The ‘Create Survey’ Use Case

Name: Create Survey
Description

A survey administrator creates Survey to be stored in system database and notifies target user group of the availability of the survey.

Entry Condition:

Authentication of Survey Administrator and selection of create survey option from main page.

Normal Flow

1. Survey administrator clicks the create button on main page.

2. A form is displayed where the survey administrator can create or choose survey questions from the database.

3. User types a question(s) into the provided text box, selects question category and clicks the add button.

4. Question(s) is displayed in the table of questions for survey being created.

5. User clicks the ‘continue’ button and is taken to a page where he/she can assign answers to the given questions.

6. User selects each question at a time and enters the answer(s) for that question selecting the format of the answer and clicks the ‘add’ (answers) button.

7. Questions are displayed with their possible answers.

8. User clicks the continue button and the survey is displayed.

9. User survey title and description and provides survey expiration date.

10. User clicks the create survey button. It is compulsory for the user to provide a survey title for the survey to be created.

Alternative Flow

The ‘Use existing questions’ Alternative
1. At step 3, user can click the display button after selecting a question category.

2. Questions for the selected category are displayed.

3. User selects questions he/she wants for the survey.

4. Clicks continue.

5. Questions are displayed with their answers, nothing to modify.

The ‘Remove/Add question’ Alternative

1. At step 6 or 9, if the user wants to modify, change, or remove the selected questions, they click the back button and make the necessary changes.

3.3.3.3 The ‘View Survey’ Use Case

Name: View Survey

Description

User accesses the system to delete or edit a previously created survey.

Entry Condition:

Surveys created by user are on display.

Normal Flow

1. User selects a survey from the list of surveys they have previously created, using the select link.

2. The survey questions for the chosen survey are displayed in a table, with the answers to those questions

Alternative Flow

The ‘Delete survey’ Alternative

1. At step 1, user clicks the ‘Delete’ link to delete the survey.

The ‘Modify Survey’ Alternative
1. At step 2, user selects the question he/she wants to change, the survey selected has not yet been sent to any recipients.

2. User makes changes to question.

The 'View Results' Alternative

1. At step 1, user clicks the 'View Statistics' link to view the response rate of the survey.

2. A chart is displayed indicating the amount of responses for the survey, and the amount of responses for each question on the survey.

3.3.3.4 The 'Notify Survey Takers' Use Case

**Name:** Notify User Group

**Description**

Survey Administrator sends email to targeted survey takers for a particular survey to inform them that survey is ready to be taken.

**Entry Condition:**

Survey Administrator authenticated and is on the main page where surveys he or she has created are displayed.

**Normal Flow**

1. User clicks the select link for the desired survey in the table of created surveys.

2. User is directed to a page where the selected survey is displayed.

3. User clicks the distribute button for a survey.

4. A textbox is displayed where user can selected the targeted user group for the survey.

5. A textbox with a prewritten message is displayed; the message contains a link to the survey takers main page. The user modifies the message as they wish and presses the send button.
6. User receives a confirmation that the emails have been sent to all users in the user
group selected for the survey.

**Alternative Flow**

The ‘Reminder Alternative’

1. At step 3, user clicks reminder button

2. A textbox displaying a prewritten reminder message is displayed and the user group
to send the reminder to is entered in the ‘To’ field.

3. The user presses the send button.

4. Confirmation emails are sent to those who have not yet taken the survey from the
selected user group.

**3.3.3.5 The ‘Complete Survey’ Use Case**

**Name:** Complete Survey

**Description**

Survey taker logs on to the system having received an email(s) saying he/she has a survey to
complete.

**Entry Condition:**

User authenticated and is on the main page where surveys to be taken are displayed along
with previous surveys already taken and expired surveys.

**Normal Flow**

1. User clicks the ‘Select’ link for the survey to be completed.

2. System displays survey for user to take.

3. User completes survey.

4. Clicks submit button.
Alternative Flow

The ‘Expired survey’ Alternative

1. At Step 1, if the survey expiration date has passed, the survey displayed is not editable.

The ‘Before Start Date of Survey’ Alternative

1. At Step 1, if the survey start date is after the current date, an error message is displayed.

3.3.3.6 The ‘View Completed Survey’ Use Case

Name: View Completed Survey

Description

Survey taker logs on to the system to view a survey he/she has completed previously or take a new survey.

Entry Condition:

User authenticated and is on the main page where surveys to be taken are displayed along with previously completed surveys with the date of completion indicated.

Normal Flow

1. User clicks the select link for the survey he/she wants to view.

2. System displays survey, the survey is disabled so that the user cannot modify previously selected answers
3.3.4 Object Models

The class diagram for the Online Survey System shown in Figure 2 shows the initial classes that will implement the system. The diagram is an overview of the classes (objects) discovered and includes initial attributes and methods. The diagram also shows the relationships (association) between the object classes. Table 2 describes each object in terms of its responsibilities.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question</td>
<td>This class represents the question object that is pulled from the database or written to the database</td>
</tr>
<tr>
<td>Answer</td>
<td>This class represents the answer object that is allocated to a question when a question is created. A question possesses an answer arraylist to hold multiple answers for a particular question.</td>
</tr>
<tr>
<td>SurveyUser</td>
<td>The class represents the user who is logged onto the system. It holds the user's credentials.</td>
</tr>
<tr>
<td>QuestionCategory</td>
<td>This is a class representing the category a question is assigned on creation.</td>
</tr>
<tr>
<td>UserAuth</td>
<td>This class is used for authenticating the user that is trying to log onto the system. It connects to the database to check that the credentials the user supplied are valid.</td>
</tr>
<tr>
<td>SurveyObject</td>
<td>This is a base class that holds error managing exceptions that may occur on the system. These exceptions are written to the system log.</td>
</tr>
<tr>
<td>UserGroup</td>
<td>This class represents the group that a user can be assigned to. A user can be assigned to one or more groups and this is held in the SurveyUser object</td>
</tr>
<tr>
<td>Survey</td>
<td>This class represents a survey object that is created when a Survey Administrator creates a survey. This survey is written to the database on the request of the creator.</td>
</tr>
<tr>
<td>SurveyPageBaseClass</td>
<td>This class is a base class for most of the above classes that need to access the database and the interface (web pages) classes that need database access. It holds a database connection object and methods for writing cookies for the application.</td>
</tr>
<tr>
<td>SurveyControlBaseClass</td>
<td>This is a base class for the web controls used in the</td>
</tr>
</tbody>
</table>

2 New classes will be discovered during the design stage.
Table 2: Initial objects Descriptions for the Online Survey System.

3.3.4.1 Class Diagram

Figure 2 shows the classes (objects) identified for the system. In the design section, this diagram will be refined showing class dependencies and any other fields and methods which are yet to be identified for the system.

Figure 2: Abstract object model for proposed system.
User Interface

The graphical user interface for the system is solely for the user. Figure 3 shows the Login interface the actors of the system will interact with in order to use the system. Figure 4 shows the survey creation interface that the survey administration will interact with in order to create a survey.

Figure 3: Login Interface of Proposed system
3.5. Functional Requirements Cross-referenced and prioritized

Table 3 details all the system functional requirements prioritized and cross-referenced with the use cases. The abbreviations are as follows:

- Must Have – MH
- Should Have – SH
- Could Have - CH
<table>
<thead>
<tr>
<th>Functional Requirements</th>
<th>Priority</th>
<th>Cross Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow survey administrators to retrieve questions used in previous surveys.</td>
<td>MH</td>
<td>Use case: Create Survey</td>
</tr>
<tr>
<td>The system should build surveys dynamically at runtime.</td>
<td>MH</td>
<td>Use case: Create Survey / Complete Survey</td>
</tr>
<tr>
<td>The system should authenticate all users.</td>
<td>MH</td>
<td>Use case: LogOn</td>
</tr>
<tr>
<td>Survey administrators should be able to create surveys, specify the questions, their types, their possible responses, and their groups.</td>
<td>MH</td>
<td>Use case: Create Survey</td>
</tr>
<tr>
<td>Survey Administrators should be able to modify survey questions and answers.</td>
<td>SH</td>
<td>Use case: View Survey</td>
</tr>
<tr>
<td>Survey Administrators should have the ability to delete created surveys.</td>
<td>SH</td>
<td>Use case: View Survey</td>
</tr>
<tr>
<td>Surveys should be inaccessible for completion after expiration date.</td>
<td>MH</td>
<td>Use case: Complete Survey</td>
</tr>
<tr>
<td>Survey takers should be notified via email when they have surveys to complete.</td>
<td>MH</td>
<td>Use case: Notify Survey Takers</td>
</tr>
<tr>
<td>Reminder emails should be sent to those who have not yet completed the survey as the survey end date approaches.</td>
<td>SH</td>
<td>Use case: Notify Survey Takers</td>
</tr>
<tr>
<td>Survey Administrator should be able to view statistical results of the responses to the survey.</td>
<td>SH</td>
<td>Use case: View Survey Results</td>
</tr>
<tr>
<td>Qualitative and quantitative results should be made available for surveys after the cut off date.</td>
<td>SH</td>
<td>Use case: View Survey Results</td>
</tr>
<tr>
<td>A system administrator should be able to give access</td>
<td>MH</td>
<td>Use case: Assign Survey Administrator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use case:</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>to survey administrators.</td>
<td>SH</td>
<td>View Survey</td>
</tr>
<tr>
<td>Survey takers should be able to view surveys that they have previously completed.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Functional requirements cross-referenced with use cases.

Summary

This chapter has dealt with requirement elicitation and requirements analysis. The functional and non-functional requirements of the system were captured. The functional requirements concerned with the functionality of the system were used to develop the functional model; this is represented in UML by use case diagrams. The use cases were then described in detail using natural language; this is so that anyone reading the report without UML knowledge will understand the purpose of the use case model. The non-functional requirements focused on the user visible aspects of the system that are not directly related with the functionality of the system, i.e. the platform in which the system should operate. After identifying scenarios and use cases, the initial classes required for the system functionality were identified. This chapter established the user interface for the system and prioritized and cross-referenced the functional requirements with the use cases.
4. Design and Implementation

4.1 Introduction

This chapter is concerned with the design and implementation of the system. It discusses the objects (classes) used to implement the system and the various techniques used to build the system. This chapter focuses on defining the subsystem interface, also referred to as the application programmer interface (API).

The web-based application being developed will have a 3-tier architecture. The three layers of the architecture will be highlighted and their functionality will be discussed. In particular, the user interface layer is presented and the business logic layer will be defined by the names of classes, operations, parameters, types, and return values.

4.2 Design decisions

This section lists all decisions made for the system to provide the services it proposes effectively.

Providing the title field and description field for created surveys

As with any survey that one takes, a title is necessary to provide some indication of the purpose of the survey. The provision of a survey title and description by survey administrators on creation of a survey will serve as a way of informing the students of the purpose of the survey they are required to take. The survey title will be required, but the survey description will be optional. These will be used in the email notification to students.
Allowing survey administrators to view all answers that have previously been used for a selected question

Survey administrators will have access to answers that have been used for a selected question by any other survey administrators. The database will record all answers that have ever been used for a question. This will save time on survey creation and decrease the chances of data duplication.

All answers should not be attached to all questions

The application will only have answers that have already been selected for a question previously to be displayed when a user is formatting answers for a question. Having all answers displayed will cause a problem once the answer pool gets large as users will have to search through hundreds of answers to find one that they wish to use. Instead, allow the users to create new answers if they cannot find the one that they wish to use and when writing the new answer to the database ensure that it does not already exist.

New questions and answers written to the database during survey creation

New questions will be written to the database when the survey administrator adds them to the survey being created. New answers however will not be written until the survey administrator has actually attached the answer to the question for the survey. This will help the user to retrieve new questions and answers if the connection is lost, and limit data duplication, as the survey administrator will only be able to view answers which have been attached to a question previously.
Three step process to configure questions and answers

The application will provide a three step process for selecting answers for questions. For step 1 the user will select a question number. For step two, the user will select an answer format. For step 3 the user will select responses for the selected question. The three-step process should be visible only when ‘Set Answer’ link button for a question is selected. The ‘add new answer’ input control should remain invisible until the user clicks the ‘add new answer’ button.

Predefined answers that do not need configuring

The application will provide answer formats that do not need to give any answer. These are Open Ended (One Line), Open Ended (Multi Lines), Dichotomous (Yes/No), Likert (Agree/Disagree). If a survey administrator chooses one of these formats when configuring an answer for a question, he or she will not need to choose answers for the question, these will be already defined.

Store Surveys and questions in a session object

To avoid null pointer exceptions and limit the amount of connections to the database, store relevant survey objects and question objects in the session once data for them is retrieved from the database.

Use Dot Net Chartings to display results

.netCHARTING is a .NET control that will enable the application to display dynamically generated data quickly and easily through a visual interface. This control is written in C# and
will be integrated with the application to enable statistical view of survey results. The control was obtained from www.dotnetcharting.com.

4.3 Proposed software architecture

Overview

The proposed architecture for the system is the three tier architecture usually used in web applications. The tiers are:

1. The presentation layer or user interface layer
2. The business rules layer, and
3. The data access layer

The presentation layer consists of HTML and ASP.NET pages; these create the look and feel of the user interface. The business layer uses the code-behind classes to control the flow of the application; these classes are written in the C# programming language. These code-behind classes call other C# classes to store and retrieve data from the database and at times forward the results to the ASP.NET pages or other code-behind objects. The data layer works with data that is stored in the database.

These three layers are relatively independent and should be kept as separate and independent as possible.
Presentation Layer
This is user interface of the application. This layer is responsible for translating tasks and results into something visible and understandable by the user.

Business Logic Layer
This layer handles the application process commands, makes logical decisions and transfers data between the presentation layer and the data access layer.

Data Access Layer
This layer stores and retrieves data from a database. Data is received from the logic layer and stored in the database. Upon request data is passed to the logic layer for processing and eventually displayed on the user interface.

Figure 5: Three Tier architecture for web applications

4.4 Subsystem decomposition
The system can be divided into three subsystems that correspond to the 3-tier architecture.

The model of the 3-tier architecture can be seen in Figure 5.

Figure 6 is a diagram of the subsystem decomposition for the online survey system.

The diagram is an overview of the subsystems and their containing classes.
Survey Application subsystem

This subsystem initializes interaction with the user by providing the system interface. It contains the .aspx files that are responsible for the user interface and the code-behind files that handle the interaction. This subsystem also contains the security subsystem, which is used in authenticating all users of the system. Within the 3-tier architecture, this subsystem is the Presentation Layer.

This subsystem is responsible for interfacing between the Survey Application subsystem and the database. It holds all objects used when the user interacts with the system. This is the Business Logic Layer.

Database Subsystem

This subsystem is responsible for storing and retrieving data used by the application. This is the Data Access Layer.

Figure 6: The subsystems for the Online Survey System.
4.5 Subsystem services

As mentioned in the previous section (subsystem decomposition), the system is divided into three subsystems: the survey application subsystem, survey object subsystem, and the database subsystem. This section describes the services these subsystems provide for other subsystems. ‘A service is a set of related operations that share a common purpose’ [Bruegge & Dutoit, 2000].

Survey Application Subsystem

This subsystem is concerned with the initialization of the system and is responsible for interfacing with the user. The subsystem contains all the user interface files for all users of the system. It contains the interface for the survey takers to complete surveys and view completed surveys, for survey administrators to create and send out email notifications for created surveys and also view survey results.

Survey Object Subsystem

This subsystem interfaces the Survey Application Subsystem with the Database Subsystem. It contains twelve classes which work together to provide database connection for the survey application subsystem. This subsystem is responsible for establishing database connections when needed by the application subsystem and holding data retrieved from the database in objects for use by the survey application subsystem.

Database Subsystem

This subsystem is responsible for holding data and querying the data store for use by the Survey Object Subsystem. The entity relationship diagram can be found in Figure 24.
4.6 Survey Application Subsystem

This section provides the interface screen shots and class diagram for the Survey Application subsystem. The class descriptions for each class with this subsystem can be found in Appendix B. The diagram contains the attributes and operations of each class and the association, which relate the objects. The class descriptions describe in detail each object in terms of attributes and operations and their visibility. The following are definitions for the visibility assigned to methods or field in the following classes:

*Public*- makes the element visible to code outside the class

*Private*- private methods and fields are visible only inside the class they are defined,

*Static*- static methods and fields belong to the class as a whole, rather than to any individual instance.
4.6.1 User Interface

The user interface is the presentation layer of the system. This section highlights the web pages of the application and details how the pages are used in order to interact with the system.

4.6.1.1 Login

This is the main login page of the system. From this page users are authenticated and redirected to the page they initially requested. If the user came directly to the system login page, the user will be directed to the default.aspx page, Figure 8.

4.6.1.2 Default

The user is directed to this page from the login.aspx page, Figure 3. Here the user is presented with three links, one for System Administrators, one for Survey Administrator, and one for Survey Takers. The user selects the appropriate link depending on which of the three user roles the user falls into. If the user selects a role to which he/she is not assigned, the user is redirected to the surveyerror.aspx page, Figure 22. When the System Administrator link is clicked, the user is redirected to the sysadminmain page. When the Survey Administrator link is clicked, the user is redirected to the adminmain.aspx page, Figure 9. When the Survey Taker link is clicked, the user is redirected to the main.aspx page, Figure 19.
### User Pages

<table>
<thead>
<tr>
<th>User Pages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Administrator</td>
<td>These are pages accessible to the system administrator for the purposes of maintainence and creating user account.</td>
</tr>
<tr>
<td>Survey Administrator</td>
<td>These are pages accessible to the survey administrator, those responsible for creating surveys. From these pages you can create surveys, administer surveys to students, send reminders, view survey response rates and survey statistics for surveys the survey administrator has administered.</td>
</tr>
<tr>
<td>Survey Taker</td>
<td>These are pages accessible to survey takers. From these pages you can complete surveys, view previously completed surveys and view survey statistics for surveys you have completed.</td>
</tr>
</tbody>
</table>

4.6.1.3 **AdminMain**

The user is directed to this screen either from the login.aspx page, Figure 3, or from the default.aspx page, Figure 8. This page displays all surveys that have been created by the user in a table. The information displayed about each survey is the survey title, the start date of the survey, the expiration date of the survey, and the date the survey was emailed to
students. For each survey in the table, the user can choose to view the survey by clicking the ‘Select’ link; in this case the user is redirected to the survey.aspx page, Figure 13. The user can choose to delete the survey by clicking the ‘Delete’ link, this will delete the survey from the database and from the user's view. The user can also choose to view the response rate of the survey by clicking the ‘View Statistics’ link, in this case the user is redirected to the results.aspx page, Figure 17. Finally, on this page the user can use the ‘Create Survey’ button to start the survey creation process. Once the ‘Create Survey’ button is clicked the user is redirected to the create.aspx page, Figure 4.
Figure 9: Survey Administrator Main page

4.6.1.4 Create

Figure 10 shows the first page in the survey creation process. The user has the option of selecting questions from the database to add to the current survey. To do so the user must select a question category from the dropdown menu in the table labeled 'Select Question from Database.' Once a question category is chosen, all questions in that category on the
database are displayed in the grid. The user can select the questions he or she wants to use by checking the check boxes or the user can select all the questions by clicking the 'select all' link. The user then needs to click the 'add selected' button to add the selected questions to the current survey. These questions are then added to the grid labeled 'New Survey Questions.' The user can add a new question to the survey by clicking the 'add new question' button. A panel is displayed, Figure 10, containing a textbox where the user can input the question they wish to add to the survey. By default the 'General category is selected. The user can choose another category and press the add button. This writes the new question the database and adds it to the grid displaying questions for the current survey. In the 'New Survey Question' grid, the user can remove questions by selecting the 'remove all' link or checking the questions they want to remove and clicking the 'Remove selected' button. The continue button takes the user to answer2.aspx page, Figure 11, which is the second stage of the survey creation process. The back button takes the user to the adminmain.aspx page.
4.6.1.5 Answer2

This answer2.aspx page, Figure 12, is used by a survey administrator. Here the user selects a question from the grid by clicking the ‘Set Answer’ link and is presented with three steps. Step one is the selection of the survey question number, step two is the selection of the question response format, and step three is the selection of the response. Once the user selects an answer format, the ‘use selected’ button is displayed. The user can select answers from the answer grid or add new responses by clicking the ‘add new answer’ button. The
‘add new answer’ button displays a panel where the user can select the amount of answers he or she wants to add, Figure 12. Once this is selected and the ‘configure’ button is clicked, a textbox appears where the user can input the desired answers. The user can then press the add button, these responses are now added to the answer grid, where the user can select the answers they want for the question and click the ‘use selected’ button. This information is updated in the question grid, which now displays the question number, the question and the answer format. To change the question number, the user can select the question then select the empty choice from the question number dropdown list and click the ‘use selected’ button. This will make the previous question number available and give the user an opportunity to choose a new question number. Once all questions are configured, the user clicks the ‘create survey’ button and is redirected to the survey page.
Figure 11: Configuration of survey question responses
4.6.1.6 Survey

The survey.aspx page, Figure 13, displays the survey to the user. It displays the questions and the responses with the web controls that correspond to the answer format chosen for the question. From this page the user can choose to submit the survey, in which case the survey is written to the database and the user is redirected to the adminmain.aspx page. To submit a survey the user needs to provide a survey title and start/end dates for the survey. The user can also choose to update questions or answers by clicking one of the update buttons or
administer the survey to students by clicking the ‘distribute’ button. Finally, the user can choose to send out a reminder for the survey by clicking the ‘reminder’ button. If the user clicks an update button, he or she is redirected to the create.aspx or the answer2.aspx page. If the user chooses the ‘distribute’ of ‘reminder’ button, the user is redirected to the sendmail.aspx page.

Figure 13: Survey created by survey administrator on survey page
4.6.1.7 SendMail

The sendmail.aspx page, Figure 14, is used to send email notifications to students. Here the user can select user groups to send a survey notification. This is done by pressing the ‘To’ link, which displays a list of survey taker groups. Once the groups are selected, the user replaces the text in the text area with a message and presses the send button. This distributes the survey notification. An email is received by a survey taker, Figure 18, providing a link to log on to the system and complete the survey. For anonymity purposes, the survey administrator cannot see the email addresses of recipients. If a reminder is being sent, Figure 15, the user only needs to type a message and click the send button. The user is then redirected to the messagesent.aspx page, Figure 16.
Figure 14: Email notification on sendmail page
Figure 15: Reminders notification on sendmail page

4.6.1.8 Message Sent

The messagesent.aspx page, Figure 16, displays a confirmation of the email sent and provides a link back to the AdminMain page.
Figure 16: Email confirmation

4.6.1.9 Results

The results.aspx page, Figure 17, displays a graph of the response rate of the selected survey. The back button takes the user back to the AdminMain page.
4.6.1.10 Main

This is the main page for survey takers. This page displays all surveys that have been assigned to the user for completion and those the user has previously completed in a table. The information displayed is the survey title, the start date of the survey and the expiration date of the survey. The user can select a survey by clicking the 'Select' link; in this case the
user is redirected to the page.aspx page, Figure 20, where the user can complete the survey.

The system will warn the user if he or she tries to select a survey before the start date.

Figure 18: Email notification received by survey taker
Figure 19: Survey Taker Main page

4.6.1.11 Page

The page.aspx page, Figure 20, displays a survey for completion by a survey taker. Five survey questions are displayed per page. The user can click the previous and next links to navigate through survey pages. Once the user has completed the survey, the user clicks the 'Submit' button which writes the user's response to the database. This page informs the user
of the number of questions on the survey and the page number they are currently viewing. Users can also view surveys that they have previously taken on this page, Figure 21. However, the controls will be disabled, preventing users from modifying their responses.

Figure 20: Survey completion page
Figure 21: Previously completed survey

4.6.1.12 Survey Error

The surveymerror.aspx page, Figure 22, is used to display an error message when users try to access a page that does not correspond to their user role.
An error has occurred.

Error message: You do not have access to the page requested.
To try again, use your back button. To start over, go here.

Figure 22: Page access error page

4.7 The Survey Application Object Subsystem

This subsystem is concerned with managing data from the database, placing them in objects that can be manipulated by the Survey Application subsystem. Figure 23 illustrates the class diagram for this subsystem. The class descriptions for each class within this subsystem can be found in Appendix C.
Figure 23: Object Model (class diagram) of the Survey Application Object subsystem
4.8 The Database Subsystem

The database subsystem is the data access layer of the application. The tables held in the database and the relationships can be seen in Figure 24, which shows the associations between each table where data from the system will be stored.
Figure 24: Entity Relationship diagram of the system database.
Summary

This chapter has discussed the design and implementation process for the online survey system. It talked about the design decisions made during the implementation of the system. The 3-tier architecture of the system was discussed. The subsystems of the system were also detailed, in particular the responsibility of each subsystem was outlined and the object models for each subsystem were presented. The entity relationship diagram for data access layer of the system was also presented.
5. System Evaluation

After implementation, an evaluation of a system is carried out. This chapter focuses on the evaluation carried out on the system developed as part of this thesis. The evaluation considers the functional analysis of the system, in which the specified system functionalities are tested one by one. The results of this evaluation can be found on section 5.2.

5.1. Functionality Testing

Functionality testing ensures that the complete system complies with the functional and non-functional requirements of the system. The testing carried out in this section is comprised of functional testing. This uses black box techniques, and the test cases were derived from the use case model. The functional tests were identified by inspecting the use case model in Chapter 3 and identifying use case instances that are likely to cause failure.

5.1.1. Choosing Test Cases

The test cases were chosen by going through the use case description of each use case in the use case model in Figure 1 and finding the features of the system which are likely to fail and should be tested. What follows is a list of features from each use case which are likely to fail.

5.1.1.1. The ‘Log On’ Test Case

1. The user have been authenticated and is directed to the default page of the application, where they are provided three links and should choose the link appropriate for their user role. The System Administrator has access to all pages.

A User tries to access a page without the appropriate user role.
2. The user may try to access a page on the system without logging onto the system.

5.1.1.2. The ‘Create Survey’ Test Case

1. The user tries to create a new question which is already in the database.

2. When configuring answers for a question, the user selects an answer format that needs answers to be supplied for the question, but the user does not provide any answers. For example, if the user selects a multiple choice answer format for a question, the user needs to indicate what responses are to be used for the multiple choice options.

3. The user selects the create survey button without providing a start or end date for the survey. Alternatively, the user selects a start date which is greater than the end date.

4. The user tries to submit a survey for creation without supplying a survey title.

5.1.1.3. The ‘View Survey’ Test Case

1. The user logs onto the system with the intention of modifying a survey that has already been sent to recipients.

2. User selects a survey that has not been sent to recipients and tries to send a reminder.

3. User selects the ‘View Statistics’ link to view statistical information on a survey that has not been sent to recipients.

4. User selects the ‘View Statistics’ link to view statistical information on a survey that has been sent to recipients, but has no responses.
5.1.1.4. The ‘Complete Survey’ Test Case

1. User selects a survey that they have previously submitted in order to change their responses and resubmit.

5.1.2. Test cases and Results of Testing

<table>
<thead>
<tr>
<th>Test case Name</th>
<th>Log_On_Test (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User has been authenticated by the system</td>
</tr>
</tbody>
</table>
| Flow of test events | 1. Survey Taker attempts to access System Administrator or Survey Administrator main page via the link on the default page.  
2. Survey Taker attempts to access System Administrator main page via the link on the default page. |
| Expected system response | User should be directed to the system error screen and informed that he or she does not have access to the requested page. The system should deny access to the Survey Administrator if he/she tries to access the main page System Administrator. The Survey Taker will be denied access to the System and Survey Administrator main page. |
| Observed system response | System responds as expected. |

<table>
<thead>
<tr>
<th>Test case Name</th>
<th>LogOn_Test (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User types in the URL to access a user main page</td>
</tr>
</tbody>
</table>
| Flow of test events | 1. The user is redirected to the system login page  
2. The user logs onto the system |
<p>| Expected system response | The system redirects the user to the page initially requested. |
| Observed system response | System responds as expected. |</p>
<table>
<thead>
<tr>
<th>Test case Name</th>
<th>Create_Survey_Test (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User is on the create.aspx page selecting questions to create a survey</td>
</tr>
</tbody>
</table>
| Flow of test events | 1. User clicks the add new question button  
2. A textbox appears where user can input the question.  
3. User Clicks the add button |
| Expected system response | 1. The system informs the user that the question or one similar already exists in the database via an error message and should identify what category the question exist in order for the user to retrieve the question from the database. |
| Observed system response | System adds question to new question grid and also writes it to the database. |

<table>
<thead>
<tr>
<th>Test case Name</th>
<th>Create_Survey_Test (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User has selected survey questions and is on the answer2.aspx page configuring question responses.</td>
</tr>
</tbody>
</table>
| Flow of test events | 1. User selects a question from the question grid.  
2. The selected question is highlighted on the grid and a response configuration panel is displayed.  
3. The user selects a question number to assign the question on the survey.  
4. The user selects the multiple choice option from the answer format dropdown list.  
5. The user clicks the use selected answers button |
| Expected system response | The system should not display the chosen answer format in the grid until the user selects responses for the format. |
| Observed system response | The system makes the configuration panel invisible and sets the chosen answer format for the selected question |

<table>
<thead>
<tr>
<th>Test case Name</th>
<th>Create_Survey_Test (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User is on the survey page viewing the survey.</td>
</tr>
</tbody>
</table>
| Flow of test events | 1. User enters the survey title in the textbox provided  
2. User enters a description of the survey.  
3. User clicks the create survey button  
4. User selects a start date but does not select an end date for the survey. |
| Expected system response | The system should inform the user of the error and should not write the survey to the database until the date issue has been rectified by the user.  
Error 1: start and end date must be provided for the survey |
<table>
<thead>
<tr>
<th>Test case Name</th>
<th>Create_Survey_Test (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User has configured answers for the survey questions and is looking at the survey on the survey screen.</td>
</tr>
<tr>
<td>Flow of test events</td>
<td>The user clicks the submit button to write the survey to the database.</td>
</tr>
<tr>
<td>Expected system response</td>
<td>The system should inform the user that a survey title is required, before the survey can be submitted to the system for creation.</td>
</tr>
<tr>
<td>Observed system response</td>
<td>System responds as expected.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test case Name</th>
<th>View_Survey_Test (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User in on the AdminMain.aspx page, a list of surveys which the user has previously created is on display.</td>
</tr>
<tr>
<td>Flow of test events</td>
<td>The user selects a survey that has already been sent to students with the intention of modifying the survey.</td>
</tr>
<tr>
<td>Expected system response</td>
<td>The system should direct the user to the survey page where the survey is displayed and disable all update buttons.</td>
</tr>
<tr>
<td>Observed system response</td>
<td>System responds as expected.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test case Name</th>
<th>View_Survey_Test (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User in on the AdminMain.aspx page, a list of surveys which the user has previously created is on display.</td>
</tr>
<tr>
<td>Flow of test events</td>
<td>User selects a survey that has not been sent to students with the intention to send out a reminder.</td>
</tr>
<tr>
<td>Expected system response</td>
<td>The system should direct the user to the survey page where the survey is displayed and disable the reminder button but enable the distribute button.</td>
</tr>
<tr>
<td>Observed system response</td>
<td>System responds as expected.</td>
</tr>
<tr>
<td>Test case Name</td>
<td>View_Survey_Test (3)</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Entry Condition</td>
<td>User is on the AdminMain.aspx page, a list of surveys which the user has previously created is on display.</td>
</tr>
</tbody>
</table>
| Flow of test events | 1. User clicks the 'View Statistics' link for a survey that has not been sent out to students.  
2. User is directed to the results page |
| Expected system response | The system should display that no data could be retrieved for the survey. |
| Observed system response | System responds as expected. |

<table>
<thead>
<tr>
<th>Test case Name</th>
<th>View_Survey_Test (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User is on the AdminMain.aspx page, a list of surveys which the user has previously created is on display.</td>
</tr>
</tbody>
</table>
| Flow of test events | 1. User clicks the 'View Statistics' link for a survey that has no responses from students.  
2. User is directed to the results page |
| Expected system response | The system should display that no data could be retrieved for the survey. |
| Observed system response | System responds as expected. |

<table>
<thead>
<tr>
<th>Test case Name</th>
<th>Complete_Survey_Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Condition</td>
<td>User log on to the system and is authenticated</td>
</tr>
<tr>
<td>Flow of test events</td>
<td>User selects a survey that has already been completed with the intention of resubmitting with different responses.</td>
</tr>
<tr>
<td>Expected system response</td>
<td>The system should display the survey with the responses chosen by the user and not allow any modification of these responses.</td>
</tr>
<tr>
<td>Observed system response</td>
<td>System responds as expected.</td>
</tr>
</tbody>
</table>
Summary

This chapter has described the testing carried out on the developed system. Functional testing was carried out on the whole system. This involved finding the differences between the functional requirements and functionality of the system. Test cases were derived from the use case model developed in Chapter 3 and the tests were recorded. Two of the test cases failed, details of these can be viewed in the Create_Survey_Test (1) and Creat_Survey_Test (2) test table.
6. Conclusion

6.1. Future Enhancements

The development of the Online Survey system was successful and provided the functionalities initially proposed. However, when considering the further development and improvements for the system, a few new features were realized. These features are outlined below with an explanation of their purpose.

Updating of questions and answers to the database

Currently the option to update questions and responses are available, but when these changes are written to the database a new survey is created. It is desirable that updates to survey questions and answers be written to the database as updates rather than new records.

Ability to resend surveys to different groups of students than those which they were originally sent

Once a survey has been distributed to students, a survey administrator can only send reminders to the group of students the survey was initially distributed to. The system does not allow a survey administrator to include another group of students to receive the survey.

Additional answer formats

Currently the system supports six answer formats, these are Likert (Agree/Disagree), Dichotomous (Yes/No), Open Ended (multi lines), Open Ended (one line), Multiple Choice (multi selection), and Multiple Choice (single selection). An extension to the system would
be to support Table (multiple choice), Table (open ended), and Multiple Choice (dropdown list).

**Easier process for changing question numbers**

The process for changing a survey question number is a little tedious. An improvement to the system would be to make the survey question number selection easier by allowing users to be able to drag and drop survey questions to the position in the grid that they want the question to be numbered on the survey.

**Utilize survey descriptions**

Currently survey descriptions are not being used by the system. On improvement, the descriptions can be sent out in the notification emails.

**Usability & Effectiveness Testing**

Usability testing tests the users' understanding of the use case model. Effectiveness testing will test the user satisfaction that the ease of use of the system for survey administrators when creating and administering surveys and the ease of use and convenience of use for survey takers. All use cases identified during the development of the analysis model will be cross-referenced by the system requirements; therefore, the usability testing will aim to derive from the users whether or not the requirements of the system were met.

**System administrator interface**

The interface would provide the system administrator with the ability to create users accounts for both survey administrators and survey takers.
**Confirmation survey delete request**

This will avoid the case of a user mistakenly deleting a survey. With a delete confirmation the user will be asked if the delete operation is what was intended.

**Rearrange the survey creation process**

The survey process should be changed to ask for survey title at the first step rather than at the last step.

### 6.2. Project Summary

Although it would be ideal for every survey sent or distributed to students for the purpose of educational evaluation to be completed and returned in a timely manner, this is usually not the case. In academia, and the same is true for the non-academia world, there are many factors that contribute to recipients not completing surveys; these were discussed in Chapter 2. Due to these factors, it is difficult to get students to complete surveys in their entirety without some sort of incentive being offered.

Despite the fact that there are several influential factors that tend to prevent students from completing surveys, the key factors are:

1. Convenience to take the surveys
2. Anonymity of responses
As mentioned in Chapter 2, the success of student evaluations is dependent upon the faculty’s support of issuing student surveys; whether online or paper-based. Because faculty members are responsible for distributing the surveys, the following are some factors that hinder faculty from promoting paper-based surveys:

1. Time taken to distribute surveys
2. Resources needed to analyze survey data.

As previously mentioned, the objective of this project was to develop a web-based application that would allow faculty members to adopt the role of survey administrators and administer surveys to students that they teach. As part of this, the intent of this project was to produce a system aimed at providing CSU Computer Science faculty members with a tool to create surveys and administer these surveys online. The system developed took into consideration the factors mentioned above and implemented functionality that would protect anonymity, provide the convenience of allowing survey administrators to create and administer surveys and the convenience for survey takers to complete surveys in their own time. The system also provides an analytical tool to interpret survey responses. The attributes of the system focused on were ease of use, ease of distribution and provision of data analysis tools. The system developed utilizes user authentication in order to restrict users from completing a survey more than once and makes surveys available only to those students specified by faculty. The system does not enforce that students should answer all questions on a survey; this eliminates factors that could discourage student from completing the surveys administered to them.
On evaluation of the finished product of this project, it was evident that the objectives were met successfully. All the functional requirements listed for the proposed system in Chapter Three with a MH (Must Have) priority were implemented.
<table>
<thead>
<tr>
<th>Reference</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denman, Robinson &amp; White, 2004</td>
<td>Robinson Paulette, White Jason, &amp; Denman Daniel W., Course Evaluation Online: Putting a Structure into Place, Proceedings of the 32nd annual ACM SIGUCCS conference on User services, p 52-55, October 2004</td>
</tr>
<tr>
<td>Gaide, 2005</td>
<td>Gaide Susan, Evaluating Distance Education Programs with Online Surveys, Distance Education Report, p4-5, October 2005</td>
</tr>
<tr>
<td>Kelly &amp; Marsh, 1999</td>
<td>Kelly M &amp; Marsh J, 1999, Going online with student evaluation of teaching. Hong Kong: Evaluation of Student Experience Project, City University Hong Kong</td>
</tr>
</tbody>
</table>


Appendix A

Ten Usability Heuristics

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.
Appendix B

Survey Application Class Descriptions

The following sections describe the classes and attributes of the Survey Application subsystem. Figure 7 depicts the object model of the survey application subsystem. Any classes that are not included in these descriptions can be found in the class diagram, Figure 7.

Class _default

public class _default implements System.Web.UI.Page

This object handles the functionality of the default page of the application

**Field Summary**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>administrator</td>
<td>LinkButton</td>
<td>Link to the system administrator main page</td>
</tr>
<tr>
<td>surveyadmin</td>
<td>LinkButton</td>
<td>Link to the Survey Administrator main page</td>
</tr>
<tr>
<td>taker</td>
<td>LinkButton</td>
<td>Link to the survey taker main page</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>private void InitializeComponent()</td>
<td>Used to initialize components on the webpage and load event handlers.</td>
</tr>
<tr>
<td>private void Page_Load(object sender, System.EventArgs e)</td>
<td>Loads the web page</td>
</tr>
<tr>
<td>private void role_Command(object sender, System.Web.UI.WebControls.CommandEventArgs e)</td>
<td>Handles the 'Administrator', 'SurveyAdministrator' and 'SurveyTaker' commands of the links buttons, redirecting the user to the appropriate main page that matches the user’s role.</td>
</tr>
</tbody>
</table>
**Class AdminMain**

public class **AdminMain** implements **SurveyPageBaseClass**

This object handles the functionality of the main page for the Survey Administrator. It populates a DataGrid with all surveys created by the Survey Administrator and allows for the creation of new surveys.

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>newSurvey</strong></td>
<td>Button used for creation of new surveys.</td>
</tr>
<tr>
<td><strong>SurveyGrid</strong></td>
<td>DataGrid which holds survey object items retrieved from the database for the Survey Administrator. These are all surveys that the survey administrator has created. The DataGrid displays the survey title, creation date, expiration date and the date sent for each item.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>newSurvey_Click</strong>(object sender, System.EventArgs e)</td>
<td>This method handles the create survey button click. Once the button is clicked the user is redirected to the create page.</td>
</tr>
<tr>
<td><strong>Page_Load</strong>(object sender, System.EventArgs e)</td>
<td>Makes a request to the SurveyApplication subsystem to obtain surveys created by the Survey Administrator the first time the page is displayed in the user's browser.</td>
</tr>
<tr>
<td><strong>SurveyGrid_ItemCommand</strong>(object source, System.Web.UI.WebControls.DataGridCommandEventArgs e)</td>
<td>Handles the 'Select' and 'Delete' and 'Statistics' commands for the DataGrid. When an Item is selected using the 'Select' link on the DataGrid, this method sets a session object indicating whether the selected survey had been sent to recipients and then calls the survey page passing the survey id as a query string. When an item is selected using the 'Delete' link, this item is deleted from the grid and the database.</td>
</tr>
</tbody>
</table>

**Class answers2**

public class **answers2** implements **SurveyPageBaseClass**
Page used to configure survey question response

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Button</td>
<td><strong>addAnswerbtn</strong> Button used to add a new answer to the AnswerGrid control</td>
</tr>
<tr>
<td>protected Table</td>
<td><strong>addTable</strong> Table that holds controls for creating new answers</td>
</tr>
<tr>
<td>protected DropdownList</td>
<td><strong>AnsNo</strong> Dropdown list that holds numbers used to create textboxes for answers</td>
</tr>
<tr>
<td>protected DropdownList</td>
<td><strong>answerFormat</strong> DropdownList that holds the different answer formats that can be assigned to a question.</td>
</tr>
<tr>
<td>protected DataGrid</td>
<td><strong>AnswerGrid</strong> DataGrid that displays all answers attached to the selected question</td>
</tr>
<tr>
<td>protected int</td>
<td><strong>AnswerID</strong> Used to hold an Answer object answered</td>
</tr>
<tr>
<td>protected Button</td>
<td><strong>back</strong> Back button</td>
</tr>
<tr>
<td>protected Table</td>
<td><strong>border</strong> Table that holds answer configure controls</td>
</tr>
<tr>
<td>protected Button</td>
<td><strong>btnQuestionAdd</strong> DropdownList that holds question numbers</td>
</tr>
<tr>
<td>protected Button</td>
<td><strong>configBut</strong> Configure new answer(s) button</td>
</tr>
<tr>
<td>protected Button</td>
<td><strong>cont</strong> Continue button</td>
</tr>
<tr>
<td>protected Question</td>
<td><strong>currentQuestion</strong> Holds the currently selected question</td>
</tr>
<tr>
<td>protected Label</td>
<td><strong>FormatType</strong> Label that holds the response format of a question</td>
</tr>
<tr>
<td>protected DropdownList</td>
<td><strong>QNoddl</strong> Question number dropdown list, used to select a question number for the selected question</td>
</tr>
<tr>
<td>protected DataGrid</td>
<td><strong>QuestionGrid</strong> DataGrid that holds and displays the question objects for the survey being created</td>
</tr>
<tr>
<td>protected Label</td>
<td><strong>questionNumber</strong> Label that displays question number</td>
</tr>
</tbody>
</table>
protected `TableCell` responseCell
- TableCell that holds all controls used to create a new answer

protected `Button` `useAnswerBut`
- Use answer button

**Method Summary**

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addAnswerbtn_Click</code></td>
<td>Sets the addTable control to visible</td>
</tr>
<tr>
<td><code>answerFormat_SelectedIndexChanged</code></td>
<td>Makes the useAnswerBut visible</td>
</tr>
<tr>
<td><code>back_Click</code></td>
<td>Redirects user to create page</td>
</tr>
<tr>
<td><code>bindQuestionNos</code></td>
<td>Puts question numbers in the QNoddl DropDownList</td>
</tr>
<tr>
<td><code>btnQuestionAdd_Click</code></td>
<td>Adds new answers to the AnswerGrid and attached the answers to the question in the database</td>
</tr>
<tr>
<td><code>configBut_Click</code></td>
<td>Used to dynamically generate textboxes for new answer input, the number of textboxes to create is retrieved from the AnsNo dropdown list</td>
</tr>
<tr>
<td><code>cont_Click</code></td>
<td>Redirects user to the survey page</td>
</tr>
<tr>
<td><code>CreateChildControls</code></td>
<td>Maintains view state of dynamically created textboxes used to get new answers from user</td>
</tr>
<tr>
<td><code>CreateTextBoxes</code></td>
<td>Dynamically creates textboxes for new answers and adds the to the responseCell control</td>
</tr>
<tr>
<td><code>formats(ArrayList arr)</code></td>
<td>Puts the &quot;Select&quot; string at the first index to the answerFormat control</td>
</tr>
<tr>
<td><code>getQuestion(int id)</code></td>
<td>Sets the 'CurrentQuestion' session object to the selected question</td>
</tr>
<tr>
<td><code>InitializeComponent()</code></td>
<td>Used to initialise components on the webpage and load event handlers.</td>
</tr>
<tr>
<td><code>LoadData()</code></td>
<td>Binds the 'NewSurveyQuestion' session object to the QuestionGrid</td>
</tr>
<tr>
<td><code>numbers()</code></td>
<td>Returns a string array containing the numbers 1 to 10 and the string &quot;Select&quot; in the first index. Returned array is bound to the AnsNo control.</td>
</tr>
</tbody>
</table>
private void Page_Load(object sender, System.EventArgs e)
    Loads the page and binds the data sources for the QuestionGrid, answerFormat DropDownList and the AnsNo DropDownList

protected void QuestionGrid_ItemCommand(object source, DataGridCommandEventArgs e)
    Handles the 'Select' command for the QuestionGrid which displays the border control and binds any answers that are attached to the selected question to the AnswerGrid

private void QuestionGrid_ItemDataBound(object sender, DataGridItemEventArgs e)
    Sets the text for the FormatType of each question in the QuestionGrid if the question has a format type the text of the edcon link button is changed to 'Change' is not it remains as 'Set Answer'

protected void QuestionGrid_PageIndexChanged(object source, DataGridPageChangedEventArgs e)
    Changes the page number of the QuestionGrid

bool setSessionQuestions()
    Updates the 'NewSurveyQuestions' session object with the currently configured question.

private void useAnswerBut_Click(object sender, System.EventArgs e)
    Sets the configured answers to be used by the selected question. Sets the answer format type of the question and the responses for the question object.

---

**Class create**

class create implements SurveyPageBaseClass

Page where questions for a survey being created are selected.

**Field Summary**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected Table addTable</td>
<td>Table that holds the control used to create a new question</td>
</tr>
<tr>
<td>protected Button back</td>
<td>Back button</td>
</tr>
<tr>
<td>protected Button btnQuestionAdd</td>
<td>Buttons used to add new questions to the SurveyGrid and database</td>
</tr>
<tr>
<td>protected Button btnShowAddForm</td>
<td>Button used to show the addTable control</td>
</tr>
<tr>
<td>protected ArrayList categories</td>
<td>ArrayList that question categories from the database</td>
</tr>
<tr>
<td>protected string category</td>
<td></td>
</tr>
</tbody>
</table>
String that holds the description of a category

- **protected Button cont**
  - Continue button

- **protected Button dbQuestAdd**
  - Button used to add database questions to the SurveyGrid

- **protected DataGrid DBQuestionGrid**
  - DataGrids that hold questions from the question pool in the database

- **protected ArrayList dbQuestions**
  - ArrayList that holds questions retrieved from the database

- **protected DropDownList cDropdown**
  - DropDownList that holds question categories

- **protected Label exists**
  - Label which indicates whether a new question already exists in the database

- **protected Question question**
  - Question object

- **protected DropDownList questionCatAdd**
  - DropDownList that holds question categories

- **protected Button removebtn**
  - Remove question button, use to remove questions from the SurveyGrid and the survey being created

- **protected DataGrid SurveyGrid**
  - Datagrid that holds the questions for the survey being created.

- **protected ArrayList surveyQuestions**
  - ArrayList that holds survey questions

- **protected TextBox txtQuestionAdd**
  - Textbox input for new survey questions, questions which do not exist already

### Method Summary

- **protected void btnBack_Click(object sender, EventArgs e)**
  - Redirects the user to the AdminMain page

- **protected void btnContinue_Click(object sender, EventArgs e)**
  - Puts the SurveyGrid in the 'QuestionGrid' session object and redirects the user to the answer2 page

- **protected void btnQuestionAdd_Click(object source, EventArgs e)**
  - Adds the question that the user has entered to the database and SurveyGrid
protected void btnShowAddForm_Click(object sender, EventArgs e)
    Makes the addTable control visible

protected void dbQuestAdd_Click(object sender, System.EventArgs e)
    Handles the click event of the dbQuestionAdd button, adds the
    selected questions to the SurveyGrid

protected void DBQuestionGrid_ItemDataBound(object sender, DataGridItemEventArgs e)
    Binds category data to the eDropdown control on the
    DBQuestionGrid control

protected void eDropdown_SelectedIndexChanged(object sender, EventArgs e)
    Handles the index changed event of the eDropdown control.

protected void getCategories()
    Sets the first index of the 'Category1' session object to "Select
    Category"

private void Grid_ItemCommand(object source, DataGridCommandEventArgs e)
    Handles the 'SelectAll' and 'RemoveAll' commands of the two
    DataGrids. Checks all the questions in the datagrid to be added to the survey
    or removed from the survey

protected void Grid_PageIndexChanged(object source, DataGridPageChangedEventArgs e)
    Changes the page index of the calling grid

protected void LoadData(DataGrid source)
    Binds question data to the DBQuestionGrid and the surveygrid

private void Page_Load(object sender, System.EventArgs e)
    Instantiates the 'NewSurveyQuestions', 'Categories' session objects.
    Binds data to the DBQuestionGrid, SurveyGrid and questionCatAdd
    controls. Makes the continue button visible if the SurveyGrid has questions
    in it otherwise sets the visibility to false

protected void removebtn_Click(object sender, System.EventArgs e)
    Checks the SurveyGrid for selected questions and removes them from
    the grid

Class Main

public class Main implements SurveyPageBaseClass

This object handles the functionality of the main page for the Survey Takers. It populates a
DataGrid with all surveys that the Survey Taker has to complete and those already
completed.
Field Summary

protected Label **beforeStartDate**

Error message label. Use to display error when a user selects a survey to complete before the survey start date.

protected DataGrid **SurveyGrid**

DataGrid which holds survey object items retrieved from the database for the Survey Taker. These are all surveys that have been assigned to the Survey Taker, whether they have been completed or not. The DataGrid displays the survey title, expiration date and the date that the survey was completed.

Method Summary

private void **Page_Load**(object sender, System.EventArgs e)

Makes a request to the SurveyApplication subsystem to obtain surveys assigned to the Survey Taker the first time the page is displayed in the user's browser.

private void **SurveyGrid_ItemCommand**(object source, System.Web.UI.WebControls.DataGridCommandEventArgs e)

Handles the 'Select' command for the DataGrid. When an Item is selected using the 'Select' link on the DataGrid this method checks the start date of the selected survey and if the current date is before the state date sets the error message of the 'beforeStartDate' label and sets the table to visible. It also sets a session object indicating whether the selected survey had been has expired or has already been completed by the user and then call the 'page' page passing the survey id as a query string.

Class **messagesent**

public class **messagesent** implements System.Web.UI.Page

Displays a email sent confirmation message, listing the user groups that the survey email has been sent to.

Field Summary

protected **recipients**

Datalist of recipients that the survey email has been sent to.
Method Summary

private void Page_Load(object sender, System.EventArgs e)
    Loads the messagesent page, binding the 'SendTo' session object to the recipients dataist.

Class page

class page implements SurveyPageBaseClass

This object handles the functionality of the 'page page for the Survey Taker. Its dynamically builds the selected survey to be completed by the survey taker.

Field Summary

protected Button back
    Back button

private int EndOfPage
    The number of the last question on the current survey page

protected Label lblPages
    Label for the number of pages fro the survey

protected Label lblRecords
    Label for the number of questions on the survey

private int Leftover
    The amount leftover upon division of the number of questions on the survey and the page size

private int NumItems
    Number of questions on the selected survey

private long Pages
    Number of pages for the survey

private int PageSize
    Number of survey questions to be displayed per survey page

private int StartOfPage
    Number of the first question on a survey page

protected Button submit
    Submit survey button

protected Label surveyTitle
    Survey title label

protected HtmlTableCell tdPages
    HTMLCell that holds survey questions

private long WholePages
Number of pages that have the amount of questions displayed equal to the pagesize

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>back_Click(object sender, System.EventArgs e)</code></td>
<td>Returns to the survey taker main page</td>
</tr>
<tr>
<td><code>BuildTables()</code></td>
<td>Dynamically generates tables to hold survey questions. Determines the number of pages that a survey will have.</td>
</tr>
<tr>
<td><code>createAnswers(Question question, HtmlTableCell cell)</code></td>
<td>Dynamically creates the answers controls for questions on the survey. Disables the controls if the session object 'Taken' or 'Expired' at true.</td>
</tr>
<tr>
<td><code>CreateChildControls()</code></td>
<td>Maintains the viewstate of dynamically generated controls for the page</td>
</tr>
<tr>
<td><code>FillPages(Question Record, int tableNumber, int pageNumber, int numOfRecords)</code></td>
<td>Dynamically generates tables to hold survey questions. Determines the number of pages that a survey will have.</td>
</tr>
<tr>
<td><code>getAnswers(ArrayList answerIds, ArrayList answers)</code></td>
<td>Returns an ArrayList Answer objects for a Question object</td>
</tr>
<tr>
<td><code>getAnswersFromDB(ArrayList answerIds)</code></td>
<td>Queries the database for answers. Returns an ArrayList of Answer objects.</td>
</tr>
<tr>
<td><code>InitializeComponent()</code></td>
<td>Used to initialise components on the webpage and load event handlers.</td>
</tr>
<tr>
<td><code>Page_Load(object sender, System.EventArgs e)</code></td>
<td>Loads the survey page. Also sets the Survey title for the survey. Checks whether the survey has expired or has been taken if so disables the submit button.</td>
</tr>
<tr>
<td><code>RenderScript(int Pages, int Items)</code></td>
<td>Generates the JavaScript to handle paging for the table holding the survey questions</td>
</tr>
<tr>
<td><code>setResponse(WebControl control, Question q)</code></td>
<td>Used to set the response of survey questions for surveys which have already been taken.</td>
</tr>
<tr>
<td><code>submit_Click(object sender, System.EventArgs e)</code></td>
<td>Collects the user responses for each question in the current survey and creates a response entry in the database for the user.</td>
</tr>
<tr>
<td><code>writeResults()</code></td>
<td>Writes the responses of the survey for the user to the database.</td>
</tr>
</tbody>
</table>
Class SendMail

public class SendMail implements SurveyPageBaseClass

This object handles the functionality of the SendMail page for the Survey Administrator.

**Field Summary**

- **protected Label audienceLbl**: Labels used to identify the ListBox of groups
- **protected Button back**: Back button
- **protected Button btnSend**: Send button, used to send email
- **protected ArrayList emails**: ArrayList that holds email recipient addresses
- **protected Label from**: Label used to identify the from inputbox
- **protected string goodbye**: String that hold the closing message of the email with the survey administrator's name.
- **protected ListBox groups**: ListBox that holds all the users groups in the system.
- **protected string message**: Message to be sent to Survey Taker
- **protected string salutation**: Greeting that will be used in the email.
- **protected Button select**: Button used to select survey recipients from the groups listbox.
- **protected Label subject**: Subject of the message
- **protected LinkButton to**: LinkButton used to display the groups listbox of survey recipient groups
- **protected TextBox txtBody**: Textbox used to hold the body (message to be sent) of the email
- **protected TextBox txtFromAddress**: Textbox which holds the survey administrators email address
<table>
<thead>
<tr>
<th>Method/Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>protected</strong> TextBox txtSubject</td>
<td>Textbox which holds the subject of the email message</td>
</tr>
<tr>
<td><strong>protected</strong> TextBox txtToAddress</td>
<td>Textbox which holds the group names of the survey recipients</td>
</tr>
<tr>
<td><strong>protected</strong> HtmlTable usergroups</td>
<td>HTMLTable that holds the groups listbox</td>
</tr>
</tbody>
</table>

**Constructor Summary**

SendMail()

Sets the SmtpServer to localhost.

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>private void back_Click(object sender, System.EventArgs e)</td>
<td>Redirects the user to the survey page, where the current survey is being displayed.</td>
</tr>
<tr>
<td>private void btnSend_Click(object sender, System.EventArgs e)</td>
<td>Handles the event of the send button. Creates a mail message, gathers the 'to', 'from', subject and body of the message from the appropriate controls and send an email message.</td>
</tr>
<tr>
<td>private void Page_Load(object sender, System.EventArgs e)</td>
<td>Sets the email message body and from fields the first time the page is loaded.</td>
</tr>
<tr>
<td>private void select_Click(object sender, System.EventArgs e)</td>
<td>Handles the event of the select button. Takes all the selected user groups from the groups listbox and sets the text for the txtToAddress control. Also stores the selected groups in a 'SendTo' session object</td>
</tr>
<tr>
<td>protected string setRecipients()</td>
<td>Traverse through the 'SendTo' session object and requests the email of the groups contained from the database via Survey Application Subsystem. Writes the email addresses to a 'GroupEmails' session object. Returns a semicolon delimited string of usergroup names.</td>
</tr>
<tr>
<td>private void to_Click(object sender, System.EventArgs e)</td>
<td>Handles the event of the 'to' linkbutton. Sets the usergroups HTMLTable to visible and populates the groups listbox with group names from the database.</td>
</tr>
</tbody>
</table>
Class survey

public class survey implements SurveyPageBaseClass

Displays the selected survey for the survey administrator.

### Field Summary

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected TableCell</td>
<td>answerCell</td>
<td>Cell which holds question answer controls</td>
</tr>
<tr>
<td>protected Table</td>
<td>answerTable</td>
<td>Table which holds answer controls</td>
</tr>
<tr>
<td>protected Button</td>
<td>back</td>
<td>Back button</td>
</tr>
<tr>
<td>protected HtmlTable</td>
<td>chooseDates</td>
<td>Table that holds calendars</td>
</tr>
<tr>
<td>protected Button</td>
<td>cont</td>
<td>Create survey button</td>
</tr>
<tr>
<td>protected Survey</td>
<td>createdSurvey</td>
<td>Survey object that has just been created</td>
</tr>
<tr>
<td>protected Label</td>
<td>dateerror</td>
<td>Label used to display error message for survey creation and expiration dates</td>
</tr>
<tr>
<td>protected Label</td>
<td>DescriptionIn</td>
<td>Survey description label</td>
</tr>
<tr>
<td>protected Button</td>
<td>distribute</td>
<td>Button used to go to the SendMail page</td>
</tr>
<tr>
<td>protected Calendar</td>
<td>enddate</td>
<td>Calendar used to choose an expiration date for the survey</td>
</tr>
<tr>
<td>protected HyperLink</td>
<td>h1Next</td>
<td>Link button used forgoing to the next page of the survey</td>
</tr>
<tr>
<td>protected HyperLink</td>
<td>h1Previous</td>
<td>Link button used forgoing to the previous page of the survey</td>
</tr>
<tr>
<td>protected Pager</td>
<td>pager</td>
<td>Object used for separating the survey questions into multiple pages</td>
</tr>
<tr>
<td>protected Button</td>
<td>reminder</td>
<td>email reminder button</td>
</tr>
<tr>
<td>protected RequiredFieldValidator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
lidator

Checks whether a survey title has been provided. If a title has not been provided and error message is displayed

protected Calendar startDate
Calendar used to choose an start date for the survey

protected TextBox SurveyDescription
Textbox for survey description

protected DataList surveyQuestionsList
Datalist which holds the survey question objects

protected HtmlTable surveyTable
Parent control of the surveyQuestionList

protected TextBox SurveyTitle
Textbox for survey title

protected Label SurveyTitlelbl
Survey title label

protected Button updateA
Update answers button

protected Button updateQ
Update question button

protected HtmlTable updates
Table used to hold the update and email buttons

Method Summary

protected void back_Click(object sender, System.EventArgs e)
Redirects the user to the page that called it.

private void createAnswers(Question question, TableCell cell)
Dynamically creates the controls for each question in the survey being created

protected void createSurvey_Click(object sender, System.EventArgs e)
Handles the create survey button. Check that the start and end dates chosen for the survey are valid and writes the survey to the database

private void distribute_Click(object sender, System.EventArgs e)
Redirects user to the sendmail page

protected ArrayList getAnswers(ArrayList answerids, ArrayList answers)
Returns an ArrayList of Answer objects that have the AnswerIDs in the answerids arrayList

protected ArrayList getAnswersFromDB(ArrayList answerids)
Returns an arrayList of Answer objects which have the answerids in the answerids arraylist
private void Page_Load(object sender, System.EventArgs e)
{
    Loads the survey page, setting the 'Referrer', 'SurveyID', 'CurrentSurvey', 'NewSurveyQuestions' session objects. Checks whether the current survey has been sent, if so disables the update buttons. If the user was redirected from the answer2 page the chooseDates table is set to visible so start and end dates can be chosen for the survey being created.
}

protected void surveyQuestionsList_ItemDataBound(object sender, System.Web.UI.WebControls.DataListItemEventArgs e)
{
    
}

private void updateA_Click(object sender, System.EventArgs e)
{
    Sets the 'UpdateAnswer' session variable to true and redirects the user to the answer2 page.
}

private void updateQ_Click(object sender, System.EventArgs e)
{
    Sets the 'UpdateQuestion' session variable to true and redirects the user to the create page.
}

Class SurveyError

public class SurveyError implements System.Web.UI.Page

Handles the functionality of the Error page for the Online Survey Application. Any errors with the system is redirected to this page.

Field Summary

protected Label lblError

Label that holds the error message

Method Summary

private void InitializeComponent()

Used to initialize components on the webpage and load event handlers.
Appendix C

Class Descriptions

The following are all the relevant classes that make up the survey application object subsystem. Any classes not included can be found in the class diagram in Figure 23.

Class Answer

public class Answer extends SurveyObject

Represents question responses retrieved from the database.

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>answerID</td>
<td>int</td>
<td>ID of an answer object</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Response string</td>
</tr>
<tr>
<td>format</td>
<td>string</td>
<td>Response format</td>
</tr>
</tbody>
</table>

Constructor Summary

**Answer**(string desc)
Default constructor

**Answer**(SqlDataReader reader)
This constructor takes a SqlDataReader and attempts to load the object from it

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddAnswer</td>
<td>This method is used to add an Answer to the DB.</td>
</tr>
<tr>
<td>GetAnswerByID</td>
<td>Call a stored procedure to get the answer with the given ID, returns an Answer object</td>
</tr>
<tr>
<td>GetAnswerByQuestionID</td>
<td>Call a stored procedure to get the answer for the given question,</td>
</tr>
</tbody>
</table>
returns an ArrayList of Answers

static ArrayList GetAnswerFormats(SurveyConnection conn)
   Call a stored procedure to get all answer formats, returns an ArrayList of answer format descriptions

static ArrayList GetAnswers(SurveyConnection conn)
   Call a stored procedure to get all answers in the database, returns an ArrayList of Answers

static int GetFormatIDByDescription(SurveyConnection conn, string format, SqlTransaction trans)
   Call a stored procedure to get format ID by the given format description, returns the format id

static int GetIDByDescription(SurveyConnection conn, string answer)
   Call a stored procedure to get answer id for a given description, returns the answer id

void LoadFromReader(SqlDataReader reader)
   Extract the values from the reader and sets the Answer properties

string ToString()
   Returns the Answer description

Class Pager

public class Pager implements SurveyPageBaseClass

Object used to set pages for the survey object on the user interface.

Field Summary

private PagedDataSource pg
   Pages datasource

Constructor Summary

Pager(int pagesize)
   Constructor, sets the page size of the pager object

Method Summary

string GetNextLink(Page p)
   Returns a string identifying the next page of the pager object
Class Question

public class Question implements SurveyPageBaseClass

Represents Question objects retrieved from the database.

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>answerFormat</td>
<td>String representing the answer format</td>
</tr>
<tr>
<td>answers</td>
<td>Holds all possible answers available for this question</td>
</tr>
<tr>
<td>answerSet</td>
<td>Holds answer ids that will be used for this question on a particular survey</td>
</tr>
<tr>
<td>surveyCategory</td>
<td>Description of the category a question belongs to</td>
</tr>
<tr>
<td>categoryID</td>
<td>QuestionCategory object that the question belongs to</td>
</tr>
<tr>
<td>description</td>
<td>Description of the question</td>
</tr>
<tr>
<td>questionID</td>
<td>Question id</td>
</tr>
<tr>
<td>questionNumber</td>
<td>The number that the question will be on a particular survey</td>
</tr>
<tr>
<td>responseString</td>
<td>String representing a user response for the given question on a survey</td>
</tr>
<tr>
<td>selectedResponse</td>
<td>Holds the responses selected for a given question on a survey</td>
</tr>
<tr>
<td>surveyQuestionID</td>
<td>Identifies the survey in which the question is used</td>
</tr>
</tbody>
</table>
Constructor Summary

**Question** (string description)

Constructor, sets the question description

**Question**(SqlDataReader reader)

This constructor takes a SqlDataReader and attempts to load the object from it

Method Summary

```java
void AddQuestion(SurveyConnection conn)

This method is used to add a Question to the Database.
```

```java
void AttachAnswer(SurveyConnection conn, Answer a)

This method is used to add an answer to a Question in the Database
```

```java
void getCategoryDescription()

Sets the category description for the Question object
```

```java
static ArrayList GetQuestionByDescription(SurveyConnection conn, string description)

Call a stored procedure to get the question by description given, returns an ArrayList containing the retrieved question
```

```java
static ArrayList GetQuestionByID(SurveyConnection conn, int questionID)

Call a stored procedure to get the question from the database with the given id, returns the Question object
```

```java
static ArrayList GetQuestionsByCategory(SurveyConnection conn, int intCategoryID)

Call a stored procedure to get the question by category ID, returns all an ArrayList of all Questions in the given category
```

```java
void LoadFromReader(SqlDataReader reader)

This method is used to populate the members of this object from a SqlDataReader
```

```java
string ToString()

Returns a description of the question
```

```java
void writeSurveyQuestionResponse(SurveyConnection conn, int userSurveyResultID, string response)

This method writes the answer selected for the question in a survey to the database
```

Class QuestionCategory

public class QuestionCategory extends SurveyObject

Represents a Question Category object
### Field Summary

- **private int** `categoryID`
  - Category id
- **private string** `description`
  - Category description

### Constructor Summary

**QuestionCategory** (string des, int id)

- Constructor instantiates the category description and QuestionCategory id

**QuestionCategory** (SqlDataReader reader)

- Constructor takes in a SqlDataReader and instantiates the properties of this object

### Method Summary

- **static ArrayList** `GetAllCategories(SurveyConnection conn)`
  - Call a stored procedure to get all of the question categories, returns an ArrayList of QuestionCategories

- **static QuestionCategory** `GetCategoryByID(SurveyConnection conn, int categoryID)`
  - Call a stored procedure to get the category with the given id, returns a QuestionCategory

- **static QuestionCategory** `GetCategoryByQuestionID(SurveyConnection conn, int questionID)`
  - Call a stored procedure to get the question categories for the given question id

- **void** `LoadFromReader(SqlDataReader reader)`
  - This method is used to populate the members of this object from a SqlDataReader

- **string** `ToString()`
  - Returns the question category description

### Class Survey

**public class Survey implements SurveyPageBaseClass**

- Represents a survey object.

### Field Summary

- **private DateTime** `creation`
  - Survey creation date
private string **creatorID**  
ID of the survey creator

private string **description**  
Description of the survey

private DateTime **expire**  
Expiration date of the survey

private ArrayList **questions**  
ArrayList of Questions that make up the survey

private string **sent**  
String indicating whether the survey has been sent to recipients

private int **surveyID**  
Survey id

private string **taken**  
String indicating whether the survey has been taken for a particular survey taker

private string **title**  
Survey title

private ArrayList **userGroups**  
ArrayList of user groups to whom the survey has been assigned

---

**Constructor Summary**

**Survey**(ArrayList questions, string title, string description, DateTime creationdate, DateTime expirationdate, string creatorID)  
Constructor, sets the survey questions, title, description, creationdate, expirationdate and creator id

**Survey**(int id)  
Constructor, set the survey ID

**Survey**(SqlDataReader reader)  
This constructor takes a SqlDataReader and attempts to load the object from it

---

**Method Summary**

void **AddResult**(SurveyConnection conn)  
This method is used to add a result entry to the database to indicate the user has taken the survey.

void **AddSurvey**(SurveyConnection conn)  
This method is used to add a new survey to the database.

void **AddSurveyQuestion**(SurveyConnection conn, Question q)  
This method is used to add a new survey question to the survey.
void `AddSurveyQuestionAnswer(SurveyConnection conn, int answerID, int surveyQuestionID)`

This method is used to add an answer to a survey question.

static void `deleteSurvey(SurveyConnection conn, int surveyID)`

Deletes survey from the database.

ArrayList `GetAnswerIDsBySQID(SqlConnection conn, int sqlID)`

Get Answer by surveyQuestionid, returns an arraylist of answers.

static ArrayList `getSurveyByCreatorID(SurveyConnection conn, string creatorID)`

Returns an arraylist of surveys for the given creator id.

static Survey `getSurveyByID(SurveyConnection conn, int surveyID)`

Returns a Survey object with the given survey ID.

static ArrayList `getSurveyByUserID(SurveyConnection conn, string userID)`

Returns an arraylist of surveys for the given user id.

ArrayList `GetSurveyQuestions(SurveyConnection conn, int surveyID)`

Gets the question survey questions for the given survey id, returns an arraylist of questions.

void `GetUserSurveyQuestionResponse(SurveyConnection conn, Question question)`

Gets the given question response for the user for a survey.

void `LoadFromReader(SqlDataReader reader)`

This method is used to populate the members of this object from a SqlDataReader.

string `ToString()`

Returns the survey description.

void `updateDateSent(SurveyConnection conn, int surveyID)`

This method is used to add the sent date of a survey to the database.

---

**Class SurveyConnection**

public class `SurveyConnection` implements `SurveyObject`

Represents a Survey Connection object.

**Field Summary**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>private m_conn</td>
<td>Sql connection object</td>
</tr>
</tbody>
</table>
Constructor Summary

**SurveyConnection***(string strConnectionString)**

The constructor takes a connection string and automatically opens the connection.

Method Summary

```java
bool Close()
    Closes the database connection
```

```java
bool isOpen()
    Returns a boolean indicating whether the connection is open
```

```java
bool Open(string strConnectionString)
    Opens a database connection
```

Class SurveyUser

```java
public class SurveyUser implements SurveyPageBaseClass
```

Represents system all system users.

Field Summary

```java
private string emailAddress
    User's email address
```

```java
private string firstName
    User's first name
```

```java
protected static ArrayList groupIDs
    Arraylist of groups user is assigned to
```

```java
private string lastName
    User's last name
```

```java
private string password
    User's password
```

```java
private int roleID
    User's role id
```

```java
private string userID
    User's login id
```
**Constructor Summary**

*SurveyUser*(SqlDataReader reader)

  constructor loads the user object

---

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static getUser(string strUid, SurveyConnection conn)</td>
<td>Call a stored procedure to get user, returns a user object</td>
</tr>
<tr>
<td>void LoadFromReader(SqlDataReader reader)</td>
<td>Retrieved values for attribute of the user object from an SqlDataReader</td>
</tr>
</tbody>
</table>

---

**Class UserAuth**

public class UserAuth extends SurveyObject

Object used to authenticate a user.

---

**Field Summary**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>err</td>
<td>string</td>
<td>Error string</td>
</tr>
</tbody>
</table>

---

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string GetErrorsFormatted()</td>
<td>Returns the error string</td>
</tr>
<tr>
<td>bool IsUserValid(string strUid, string strPwd, bool isValid, SurveyConnection conn)</td>
<td>Returns a boolean indicating whether the user is a valid user</td>
</tr>
</tbody>
</table>

---

**Class UserGroup**

public class UserGroup

Represents a user group object

---

**Field Summary**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>
### Constructor Summary

**UserGroup**(SqlDataReader reader)

This constructor takes a SqlDataReader and attempts to load the object from it.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void <strong>AddUserGroupToSurvey</strong></td>
<td>This method is used to add a usergroup to a given survey.</td>
</tr>
<tr>
<td>static ArrayList <strong>GetAllGroups</strong></td>
<td>Returns an arraylist of all user groups</td>
</tr>
<tr>
<td>static <strong>GetGroupByID</strong></td>
<td>Returns a user group object for the given usergroup id</td>
</tr>
<tr>
<td>static ArrayList <strong>GetGroupEmails</strong></td>
<td>Get the email address and name of recipients format returned in ArrayList 'email;name'</td>
</tr>
<tr>
<td>static ArrayList <strong>getGroupIDs</strong></td>
<td>Returns an array List of Usergroups for the given user id</td>
</tr>
<tr>
<td>static ArrayList <strong>getGroupIDs</strong></td>
<td>Returns an ArrayList of Usergroups for the given survey id</td>
</tr>
<tr>
<td>void <strong>LoadFromReader</strong></td>
<td>This method is used to populate the members of this object from a SqlDataReader</td>
</tr>
</tbody>
</table>

**User group description**

private int **userGroupID**

User group id