A Requirement Elicitation Tool for Gathering Specifications for Data Migration

Matthew Robertshaw
Computing (with Industry)
Session (2007/2008)

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)________________________
Summary

The use of web surveys to gather data from individuals is a common practice in today’s society. As such, surveys are used by many organisations and groups as an elicitation technique for gathering views and opinions within many domains and fields of endeavour. Currently tools do exist to facilitate the creation of simple web surveys by allowing designers to formulate a survey with a set of pre-defined questions, which are then answered by other individuals. However a new challenge arises when pre-cursive and successive conditions are defined to determine which question is displayed next, given a previous answer. Furthermore, to facilitate this process a dynamic method of loading questions onto the screen is required to prepare the application for the next response.

The demand for such an elicitation tool exists within the Computer Sciences Corporation (CSC) who are currently contracted by the UK government and working collaboratively with the National Health Service (NHS) to upgrade the current NHS IT infrastructure. By providing CSC with a solution and using it as a basis for the general development of a dynamic elicitation tool, the project aims to implement an application to enable the customisable creation and answering of a dynamic web survey through the use of pre-cursive and/or successive constraints.
Acknowledgements

First of all I would like to thank my project supervisor Dr. Brandon Bennett for his invaluable advice during the development of the algorithm used in the project, as well his patience and support throughout the year. Extended thanks go to my assessor Dr. Natasha Shakhlevich for her suggestions and advice over the last year, both of whom I have enjoyed working with.

I would also like to thank the School of Computing support team, more specifically Savio Pirondini and Ollie Clark for their help in setting up my web application and database on the schools web servers.

Furthermore many thanks go to my client and friend Dr. William Thirsk–Gaskill of the Computer Sciences Corporation (CSC) as without his vast experience and foresight this project would not have come to light.

Further thanks must go to Dr. Mark Walkley, my personal tutor, for his friendly advice and support over the last four years whilst studying at the University of Leeds.

Finally, I would like to thank my friends and family for their support, patience and encouragement this year, it’s been very much appreciated.
## Contents

1 Introduction .................................................. 1
   1.1 Aim ....................................................... 1
   1.2 Structure of Report ...................................... 1
   1.3 Objectives ............................................... 2
   1.4 Minimum Requirements ................................... 2
   1.5 Schedule .................................................. 2

2 Methodologies and Technologies .............................. 3
   2.1 Methodologies ............................................ 3
      2.1.1 SSM (Soft Systems Methodology) .................... 3
      2.1.2 SSADM (Structured Systems Analysis and Design Method) 4
      2.1.3 USDP (Unified Software Development Process) .......... 5
      2.1.4 MoSCoW Prioritisation Technique .................... 6
   2.2 Technologies .............................................. 7
      2.2.1 Standalone vs. Web Based ........................... 7
      2.2.2 Client-Side Technologies ............................. 7
         2.2.2.1 XHMTL (Extensible HyperText Mark up Language) 7
         2.2.2.2 CSS (Cascading Style Sheets) .................... 8
         2.2.2.3 JavaScript ....................................... 8
      2.2.3 Server-Side Technologies ............................. 8
         2.2.3.1 ASP.NET ......................................... 8
         2.2.3.2 PHP (Hypertext Preprocessor) .................... 9
         2.2.3.3 JSP (Java Server Pages) ......................... 9
2.2.4 Data Access Layer ............................................... 10
  2.2.4.1 PostgreSQL ............................................... 10
  2.2.4.2 Microsoft SQL Server 2005 ............................. 11
2.2.5 Chosen Technologies ............................................ 11
2.3 Security ............................................................. 11
  2.3.1 Authentication ............................................... 12
  2.3.2 Membership and Roles ....................................... 12
3 Requirements Capture and Analysis ................................ 13
  3.1 Problem Background ............................................ 13
  3.2 System Overview ............................................... 14
    3.2.1 Current System In Place ................................ 14
    3.2.2 Hardware or Software Constraints ......................... 15
  3.3 Proposed Solution ............................................. 16
    3.3.1 Structuring the survey ................................... 16
    3.3.2 Proposed User Definitions ............................... 18
  3.4 Functional Requirements ....................................... 19
    3.4.1 Question Processing ...................................... 19
    3.4.2 Dynamic Controls Processing ............................ 19
    3.4.3 Report Processing ...................................... 20
    3.4.4 Required Inputs and Outputs of the System ............. 20
  3.5 Non-Functional Requirements .................................. 20
    3.5.1 Performance criteria .................................... 20
    3.5.2 Help Considerations ..................................... 20
    3.5.3 Security considerations .................................. 21
  3.6 Usability Requirements ......................................... 21
    3.6.1 Nielsens Usability Heuristics ........................... 21
    3.6.2 Application Users tasks and goals whilst using the system ............................................. 21
4 Design of Proposed Solution ...................................... 22
  4.1 Deployment Modelling .......................................... 22
  4.2 Elicitation Algorithm Design ................................... 23
Chapter 1

Introduction

1.1 Aim

The requirement for this application originates within the Computer Sciences Corporation (CSC), who are currently contracted by the government and working collaboratively with the National Health Service (NHS) to upgrade the current NHS IT infrastructure, under the acronym NPfIT.

The aim of the project is to produce an elicitation tool for CSC, to aid them in the Child Health data migration process, which forms part of the deployment framework for the Child Health System, a product CSC markets to various NHS Trusts. The solution will gather data from the users regarding their current Child Health System by asking a series of questions, each of which are dependant on the users’ answer to the previous question. The results will go on to define the rules for migrating data from a legacy application to the Child Health system.

1.2 Structure of Report

The structure of the report is based upon the workflows found within the chosen methodology (discussed in Chapter 2) with reference to its phases.
1.3 Objectives

The objectives of the project are to:

- Analyse the current system in place for data migration within CSC.
- Decide upon the system architecture (standalone or web based) based upon clients requirements.
- Research similar systems in place, methodologies and technologies that can be used for developing the system.
- Produce a design for the system.
- Build the system using the chosen technology to meet the minimum requirements.
- Test the system with suitable test data to highlight any problems.
- Evaluate the solution based upon the clients requirements.

1.4 Minimum Requirements

The minimum requirements are:

- Implementation of a database suitable for storing question sets, requirements information and user details.
- Functionality for generating dynamic web forms for requirements elicitation.
- Functionality for generating reports based on the requirements information that has been gathered using the system.

1.5 Schedule

The project schedule (found in Appendix G) is based upon the workflows found within the chosen project methodology. The workflows themselves are sub-divided into further tasks that must be completed in order for the project to progress.
Chapter 2

Methodologies and Technologies

The following chapter discusses the methodologies and technologies appropriate for the project. After a critical discussion and analysis, a conclusion is made as to which development methodology is to be followed and the technologies used for implementing a solution.

As stated, the purpose of the project is to produce a software solution to facilitate requirements elicitation. As the clients requirements for the project are well defined, a fully fledged methodology will be employed for their capture and analysis. However at the same time, the chosen methodology will be required to support the design, implementation and testing of a solution but more specifically an elicitation algorithm which in itself forms the basis of the project. An appropriate technology is also required with the ability to create dynamic controls, allowing questions to be answered. Finally, given the projects complex nature the methodology should be flexible and allow for an iterative design and development approach.

2.1 Methodologies

2.1.1 SSM (Soft Systems Methodology)

The Soft Systems Methodology is used to assess the business processes prior to any Information Systems development. Skidmore and Eva point out that the methodology is more concerned with the assessment
of organisational strategies and processes rather than on how technological solutions can be applied and integrated [26]. Although this would be useful in assessing the clients current organisational processes, it doesn’t provide the developer with a means of designing and implementing a requirements elicitation system. In doing so, the methodology itself, seems to provide superfluous steps and diverts attention away from what the project is attempting to achieve.

Whilst Avison and Fitzgerald [7] discuss the issue of having debates with the stakeholders of the system to further understand the problem, such steps seem irrelevant for the project since the clients requirements are clear and more concern should be placed upon the development of an efficient elicitation algorithm, which requires theoretical development rather than stakeholder debates and consultations.

Furthermore within the seven stage SSM framework, the decision as to the type of solution appropriate is not made until the final stage and “may, or may not, include computers” [26], indicating no guarantee of a technical solution. In theory, such an assessment should have already been completed by the client who found a software solution necessary. Therefore, given these limitations and its extensive focus on organisational practices in conjunction with a lack of desired characteristics of a methodology, SSM would not be appropriate.

### 2.1.2 SSADM (Structured Systems Analysis and Design Method)

SSADM is a structured methodology that has been proven successful in the analysis and design of software applications. Similar to SSM, implementation is not considered with a large amount of time spent on feasibility studies and assessing all possible requirements which as Bennett et al [9] discusses risks making the project unmanageably complex. SSADM also provides flexibility for design and would therefore be scheduled in the project plan but as Avison and Fitzgerald [7] point out a lack of consideration with regards to implementation (categorised as installation-specific), highlights its inappropriateness, given the project is aimed at the incremental implementation of a requirements elicitation tool.

In addition to this, Avison and Fitzgerald further note that SSADM was originally aimed at large-scale UK Civil Service projects [7] with Bocij et al [11] indicating its use in conjunction with PRINCE (used for project management). However due to the small comparable size and nature of this project with a government contract, it provides further justification for not using SSADM.

In conclusion, although SSADM would provide useful in the requirements gathering and analysis phase of the project, implementation is not considered and given the methodologies association with large scale projects, it may in itself overcomplicate this project. Also given the complex nature of
SSADM and its strict adherence to time management, focus could be misplaced on meeting successive milestone deadlines, rather than on the flexibility and iterative delivery associated with more agile methodologies.

2.1.3 USDP (Unified Software Development Process)

The USDP (originally proposed by Jacobson et al [17]), is an iterative development process used in the design and development of software applications. Jacobson et al recognises that the USDP is not a methodology but rather a set of processes that can be adapted to any project [17], which highlights its flexibility and possible usage for the project. A characteristic of the project is that the design and implementation of the solution is carried out via an iterative process. By selecting the USDP, it would satisfy this condition as within the process itself Jacobson et al [17] claim that the work is divided into smaller tasks, known as iterations. From further research it is apparent that the USDP itself is divided into four phases (inception, elaboration, construction and transition) which span over five workflows (bulleted below). The inception phase deals primarily with requirements analysis [9], whereas the elaboration and construction phases cover the design and implementation aspects of the project (respectively) with the final transition phase concerned with deployment. In addition, Jacobson et al [17] suggest that testing is carried through beta releases. Whilst testing will be carried out throughout the implementation and at the end, the use of beta releases go beyond the scope of the project.

Owing to the fact that USDP is use-case driven, as discussed by Jacobson et al [17], Avison and Fitzgerald [7] make the case for using UML (Unified Modelling Language) use case diagrams as part of the initial process, in order to capture and model user requirements. Although this may not be directly useful for the development of the elicitation algorithm, it will prove invaluable when capturing the requirements from the client and modelling workflows, which after analysis will go on to define the application design.

Therefore, given its iterative nature, the USDP would be a suitable choice as per the projects characteristics. As mentioned task iterations may occur which indicate the flexibility of the process typically unseen with earlier methodologies. In using an iterative process, several iterations of the elicitation algorithm could be developed and the application as a whole can be developed incrementally, to ensure that the critical components of the design are implemented first. Jacobson et al [17] breaks the USDP down into the five workflows:

- **Requirements**: The stage at which the requirements for the proposed system are gathered from
users and other available sources.

- **Analysis:** Use the requirements obtained to produce a specification. Avison and Fitzgerald [7] suggests at this stage only functional requirements are analysed.

- **Design:** Outlines a design for the system using the points raised in the requirements specification. It is here where Avison and Fitzgerald [7] consider the research of non-functional requirements.

- **Implementation:** At this stage the design of the system is implemented.

- **Test:** Testing occurs to ensure the system functions correctly.

Following the assessment of the methodologies and processes above, the USDP will be used as the development framework, not only because of its suitability for the project but also due to the authors previous experience with it. As the SSM seems to be a methodology carried out before the development of a solution and the SSADMs lack of guidance for tasks post design and its association with typically large scale projects, both methodologies do not fulfil the characteristics of the project and therefore are not suitable.

### 2.1.4 MoSCoW Prioritisation Technique

MoSCoW is technique used for prioritising tasks during a project. Bennett et al refers to this overall process as Timeboxing [10]. The technique can be used in conjunction with the USDP methodology to prioritise the use case requirements, derived from the UML modelling. Furthermore, by using MoSCoW it would ensure that the work required to fulfil the minimum requirements is prioritised over the other tasks. MoSCoW uses rules to prioritise tasks (requirements), referred to as ‘Must haves’, ‘Should haves’, ‘Could haves’ and ‘Won’t haves’ [10]. During the requirements analysis phase of the project, UML use cases will be assigned to one of these rules. The ‘Must haves’ and ‘Should haves’ will be implemented as part of the first iteration and will include the minimum requirements for the project (hence the need to be completed as early as possible). Further tasks will then be assigned to the remaining rules, depending upon task criticality.
2.2 Technologies

2.2.1 Standalone vs. Web Based

In producing a solution, it is important to analyse the architectural options available. A standalone application would be the simplest approach in building the system. However, a lack of scalability will require users of the application to have it installed locally. Furthermore, as the client intends to enable other individuals to use the application (at different physical locations), suggests that a standalone system will not be appropriate. Developing a web based solution does indeed have security implications, however the scalable nature of such systems and the clients plans suggest that a web based system is necessary. Platform dependence is therefore not an issue and security risks can be reduced by following appropriate and secure development techniques. The organisations current underlying infrastructure also supports the deployment of web based applications and is therefore an appropriate method. Owing to these facts, a web based solution will be produced.

2.2.2 Client-Side Technologies

2.2.2.1 XHTML (Extensible HyperText Mark up Language)

The W3C describes XHTML as a family of document types that extends to the original functionality found within HTML 4.01 and is based upon XML (Extensible Markup Language) [6]. In producing the web based solution, XHTML will be used to structurally define the content of a web page through the use of tags. Furthermore, as the application is required to generate dynamic controls for requirements elicitation, XHTML must be able to support the creation and layout of such controls. With regards to the future development of the solution, both Sebesta [24] and the W3C [6] agree that XHTML should be used. This could be due to its close association with XML, which is now seen as the long-term standard for structurally defining web pages.

Therefore whilst it’s important to consider the future benefits of using XHTML, consistency and conformance must also be highly prioritised when selecting a suitable technology, to ensure the web site behaviour is the same across different web browsers.
2.2.2.2 CSS (Cascading Style Sheets)

Cascading Style Sheets were introduced in the 1990s and provide web developers with a way of separating the web pages content from its style and layout. The W3C [3] indicates that style sheets can be implemented at a document level, inline level or external level. Although inline level style sheets may be considered, the solution will attempt to implement external level style sheets, so as to separate the content and layout further and allow for a consistent style across multiple web pages. Lastly, with having experience of writing Cascading Style Sheets, their creation and integration into the presentation layer should be met with relative ease.

2.2.2.3 JavaScript

Given that the solution will require different forms of user input, validating that input is important for security and integrity reasons. Hoque [15] discusses the use of JavaScript for validating input before the data is sent to the server. As such this will be implemented to ensure that the necessary fields are populated before submission to the database. Hoque [15] also notes the use of dynamic HTML, which may be considered when implementing the dynamic parts of the project.

2.2.3 Server-Side Technologies

2.2.3.1 ASP.NET

ASP.NET (Active Server Pages) is a Microsoft server-side scripting language, used in the development of dynamic web pages. The scripting language itself is part of the .NET framework, which Sebesta describes is a Microsoft term for the collection of its various technologies [24].

Liberty & Hurwitz argue that ASP.NET is now one of the most popular ways of writing interactive web applications and is regarded as the alternative to JSP (Java Server Pages) [19]. Given that Sebesta [24] notes that the programming code used in ASP.NET is already compiled when on the server (unlike JSP) implies that performance will be high and its use will be beneficial given the requirement for generating dynamic form controls, which may be expensive if the code is not pre-compiled on the server.

Each ASP.NET page has associated with it a code-behind file, containing the application logic code to control the dynamic behaviour of the web page in question. It also provides a method of separating the presentation layer from the application logic layer. The programming languages C# and VB can be
used to write the code-behind files but for the purposes of the project, C# will be used due to previous experiences had with the language.

In order to access a database containing the users, question details and answers for the project, Microsoft’s ActiveX Data Objects (ADO) connectivity drivers will be used to manage secure database connections.

As the clients organisation currently have the infrastructure for building ASP.NET web sites and given past knowledge and experience of the technology, it would seem appropriate to build the system using ASP.NET and C#.

2.2.3.2 PHP (Hypertext Preprocessor)

PHP is an open source server side scripting language, used for the creation of dynamic web pages. After its initial development in 1994 “by 1997 more than 50,000 Web sites were using PHP/FI to accomplish different tasks-connecting to a database, displaying dynamic content” [20], an incredible uptake in just 3 years, though given the fact that PHP is open source the figure is hardly surprising.

In terms of performance, PHP works by receiving a request for a PHP page via a web browser at which point the request is passed to the PHP parser, to process the requested page and then sends the output to the clients web browser in HTML form [20]. This indicates a heavy workload on the server which may lead to slow performance when many requests are made for creating dynamic controls and as such would not outperform methods where the code is pre-compiled.

PHP also has functionality to maintain state and database connections, of which are handled by the necessary drivers. These are further arguments for the use of PHP in developing a solution, given the projects characteristics. Meloni [20] argues that the main advantage of using PHP over others is it’s platform independence. Although this is an important factor if the web site is migrated to another web server (running on a different platform), it is not important for the project since the clients organisation currently have infrastructure in place to handle this.

Such attributes make PHP a suitable technology to use, however due to potential performance overheads and a lack of knowledge and experience in PHP it would not be an appropriate choice.

2.2.3.3 JSP (Java Server Pages)

Java Server Pages are recognised as the alternative to using Active Server Pages for developing page content but Sebesta [24] points out that JSP was built on top of the functionality provided by Java
Servlets. Although built upon the same functionality, JSPs are used when the content of a page is loaded statically, where as Servlets are necessary when the page is loaded dynamically [24] but this tends to lead to a mix of Java and XHTML in a single file (a mix of application logic and presentation).

Sebesta notes that the technical implementation of a JSP results in a compiled Servlet after a request for that JSP page has been processed through the container (which is basically a component of JSP which compiles the JSP document) [24]. However it can be argued that such intensive processing of requests will have a negative impact on performance.

As a technology JSP would not be sufficient given its static nature. Although the clients infrastructure does support JSP, a lack of experience and potential performance overheads suggests that it would not be an appropriate choice of technology.

2.2.4 Data Access Layer

2.2.4.1 PostgreSQL

PostgreSQL is a platform independent open source Relational Database Management System (RDMS). PostgreSQL [1] points out that now with over 15 years worth of usage it has won a strong acclaim amongst software developers, as a reliable database system. Furthermore it supports the concept of an ACID transaction, which basically stipulates that a transaction within a database should take place completely or not at all.

Given that PostgreSQL [1] highlights the database systems scalability, this must be considered for the proposed solution since the number of question details, answers and users within the database will inevitably grow and the database must be able to cope with that. Stored procedures are also available for use within PostgreSQL, which is appealing to a developer due to the enhanced security they can provide to a system by reducing the risks posed by SQL injections, which are attacks that attempt to change the way an SQL statement works [16]. Furthermore using stored procedures cleanly separates the application logic layer from the data access layer as the developer does not have to mix SQL with application code.

It is clear therefore that PostgreSQL is a widely accepted RDBMS with recognition from many developers and organisations. With some prior experience with PostgreSQL it could be used, however given my clients current technical infrastructure, PostgreSQL is not supported.
2.2.4.2 Microsoft SQL Server 2005

Microsoft SQL Server is a platform dependent relational database system for “large-scale online transaction processing (OLTP), data warehousing, and e-commerce applications” [4], indicating its potential for use. SQL Server has a built in front end user interface known as the Management Studio, which provides the user with a graphical interface to the application. Furthermore it provides the developer with a set of tools to create and maintain large databases. Microsoft [4] claim that SQL Server operates closely with their Visual Studio integrated development environment (IDE). Such a close relationship between the two suggests that developing the solution will be easier, since there will be no apparent platform issues. Stored procedures are also supported by the system which is beneficial for the security reasons noted in section 2.2.4.1. Further database security must also be implemented in the form of a username and password to allow the application to connect to the database and as the security features within SQL Server are highly precise and configurable [4], such requirements will be achievable.

Using SQL Server comes at the expense of financial costs and platform dependency, however as the clients organisation currently uses SQL Server, integrating the projects database should be relatively simple and at a low cost. Finally, having had previous experience with SQL Server makes it an appropriate choice of database system.

2.2.5 Chosen Technologies

- **ASP.NET**: Will use XHTML to define the structure of the web pages content. The ‘code-behind’ files will be programmed using C#. Furthermore, JavaScript will be utilised for validation purposes. The entire web application will be deployed via an Internet Information Services®(IIS) web server.

- **Microsoft SQL Server 2005®**: Used to store the questions, answers and user details centrally. SQL stored procedures will be used to query the data held within the database.

2.3 Security

Given the application is web based security is of utmost importance. ASP.NET provides a host of security features within its framework but the purpose of this section is to address just some of those features available, more specifically the use of Forms Authentication, Membership and Roles.
2.3.1 Authentication

Liberty & Hurwitz define authentication as a means of ensuring that an individual claiming to be someone, is indeed that individual [19]. ASP.NET provides such functionality through the Forms Authentication technique, which allows the user to login to an application via a form, meaning less coding is required from the developer. To ensure the user is authenticated on each request to the server and therefore heighten security, a cookie is sent to the client upon authentication to indicate an authenticated user, which is then passed back to the server with every request [19]. In using Forms Authentication, the developer will be able to make use of communication gateways that exist between the application layer and the data access layer, with minimal intervention on the part of the developer. Furthermore and most importantly, this method is far more secure than the developer implementing a bespoke authentication procedure, due to a lack of secure computing experience on the developers part.

2.3.2 Membership and Roles

Membership is a part of ASP.NET that provides functionality for managing and validating user credentials. The Membership class itself contains methods for creating users, deleting users and validating users amongst others, which when called will invoke the relevant pre-coded ASP.NET stored procedures within the SQL Server database.

In order for the application logic layer to communicate with the data access layer, a Membership provider must be defined and placed within the web.config file, explained later in the report. Therefore, in using the static Membership class, a simple yet seamless way of dealing with user administration is provided, without the need for developing new methods.

Liberty & Hurwitz [19] discuss ASP.NETs functionality for setting permissions to a group of users through Roles, implemented in the same way as Membership. As such it will be used to ensure that a non-administrative user will not be able to access administrative functionality within the web application.
Chapter 3

Requirements Capture and Analysis

The proceeding chapter captures and outlines the clients requirements obtained through a meeting (notes from which can be found in Figures B.12 to B.14, Appendix B) and includes a detailed analysis of the findings. In doing a requirements specification is derived to fulfil the first and second workflows of the USDP. Furthermore it satisfies the entirety of the inception phase and the necessary components of the elaboration phase of the USDP. All acronyms and keywords throughout this chapter are in italic whose definitions can be found within the project dictionary in Figure B.11, Appendix B.

3.1 Problem Background

The requirement for this application originates within the Computer Sciences Corporation (CSC), who are currently contracted by the government and working collaboratively with the National Health Service (NHS) to upgrade the current NHS IT infrastructure, under the acronym NPfIT. The aim is to improve patient care by means of a national database of patient records including every person in England.

As part of this NHS project, different applications exist such as CH, Community Care, SAP, Out of Ours, Prison and Hospice. Others are also under development including Diabetes and Walk-in Centre. All these are modules of SystmOne (spelled correctly), a Primary Care solution. It is important to note
at this point that the abbreviation \textit{CH} refers to Child Health throughout this document. It is within this Primary Care solution where the idea for this final year project was conceived. My client deals with the migration of data from \textit{legacy CH systems} to the new \textit{CH system}, a module which is part of the SystmOne solution. Clients for this data migration service are \textit{NHS Trusts}.

Whilst on work placement with CSC, a colleague (the client) provided an opportunity to produce an advanced elicitation tool, which would gather information regarding the requirements for the \textit{CH} data migration process.

### 3.2 System Overview

The \textit{CH} deployment process is made up of many milestones and stages which represent a complex methodology in themselves. It is not the purpose of this document to analyse this whole process but rather to concentrate on one specific area of the deployment process, which is Data Migration. Most of the critical path to go-live in the vast majority of projects is the data migration stage.

#### 3.2.1 Current System In Place

The process of data migration requires that certain rules and constraints be defined before any data is transferred. Information that needs to be captured regarding this process includes; Business rules, Data recording conventions, Functional constraints.

The old method of obtaining this information was by asking a list of questions of the \textit{NHS CH users} responsible for their \textit{CH System} and recording the answers on paper. After each successive \textit{CH Project}, new questions were added to the list. The list of questions grew unmanageably long and there was a risk of asking questions which were not relevant to the \textit{NHS Trust} currently under consideration. There was also a risk of alienating some \textit{NHS CH users} by asking questions not relevant to their current \textit{CH System}. Furthermore my client has pointed out that asking \textit{NHS CH Users} too many questions can lead to hasty or incomplete answers which can potentially lead to incorrect data migration rules. Mistakes or ambiguity in the migration rules can be very costly and cause the \textit{CH project} to overrun. The client has made it clear that this is a crucial step in the lifetime of a \textit{CH System} deployment and getting the rules right at this stage is of up utmost importance.

During a work placement year the client and author collaborated on a primitive (short-term) application whereby rules were gathered for this process by asking a series of questions of the \textit{NHS CH users}. 
The application is still in use, despite its limited functional scope. It has been suggested however that a more advanced version of the application be created.

The current program is a standalone application which operates locally on a client machine. The application is used by the client who sits with an *NHS CH User* and keys in their answers to a series of questions which are hard-coded into the system. The actual decision making as to whether a question should be answered or not based upon previous answers is again hard-coded into the application. Upon completing the questions, the application then exports all the questions and answers to a Microsoft Word® document. This document is then used by the client in formulating the data migration rules (beyond the scope of this project). In assessing the current application, the following limitations have been identified:

- The current application has all the questions hard coded, and therefore is not expandable.
- As the application sits locally on a machine, other users must also have the application installed. It is therefore not scalable or easily deployable. The client is the only person who uses the application to date but has stressed that others would be trained to use it.
- The answers to the questions are not stored centrally on a database and rely upon the client organising the results (in the form of Microsoft Word® documents) manually into folders.
- The decision making as to which question should be asked next is hard coded so there is no possibility of re-ordering questions.
- The current application assumes that *NHS CH Users* have the same job title (and hence same technical knowledge). In practice this is not the case and either an *NHS CH Technical User* or an *NHS CH Operational User* will be answering questions. Therefore questions asked are dependent on an *NHS CH Users* role.

### 3.2.2 Hardware or Software Constraints

The constraints are that the current infrastructure supports applications written in the .NET framework 2.0®, through the use of Internet Information Services®. Furthermore, the database management system software is Microsoft SQL Server®.
3.3 Proposed Solution

The proposed solution is to develop a replacement application which enhances the functionality and flexibility of the current system. It must provide a user friendly environment for *NHS CH Users* to answer questions regarding their *CH System*. The proposed application would be used as a tool in gathering rules and definitions for the migration of data from the *legacy CH System* to the new *CH System*. The solution would sit on a web server and communicate with a database. The database would be used to store user details such as usernames and passwords along with the questions (provided by the administrator) and answers provided by the users.

The application users (section 3.3.2) will complete questions for their *legacy CH System* during a *Q&A Session*, which will be given the title of the *CH Project*. When the user submits an answer, a new question will be loaded but will depend upon the answer to the previous question. This will ensure that only related questions are displayed (important as this method will attempt to get as much detail (and clarity) in the answers as possible). Finally when the *Q&A Session* is complete, a report will be generated for the user to view, detailing which questions were answered along with the answers themselves.

During the initial meeting the client provided a set of questions that the system would be required to store. These can be found in Appendix B and will be used when testing the system. Please refer to the UML diagrams found in Appendix B which illustrate user interactions through UML use case and activity diagrams. Furthermore, the MoSCoW prioritisation technique (section 2.1.4) has been used to classify the use cases in order of importance (Figure B.1 in Appendix B). Use cases relating to the minimum requirements have been classified as 'Must haves' and 'Should haves.'

3.3.1 Structuring the survey

In order to illustrate the problem (and a possible solution) consider the following questions (unrelated to the project):

1. Do you like Italian food?
2. What is your favourite Italian meal?
3. Why don’t you like Italian food?
4. Do you like English food?
As can be seen these questions are related to the culinary domain, with Q1 as the entry point into the survey. Here the users’ answer will determine which question is asked next. If Q1 = True, then Q2 should be asked next. If Q1 = False, then Q3 should be asked next. This therefore demonstrates the nature of the problem. The modelling becomes more complicated when the user has finished answering either Q2 or Q3 thus, when Q2 has been answered, how will the application know that it has to go to Q4 next? Q3 would not have this problem since Q4 logically comes after Q3. The problem becomes more complex when there are many questions each with individual dependencies. Table 3.1 models this domain but would not be a suitable solution. The concepts however may be useful in developing a prototype.

<table>
<thead>
<tr>
<th>Question ID</th>
<th>Dependency</th>
<th>Model Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>4</td>
<td>2 or 3</td>
<td></td>
</tr>
</tbody>
</table>

Belani et al [8] discusses the implementation of a web based requirements elicitation tool, called a “Web-Survey” [8]. Most of the principles outlined by Belani et al [8] can be adopted to meet the needs of this project. An architecture is proposed whereby functionality exists for “creating surveys, responding to surveys and browsing filtered survey statistics.” [8]. The ability to create surveys and respond to them is a requirement of the proposed system, as this will form the basis of the entire solution. Belani et al [8] defines two distinct users of a web survey; Creators and Responders. Creators are users who have the ability to create surveys and the Responders provide answers to them. In the case of this project, the Creator would be the application administrator and the responders would be the NHS Child Health Users. Belani et al [8] goes onto identify five types of possible questions, including “text-box, text-line, combo-box, radio-buttons and check-boxes.” [8]. These are the types of answers that are expected and are defined by the Creator when building a web-survey.

The solution will therefore require an implementation of a database to store the questions, answers and user details. Furthermore, a graphical user interface will be necessary to allow the Responder to enter answers. The controls (e.g. a checkbox) as Belani et al [8] discusses, will need to be generated...
dynamically onto a form. Hard coding the controls into the application will not be feasible since each
question may require a different type of answer (e.g. a written answer will require a textbox).

There are currently web-survey tools that take a similar approach, such as SurveyMonkey [5] and
Bristol Online Surveys [2]. However both tools do not implement functionality to allow the next ques-
tion to be pre-determined by the previous answer as well as question insertion and re-ordering. In order
for such functionality to be achieved an algorithm will need to be implemented to look at previous an-
swers to ascertain which question to display next. This will require each question to have associated
predecessors and successors. The predecessor will indicate which question needs to be already fulfilled
in order for the current question to be considered. The successor will indicate which question is selected
next, given that the current question has just been answered.

3.3.2 Proposed User Definitions

As explained, the CH System has two types of user: an NHS CH Operational User and an NHS CH
Database Administrator. Figure B.3 in Appendix B, illustrates a UML use case diagram, showing how
the NHS CH Users have been derived by considering the business and system domains.

The NHS CH Operational User is an individual who uses a legacy CH System. They have little or
no technical knowledge regarding their legacy CH system, meaning that asking these users technical
questions is unwarranted. The NHS CH Database Administrator will have knowledge of the technical
capabilities of their legacy CH system, meaning that they should be asked technically oriented questions.
The types of NHS CH Users are therefore external to the proposed solution but their job types can be
mapped onto the user types of the proposed application. In order to map the NHS CH Users onto the
proposed application, it is proposed that the solution uses the following user groups for its definition of
an application user:

- Limited Application User
- Application Administrator

The Limited Application User is in all cases equal to NHS CH Users of type; NHS CH Operational
User and NHS CH Database Administrator. The Limited Application User will only be allowed to an-
swer questions that are specific to their legacy CH System. The Application Administrator; an individual
in charge of the data migration process for the CH system within CSC, should be able to, create new
application user accounts, change application users passwords, delete application user accounts, add
new questions, edit the questions and other related details (i.e. the criteria for when it should appear) and finally change the order of questions (though this is not critical).

Furthermore, an *NHS CH Operational User* and *NHS CH Database Administrator* may well be (in some cases) the same person. An important point to bear in mind here is that this application is intended to work within any domain or field of endeavour, not just the *CH System*. The report is based on the *CH system* as means of explaining the concepts and theories behind the structure of a dynamic web survey system. So an *NHS CH User* can be thought of as an individual whose field of endeavour we are attempting to find information about (a person answering the survey).

### 3.4 Functional Requirements

#### 3.4.1 Question Processing

The question processing within the proposed application will be the root of its functionality. A method will need to be employed so that, for example, when a user answers question \( x \), the answer will be submitted to the database and processed, then the next question will be displayed. This could be achieved through the use of SQL queries. Therefore the next question to be displayed is dependent on the answer to the previous question. Furthermore, the process of selecting the next question to display is done entirely within the database. This will result in a more robust and generic application which has the potential to be applied to any domain. It will also be far more configurable than the current application as it will allow questions to be added, removed and edited within the database.

With regards to the questions, the *application administrator* will have functionality to add questions, alter the question ordering and edit the question itself. This will be achieved via an administration area within the application (only accessible by the *application administrator*), which should display all the questions within the system. Furthermore, each question will be assigned as either ‘Operational’ or ‘Technical’, thus ensuring that questions regarding Operational or Technical matters are asked to the correct type of application user.

#### 3.4.2 Dynamic Controls Processing

As there is no guaranteed order of the questions being displayed, an efficient method of displaying the questions to the application user is required. Furthermore, a way of allowing an application user to answer questions via the interface is also necessary. When a question is loaded, the application will
decide which control to add to the user interface based upon the expected data type (the answer) of the question. For example, if the question being loaded requires an answer in the form of text then the application will generate a textbox control for the user to provide an answer.

### 3.4.3 Report Processing

When a Q&A Session has been completed by a user, a report will need to be generated to allow the users to view what questions have answered during that session. The report will be generated by an SQL script which will extract the necessary data from the database. Further to this, the client (application administrator) has requested functionality to run administrative reports to find out users answers for a particular Q&A Session.

### 3.4.4 Required Inputs and Outputs of the System

The inputs for the system will include the questions, entered by the application administrator and the answers entered by the limited application users. Further input will be in the form of adding users and editing user accounts, again made by the application administrator. The primary output from the system will be the questions, to which limited application users provide answers. Depending upon the answers, questions will be output from the database, until the Q&A session is complete. This stage will be signalled by the creation of a report, as mentioned in section 3.4.3. As a further form of output the generated report may then be printed.

### 3.5 Non-Functional Requirements

#### 3.5.1 Performance criteria

As the proposed solution is not an application which is being marketed or sold, there is no need to discuss the implementation of Service Level Agreements (SLAs). However the system should provide the user with appropriate response times, certainly when inserting, updating or retrieving data from the database.

#### 3.5.2 Help Considerations

As part of the proposed solution there is a requirement for a help feature, to allow limited application users to find out more information regarding the question they are attempting to answer. This could be
a simple label on the screen containing text explaining to the user what the question is trying to achieve and what their answer should contain. In doing so it will reduce the likelihood of vague answers, which is what the proposed solution is trying to eradicate.

### 3.5.3 Security considerations

As a standard measure, security will be enabled within the system, requiring application users to provide credentials when logging in. Furthermore SQL stored procedures will be used to reduce the risk of SQL injections within input regions and finally user details will be hashed within the database, to provide protection at the data access layer.

### 3.6 Usability Requirements

#### 3.6.1 Nielsens Usability Heuristics

In accordance with standard design guidelines, Nielsens Usability Heuristics will be considered when designing the applications interface. Nielsen outlines ten usability principles, each of which should be followed when designing user interfaces [21]. During the evaluation phase of the project, the usability principles will be used as the basis for evaluating the interface.

#### 3.6.2 Application Users tasks and goals whilst using the system

Tasks and goals will differ within the system depending upon user type. Limited application users will have the task of firstly logging in then starting a Q&A Session (for a CH Project), during which they will answer questions by inputting their answer via a dynamically created control displayed on the form. Completion of a Q&A Session will be marked by the creation of a report, listing all the questions and answers provided. The Application administrators’ main tasks will be to log in and carry out general administration work, with goal of either successfully inserting, updating or deleting records relating to questions and users in the database. The application administrator will also be able to generate reports from a list of Q&A Sessions, with the goal of viewing and printing the report if necessary.
Chapter 4

Design of Proposed Solution

This chapter addresses the design phase of the project, to satisfy the third workflow of the USDP. In conjunction with Chapter 3, the remaining work required for the elaboration phase of the USDP development process is completed. The chapter discusses in detail the design iterations of the requirements elicitation algorithm followed by the database, user interface and navigational designs.

4.1 Deployment Modelling

In accordance with the USDP, a UML modelling exercise took place in order to understand the business workflows and interactions between the stakeholders and the proposed system (all diagrams can be found in Appendix B). The UML implementation diagram (Figure B.1) shows how the system would be deployed from an architectural point of view. As the figure indicates, a 3-tier architecture will be adopted, with the application logic layer residing on a web server running Internet Information Services® and the .NET Framework 2.0®. Lastly the data access components will reside on a database server running Microsoft SQL Server 2005®. The application will be developed locally and uploaded onto the web server for testing and evaluation purposes.
4.2 Elicitation Algorithm Design

As discussed the focal point of the project is selecting a question based upon a users previous answer. In order for such functionality to be realised a method of ascertaining the next plausible question needs to be developed, this will be referred to as the “elicitation algorithm”. In keeping with the traditional views of the USDP outlined by Bennett et al as “an iterative approach within four main phases” [9] the proceeding sections address three design iterations of the algorithm. The elicitation algorithm is to be implemented within the database through the use of SQL as the overall performance of the system will benefit given the reduced number of calls to the server. Furthermore, the entire web application will be far more flexible, since the algorithm could then be applied to any domain without the need for specific changes. Before the algorithms outlined below are invoked, a Q&A Session should be created by the answering individual by selecting the Project for which they are affiliated with along with the types of questions they wish to answer (Operational or Technical).

4.2.1 First Iteration

The initial design of the algorithm employed the use of successors and predecessors to determine which question should be asked next given an answer to a previous question. A successor can be defined as the next eligible question that can be asked given the answer to the current question. Conversely a predecessor determines whether an arbitrary question can be asked based on the answer to a previous question.

1. Retrieve all questions where all predecessors have been fulfilled.
2. From (1), select questions with the most predecessors.
3. From (2), select questions with fewest successors.
4. If there is more than one possibility in (3), pick the question with the lowest position.

In this iteration, the algorithm is divided into four main phases, outlined above. Its purpose is to find a single yet plausible question to ask the user. The first phase retrieves all the questions where the predecessors have been satisfied (i.e. a question whose pre-condition has been met due to a previous answer). If this phase yields more than one possible question (i.e. there are many questions whose pre-conditions have been satisfied), the second phase is employed. Phase two selects questions from the first
phase that have the highest number of satisfied predecessors. In performing this action the algorithm is attempting to find a set of the most relevant questions, indicating a reliability on the premise that the question with the highest number of fulfilled predecessors has to be the most eligible. The third phase attempts to narrow the number of eligible questions further by selecting those from phase two that have the fewest or no successors. Implementing phase three at this point in the algorithm will theoretically pick such questions towards the end of a Q&A Session, as the second phase will exhaust all questions with predecessors first, thus reducing the number of questions to ask. The final phase in the algorithm selects the question with the lowest position number (relative to the question list) but is only employed if phase three yields more than one question. By performing this step it ensures that a reasonable order is maintained even if there appears to be no eligible question. Furthermore, the prioritisation technique of ordering the questions should be defined by the system administrator, who designs and creates the survey.

With the aid of Levitin [18], the above elicitation algorithm has a calculated worst case time complexity of $O(n^2)$, due to the fact that all predecessors may well be satisfied by an answer and therefore all questions should be asked.

It is important to note at this point that the criteria used for phases two and three could be altered, thereby assessing the fewest predecessors or the most successors. Such a change in criteria seems more logical for the early stages of a Q&A Session, as it would mean that the algorithm deals with such questions at a time during the Q&A Session when few predecessors would have been fulfilled anyway and as a result could have the potential of optimising the process.

### 4.2.2 Second Iteration

In designing the second iteration of the algorithm, the implementation of the database was taken into greater consideration as was the general domain for which the application was being built. The ideas behind the phases outlined in the previous iteration were used, however it was found that the use of successors in the third phase was unnecessary and that using predecessors independently of successors makes the algorithm simpler and easier to implement.

1. Retrieve all questions where all predecessors have been fulfilled.

2. From (1), select questions with the most predecessors.

3. From (2), select and order the questions (lowest to highest) that have not already been answered.
4. From (3), select the first question in the list.

The algorithm above starts as before, by selecting all questions that have had predecessors fulfilled. Phase two selects questions from phase one that have the most fulfilled predecessors, implying they are the most plausible to ask. The algorithm then proceeds to phase three, to select questions from phase two that have not already been answered and arranges them in ascending order, to ensure a single question is not answered twice. Most importantly the questions selected are arranged in ascending order, resulting in the most eligible question positioned towards the top of the list. The final phase selects the top question in the list as the next question to ask. Again this strategy ensures that question ordering is maintained and in doing so provides a transparent feel of continuity during a Q&A Session.

4.2.3 Third Iteration

The third and final iteration for the design of the algorithm was based largely on the second. Further enhancements were made by taking into account the clients requirements for the project. As mentioned in Chapter 3, a requirement of the application is to distinguish between two different types of users, with regards to the Child Health system. These Child Health users are, ‘Operational Users’ and ‘Technical Users’. With regards to the proposed application, each of these user types should only answer questions related to their field of endeavour, as asking an Operational user a technical question risks alienating the user and vice versa. A further requirement of the application is that CH Project names should be used to identify a Q&A Session. By taking into account the above requirements, the following algorithm was designed.

1. Retrieve all questions where all predecessors have been fulfilled.

2. From (1), select questions with the most predecessors.

3. From (2), select the questions that have not already been answered for the Project in question.

4. From (3), select and order the questions (lowest to highest), that have the required user type (Operational or Technical).

5. From (4), select the first question in the list.

From the algorithm above, an additional phase has been added to what was the second iteration. Phase three not only selects questions that have not already been answered but ensures that the question has not been answered for the CH Project in question, as there is potential for many different CH
Projects within the system and hence many answers to the same question. Furthermore, as an additional predecessor to a question, the CH User Type is considered in the third phase, to select questions defined for that user type (Operational or Technical). It is at this point that the questions are arranged in ascending order, before the top question in the ordered list is chosen as the question to ask.

4.3 Data Access Layer

By considering the elicitation algorithm design (above) it is possible to design the database for which the user details, question details and answers would be stored. As the elicitation algorithm is designed to be implemented using SQL stored procedures, the database should be built around the needs of the algorithm but at the same time maintaining the principles of legal database design.

4.3.1 Entity Relationship Diagram

Elmasri & Navathe indicate that designing the entity relationship diagram should take place after the requirements capture and analysis phase [12]. Given that this condition has been satisfied in Chapter 3, entity relationship diagrams have been produced (Figures C.2 and C.3, Appendix C). The diagrams illustrate the relationships that exist between entities. An entity is described by Elmasri & Navathe as “a thing in the real world with an independent existence” [12] so with regards to the proposed application, an entity could refer to a Project, Question, Answer or User. Given the requirement for two types of users (an Administrator and Limited User), it was deemed necessary to design two entity relationship diagrams to illustrate the different associations between the two user types and other entities. The relationships between entities are regarded as either 1:1 (one-to-one), 1:N (one-to-many), M:N (many-to-many), which basically define the constraints placed on entities within relations.

4.3.2 Database Schema

The database schema below reflects the entity relationship diagram above and is suitable for storing questions, requirements information and user details. Further to the list below, ASP.NET creates its own set of table as part of the Forms Authentication technique, discussed in section 2.3. Such tables are responsible for managing users details and from initial research, appear to have the prefix ‘ASP.NET.’ Given this naming convention it seemed appropriate to provide the prefix, ‘ELICATION.’ to distinguish the tables created by the developer.
The tables above are necessary for a fully customisable requirements elicitation system. The ELICITATION_QUESTION table will store the question information, with predecessors being stored in the ELICITATION_PREDECESSOR table, thus allowing for many predecessors to be assigned to one question. The Value field in the predecessor table is the intrinsic pre-condition which is compared to the users answer to check whether a predecessor is satisfied. The QuestionOrder field within ELICITATION_QUESTION will order the questions independently of the QuestionID and allow for easy re-ordering. To track which predecessors have been satisfied given an answer, the table ELICITATION_PREDECESSORSSATISFIED has been designed. The ELICITATION_QUESTIONTYPE table holds the terms used to identify the type of question (Operational or Technical), as per the requirement outlined in Chapter 3. The Project names for which requirements elicitation takes place are stored in the ELICITATION_PROJECT table with the answers being stored in the ELICITATION_ANSWER table.

As outlined in Chapter 3, questions can be asked in any order and therefore a method of loading form
controls onto a web page dynamically is required (second minimum requirement). Such functionality can be achieved by storing the required control in the ELICITATION\_CONTROL table, which provides as a foreign key into the ELICITATION\_QUESTION table. Furthermore, if the question requires answer options (such as Yes or No), each are stored within the ELICITATION\_ANSWEROPTION table and due to the relational constraint with the ELICITATION\_QUESTION table, many answer options can be stored for one question. The ELICITATION\_OPERAND table will store the different types of operators (<, >, =, ! =) available when setting a predecessor for a question. As a primitive yet effective method of ensuring the correct data type has been entered for a question, the ELICITATION\_DATATYPE table will hold the data type required for a question (providing a form of validation). To facilitate the storage of help text for a question, the ELICITATION\_HELP table has been proposed.

4.4 Presentation Layer

This section discusses the design of the interface and navigational components. After initial designs were prototyped, a meeting was scheduled with the client (indicated in the project plan) in order to sign-off the proposed user interface.

4.4.1 Pages

In order to fulfil the minimum requirements outlined in section 1.4, a set of web pages will be required to allow for user interaction. To enable users to login to the application using supplied credentials a login page must be implemented. Conversely to facilitate logging out of the application, a logout page is necessary. After successfully logging in, a home page will provide succinct yet useful information regarding the purpose of the application. Furthermore Q&A Sessions will be initiated here. To facilitate user, question and project management pages must be implemented to allow the administrator to perform the tasks outlined in section 3.3.2. To allow users to start answering a survey, a page will be required to generate dynamic form controls (such as labels, buttons and textboxes etc) required for a question. To fulfil the third minimum requirement, reporting pages will be implemented to allow administrators and users to generate reports for particular Q&A Sessions and export the results if necessary.
4.4.2 User Interface

Designing the interface for an application is seen as a crucial stage in the development process. Schniederman & Plaisant state that interface design has a vast scope of potential from creating success stories in business to being able to change the way professionals operate [23].

Whilst designing the user interface, relevant Nielsen usability heuristics have been used. Nielsen notes that information displayed to the user must be relevant [21]. Given this, the proposed design attempts to minimise the amount of unnecessary information commonly found within labels and message boxes. In doing so, this satisfies another Nielsen heuristic which proposes to reduce the amount of information a user has to remember [21], thereby having users only remember what’s important. Furthermore, the information fed back to the user will be consistent and understandable, as providing non-descriptive or technical terms risks alienating the users and resulting in a non-user friendly interface. Nielsen further states that a system may contain a help feature which users can refer to [21]. This in itself is a requirement of the project (see section 3.5.2). Help text will be inputted by the survey creator, for a question that is being added into the system, which a user may then refer to when answering a question. Nielsen also promotes the correct use of error messages in systems [21], which will be generated via JavaScript functions as a result of validating forms before submission. The messages themselves will be clear, descriptive and to the point. However when users are answering questions, the usability may be affected by the grammatical construction of a question and so administrators should avoid the use pronouns and adjectives when specifying questions. Unfortunately, without going down a natural language processing (NLP) route (beyond the scope of the project) this problem will persist.

The appropriate use of colour must also be considered when designing a user interface. Nielsen outlines three guidelines as follows, firstly, use a set number of colours, secondly consider people who are colourblind and lastly, use colour to group, not present information [21]. Therefore, by using at most three shades of blue for grouping information on the web pages but using black for textual areas (shown by Figure C.1, Appendix C), the web site will be consistent and easy to view, without compromising the three principles above. Accessibility of a web site is an issue that Holzschlag discusses, noting that any individual should be able to access a web site, regardless of platform, browser, physical location or physical disability [14]. To ensure the application is easily accessible and that functionality does not undergo significant alterations, testing will occur within the web browsers; Mozilla Firefox, Opera and Microsoft Internet Explorer due to their popularity within the web community.

The design shown by Figure C.1 illustrates a general template that can be applied across all the
web pages, thereby characterising a strive for consistency and conformity. At the same time however, it attempts provide a simple yet attractive interface through the use of browser safe colours. Within ASP.NET, Liberty & Hurwitz [19] explain that consistency is achieved through the use of Master Pages. As such these will be implemented in conjunction with CSS.

4.4.3 Navigation

Navigating through the web site will be achieved through hierarchical menus. Holzschlag notes that by positioning the menu at the top of the page, more room is left in the centre for the content [14] and since there will only be a maximum of 10 pages in the web site, that in itself does not justify the implementation of a vertical menu, which may require more space. Nielsen notes that a trade off exists in hierarchical menus between broad yet detailed menus and deeply nested menus with high navigational times [21]. By considering these points, Figure 4.1 proposes a hierarchical menu, which groups nodes together based on the task and in doing so should yield fast user navigational times.

4.4.4 Dynamic Form Controls

A method to create dynamic form controls is necessary as each question will require various forms of input from the user, such as booleans, strings, integers or floating points. In section 4.3.2 a database table has been proposed to store the control required for a question, which can then be retrieved and created dynamically on the web form via a program class that creates instances of such controls. Furthermore dynamic event handlers will also be implemented to manage the click event on the button that proceeds to the next question.
Chapter 5

Implementation

The following chapter discusses the implementation of the web application and addresses the fourth workflow of the USDP. Furthermore, the majority of the USDP construction phase is also fulfilled at this point. The chapter is structured to enable the implementation to be discussed in terms of two iterations. The first iteration deals largely with fulfilling the minimum requirements and the final iteration explains the optional enhancements made.

5.1 First Iteration

5.1.1 Data Access Layer

The database was deployed on an internal School of Computing web server. The tables outlined in section 4.3.2 were fully implemented indicated by Figure D.37, Appendix D. However the Question-TypeID field, originally found within the ELICITATION_QUESTION table was modified as it seemed more logical to include this field within the ELICITATION_PREDECESSOR table and have it act as a joint predecessor, rather than attribute of a question. As mentioned in section 4.3.2, user accounts are managed through a set of pre-defined ASP.NET tables. These tables were set up in Microsoft SQL Server 2005© by running the command aspnet_regsql.exe at the SQL Services command prompt. Security measures were implemented, with specifics discussed in section 5.1.1.2. SQL stored procedures
were also written to facilitate the elicitation algorithm extraction of other necessary data required by the first iteration of the application.

5.1.1.1 Elicitation Algorithm Implementation

The algorithm derived in section 4.2 was implemented as part of four SQL stored procedures, explained below. Whilst explaining the queries, the algorithm derived in section 4.2 will be referenced to allow the stages of the elicitation algorithm to be mapped with the stages of the procedures.

The *CreateSession* procedure, although not formally part of elicitation algorithm, is still an important stage with the purpose of creating a new *Q&A session* for a given *Project* name. The query starts by copying predecessors from the ELICITATION_PREDECESSOR table to the ELICITATION_PREDECESSORSSATISFIED table for a given *Project* and *Question Type* and finds the *PredecessorID* for the opening question and sets its *Satisfied* column (of the predecessors satisfied table) to ’1’ (fulfilled) as the opening question must be satisfied in order for the survey to begin. The remaining tuples within that column are set to ’NULL’ to indicate that no other predecessors have yet been fulfilled. Then as predecessors are satisfied, tuples within this table are updated accordingly. No difficulties were encountered at this stage however a design decision had to be made as to whether all *predecessorIDs* were inserted into the ELICITATION_PREDECESSORSSATISFIED table or whether just the predecessors for the selected *Question Type* were inserted. Following further analysis, a decision was made to insert predecessors of a given *Question Type*. In doing so, it prevents complications from occurring when for example a ’technical’ question has the predecessor of an ’operational’ question and ensures only predecessors of the selected *Question Type* are marked as satisfied. Furthermore as the number of questions and predecessors increases, there may be a performance overhead if the query is operating on predecessors which are not of the selected *Question Type*.

With reference to the phases of the elicitation algorithm and the *GetFirstQuestion* procedure below in Figure 5.1, phase 1 is carried out on line 42, by selecting predecessors that have been satisfied (i.e. not ’NULL’). Implementation of phase 2 was attempted to allow for mixed disjunctive and conjunctive pre-conditions. However, due to difficulties in finding the satisfied predecessors and time constraints, phase 2 was not implemented and thereby the resulting elicitation algorithm supports only disjunctive pre-conditions. Phase 3 of the algorithm is implemented on lines 40 and 43-45, selecting questions that have not already been asked for the *Project*. Phase 4 is achieved in lines 41 and 46, by selecting questions of a given *Question Type*, then arranging the list in ascending order using the *QuestionOrder*
field. Finally, phase 5 of the algorithm is satisfied by line 31, which selects the top question (and therefore the most plausible) in the list from phase 4.

The **InsertAnswer** procedure is broken down into three phases to insert a user's answer. The first of which inserts the answer provided by the user into the ELICITATION_ANSWER table. The second phase then attempts to find the predecessors which have been satisfied by the answer and sets them to '1' (fulfilled) in the ELICITATION_PREDECESSORS_SATISFIED table, illustrated in 5.2.

Lines 49 to 53 (Figure 5.2) check whether any predecessors are satisfied, by comparing the answer and the pre-conditions `value`. If true, then an update statement is invoked (lines 55–61) and the corresponding predecessors in the ELICITATION_PREDECESSORS_SATISFIED table are set to 1 (fulfilled), for the `Project`. It is impor-
tant to note at this point that if many predecessors are satisfied by the answer, the ‘IN’ statement (line 56–61) will allow all appropriate predecessors to be set to ‘1’. In addition to the SQL snippet shown in figure 5.2 which finds predecessors where the users answer is less than the specified predecessor value, similar code exists to satisfy predecessors where the answer is =, != or > the predecessor value. The final phase of the query, calls the GetNextQuestion (below) procedure to select the next question, given the satisfied predecessors.

Figure 5.3: SQL snippet: GetNextQuestion

As can be seen, the statement in figure 5.3 is similar to that of figure 5.1 but instead checks that the QuestionID being selected is not equal to the current QuestionID. The constraint was put in place to ensure that the same question is not asked twice in succession. In hindsight however GetFirstQuestion and GetNextQuestion could have been implemented as one procedure, therefore simplifying the implementation and promoting the re-use of code.

To enable the generation of reports containing questions and answers provided for a particular Project, the getAnswersForReport query was created (Figure D.23, Appendix D).

5.1.1.2 Security

All database queries used by the web application were implemented as SQL stored procedures to reduce the likelihood of a successful SQL injection attack, as mentioned in section 2.2.4.2. Although Howard et al points out that using stored procedures does not completely eradicate the risks posed by SQL
injections [16], they provide a basic level of security. The procedures themselves will be tested by injecting SQL into points of user input to ensure there are no adverse affects. A further security measure was taken in using the ASP.NET built in Membership and Role providers to manage user details securely, however in using these providers they inherently require ASP.NET tables within the database (created using the method described in section 5.1.1). When users accounts are created for the web application, the Membership and Role providers are invoked to add the user and hash their credentials within the database, shown by Figure D.39 in Appendix D.

5.1.2 Application Logic Layer

5.1.2.1 Implemented Program Methods

C# was the programming language used to code the application logic layer. A diagram illustrating the full set of classes and methods implemented is shown by Figure D.38 in Appendix D, created in Microsoft Visual Studio 2005®. The 'ElicitationDAL.cs' class was written to permit communication between the application logic layer and the data access layer, by opening and closing database connections as well as executing stored procedures and returning results. The class methods below (found in Appendix D) are used when adding questions to the database via the UserAdministration.aspx page.

- **InsertNewQuestion**: (Figure D.24). After establishing a connection to the database, it creates an 'SqlCommand' object, appending the necessary parameters and calls the `insertNewQuestion` stored procedure to add a new question.

- **InsertNewPredecessor**: (Figure D.25). Upon passing in the appropriate parameters the `insertNewPredecessor` procedure is executed to add pre-condition(s). Multiple disjunctive pre-conditions are inserted by looping through the list of specified predecessor.

- **InsertNewAnswerOption**: (Figure D.26). Answer options are displayed to the user as options to a question. After being defined by the survey creator for a question, the `insertNewAnswerOption` procedure is called to insert the specified options.

- **InsertHelpText**: Help text (discussed in section 3.5.2) is specified by the administrator at survey design time. When called, the method invokes the `insertHelpText` query to insert the help text for the question.

The following C# methods allow users to answer a survey.
- **CreateSession():** (Figure D.27). Invoking this method will call the `createSession` stored procedure and pass the parameters provided (`ProjectID` and `QuestionTypeID`) to the method, thus creating a new Q&A Session.

- **GetFirstQuestion():** (Figure D.28). A method to return the opening question to ask the user by invoking the `getFirstQuestion` stored procedure.

- **InsertAnswer():** (Figure D.29). Calls the `insertAnswer` query to insert an answer for a particular question and `Project` into the database.

- **GetNextQuestion():** Using this method, the next plausible question is retrieved from the database (given an answer), by calling the `getNextQuestion` query.

- **GetAnswerOptions():** The method passes the `QuestionID` parameter to the `getAnswerOption` query which then returns a set of options for that question.

### 5.1.2.2 Dynamic Controls

Dynamic controls were achieved by implementing the `ControlCreator.cs` class (a snippet of which is found in Figure D.30 Appendix D) to instantiate and return form controls on the `QuestionAnswerSession.aspx` page. The controls are used to display question information and allow users to input answers. Figure D.30 shows an example of a method to create a `Label` control, at which point all properties of the label are set. After the user submits an answer, the `GetNextQuestion` query returns the next question along with the name of the control required, as specified by the creator of the question. A table is then built up to present the controls to the user, explained in section 5.1.3.1. The controls created dynamically include `Labels`, `Textboxes`, `DropDownLists` and `Buttons`. `RadioButtons` were experimented with however due to technical issues and time constraints, they were not fully implemented.

### 5.1.2.3 Reports

Functionality to generate reports containing questions and answers for a `Project` is managed via the `Reports.aspx` page, on which a `DropDownList` is populated with all the `Projects` where answers have been provided. From here, the administrator can generate reports by selecting a `Project`. Results from the query are passed to the ASP.NET `GridView` control for presentation, which was used over a regular
XHTML table as it is capable of building dynamic tables, independent of the number of columns returned by the query, thus satisfying the third minimum requirement. Furthermore the MyReports.aspx page allows the answering user to review the answers they have provided.

5.1.2.4 Pages

The pages outlined in this section were implemented as part of the first iteration but have not yet been addressed in the chapter. All the implemented pages can be found in Appendix D. To ensure pages are not accessed by unauthorised users the web.config file contains settings that define user Roles (discussed in section 2.3.2) and privileges. Figure 5.4 illustrates a permission being set where only the administrator can view the User Administration.aspx page.

```
<location path="pages/UserAdministration.aspx">  
  <system.web>  
    <authorization>  
      <allow roles="Administrator="/>
      <deny users="*"/>  
    </authorization>  
  </system.web>  
</location>
```

The Default.aspx and Logout.aspx pages allow a user to log in and out of the web application, respectively. Functionality to remember the user after login was also included as a later requirement, as was the page QuestionsWithPreConditions.aspx which allows all questions and pre-conditions to be viewed on screen. The UserAdministration.aspx page enables administrators to add or delete user accounts as well as change user passwords. Add Question.aspx allows an administrator (survey creator) to add questions along with help text, answer options and predecessors for a question. Q&A Sessions are created via the Welcome.aspx page, which provides introductory information regarding the web application and allows a user to select a Project and their chosen CH User Type (the ‘Question Type’) for requirements elicitation. Further to the pages above, in order to satisfy the remaining ‘Should have’ use cases, the page Add Project.aspx was implemented to allow new Projects to be added into the database for which requirements elicitation needs to take place.

Largely there were no major issues encountered during the implementation of the web pages. However the issue of posting the QuestionAnswerSession.aspx page to the web server became a problem, since the load event of the page contained a method to retrieve the opening question. This resulted in the GetFirstQuestion() method being invoked each time an answer was submitted and the page re-
freshed. The problem was mitigated using the following code, which retrieves the opening question if the page is loaded for the first time.

```csharp
if (!IsPostBack)
{
    GetFirstQuestion();
    lnkHelp.Text = "Show Help";
}
else
{
    GetNextQuestion();
}
```

### 5.1.2.5 Security

Security at this layer within the 3-tier architecture is managed and configured within the `web.config` file, using XML (eXtensible Markup Language). Here the connection string to the database is declared, as are the definitions and connection strings for the Membership and Role providers (shown by Figure D.33, Appendix D). Furthermore the developer was able to configure the attributes which state the conditions that must be satisfied for a password, before a user account is created (Figure D.33, lines 116 to 129), thus reducing the quantity of code required from the developer. The state of the database connection is managed in the `ElicitationDAL.cs` class in accordance with Howard et al who emphasises the point that a database connection should never be constantly active [16]. This can be seen in Figure D.28 Appendix D, lines 854 and 879 where connections are only opened and closed when required.

### 5.1.3 Presentation Layer

#### 5.1.3.1 User Interface

After considering the usability and design principles outlined by Nielsen [21] and Holzschlag [14], the user interface design proposed in section 4.4.2 was implemented. The consistent style is not only characterised by the implementation of Cascading Style Sheets (a snippet from which can be found in Figure D.36, Appendix D) but also with the utilisation of ASP.NET Master Pages. A single Master Page (the code of which can be found in Figure D.31, Appendix D) is used to display the banner and menu across all web pages, thus providing for a coherent look and feel. The Master Page itself contains four `ContentPlaceHolder` controls each of which displays relevant content to the user.

The creation of dynamic controls to facilitate requirements elicitation proved to be a challenge at this stage. As controls were required to be created on-the-fly, a reliable yet flexible method was re-
quired in order for controls to be positioned correctly on the form. Experimentation took place using dynamic Cascading Style Sheets but resulted in application logic code being mixed with unstructured presentational markup. At this stage, it was decided to use the ASP.NET Table object, which contains methods to append Cell, Row and Column objects to a table dynamically. A method called CreateComponentTable() (see figure D.32, Appendix D) was implemented to build a dynamic table containing the necessary controls. After a question is answered, the table disposes itself to remove any old controls from the previous question and adds three new Row objects to the table, primed for the next question and response. The top is used to contain the question, the second contains the dynamically generated controls required for the answer and the third row holds the Submit and Exit buttons (also created dynamically). Using the results passed back by the GetNextQuestion query, the method invokes a particular function (an example of which can be found in Figure D.34 in Appendix D) to return a Cell object containing the required control, (the control of which is created using the 'ControlCreator.cs' class, see section 5.1.2.2). The process continues until all the necessary controls are added into the table to facilitate requirements elicitation, therefore satisfying the second minimum requirement. In doing so, event handlers are also added to the created Button controls. If an incorrect data type is entered for an answer JavaScript is used to alert the user.

5.1.3.2 Navigation

The navigational structure outlined in section 4.4.3 was implemented using an ASP.NET web.sitemap file (Figure D.35, Appendix D). Within this file, the menu structure is defined within nodes, using XML. The menu itself is laid horizontally, allowing for more room on the page and hence more controls. As the mouse cursor hovers over a root node, the node expands to display to all its child nodes. As the mouse is moved away, it takes 500 milliseconds for the expanded node to disappear.

5.2 Second Iteration

At this stage in the project, all the minimum requirements as well as the 'Must have' and 'Should have' use cases had been fulfilled. The second iteration saw the introduction of features that were regarded as optional and hence were categorised as 'Could haves' and 'Won’t haves’ in the MoSCoW prioritisation technique (Figure B.1 Appendix B). However due to time constraints, only the Delete Project ‘Could have’ use case was implemented, to allow an administrator to delete a Project.
Chapter 6

Testing

The following chapter addresses the Testing workflow of the USDP and in doing so concludes the work required to fulfil the USDP development process. The purpose of Testing is to ensure that the use cases outlined in Appendix B function correctly. As per the deployment model outlined in section 4.1, functional testing will take place via the internet, to simulate how potential users access the web application. Furthermore, in accordance with standard testing guidelines, the IEEE Std 829-1998 (IEEE Standard for Software Test Documentation) paper [22] has been used as a basis for test planning.

6.1 Test Plan

By using the guidelines proposed by the IEEE Std 829-1998 [22], test plans have been devised in Appendix E. Bennett et al [9] suggests that the use cases derived in the Requirements and Analysis workflows should provide the basis for testing. Given this, the use cases defined in Figures B.4 and B.5 in Appendix B, have been added to a table of test items (Figures E.1 to E.5). As can be seen each test item is split into a series of sub-tests to ensure that elements of the application operate correctly under different inputs and actions. The test plans themselves contain columns suggested by the IEEE Std 829-1998 paper [22]. The first column uniquely identifies the test item and the second provides it with a name. The third and fourth columns state the required inputs and outputs from the tests carried
out, with the fifth, sixth and seventh columns listing any actions that need fulfilling prior to the tests being carried out. Finally, results are explained in the last column.

6.2 Summary of Test Results

All tests were carried out by the developer and client, with results noted and highlighted in red, as indicated in Appendix E. Overall, tests carried out on the use cases were successful, mainly due to the ongoing testing performed during the implementation. The proceeding sections provide more specific testing details regarding the aim of the application:- to produce a requirements elicitation tool. Further test items can be found in Appendix E.

6.2.1 Creating a Survey

The Add Question use case was tested here, results are found in Figure E.2, Test Item ID: U8. This test item was expanded on by adding a set of ten questions (provided by the client and found in Appendix B) into the database, via the AddQuestion.aspx page. Findings are discussed in Figures E.6 to E.8. From the results, the client found the same predecessor could be entered twice for one question, which although has no impact on the algorithm, wastes space in the database. This flaw has been amended.

6.2.2 Elicitation Algorithm

In testing the elicitation algorithm, a set of ten questions (entered as part of the previous test) were answered. In doing so, the Submit Answer use case was being tested (Figure E.3, Test Case ID: U10). Various answers were provided over multiple Q&A Sessions to determine whether predecessors were being satisfied and hence the correct question was being displayed. As can be seen from the results in Figures E.9 to E.11, the elicitation algorithm worked as desired even with adverse conditions. A minor bug did arise in Test Case ID A3 (Figure E.9) regarding operational and technical questions with mixed predecessors, which although has now been resolved does provide a basis for further research.

6.2.3 Reports Generation

The testing process for the Generate Report use case involved the generation of a report for one of the Q&A Sessions completed in the previous section (Test Case ID U12). As a further test, the report was exported to a spreadsheet application installed locally, to satisfy the Export use case (Test Case ID U14)
and to ensure there were no platform issues. As can be seen from the results in Figure E.4, the tests performed were successful.

6.2.4 Security

In testing the security of the web site, valid and invalid user credentials were supplied (Figure E.1 Test Case IDs U1 and U2). As noted in the results, the web site did not allow any unauthorised access. Test Case ID U1 simulated an unsuccessful SQL injection attack, in an attempt to access the web site using invalid login credentials. Test Case ID U3 found in Figure E.1 attempted to add a user with a weak password which as the results indicate, was not accepted.

6.2.5 Web Browser Testing

To ensure the consistency of the web site was maintained, each page was tested in the three web browsers outlined in section 4.4.2. Results showed that there was little difference across each browser. The layout and positioning of text and controls was identical, characterised by the use of CSS and ASP.NET Master Pages. Apart from small variations in font sizes and the styles applied to the controls, consistency was largely maintained throughout. Furthermore, validation techniques through the use of JavaScript also functioned correctly across the browsers.

6.3 User Acceptance Testing

Given that Nielsen points out that “designers are not users” [21], it was necessary for the client and other random individuals to be invited to test the application to ensure that all initial requirements captured in Chapter 3 were met, as were the usability considerations. To provide structure, a test plan was devised with a set of tasks the users had to complete alongside the developer, who noted any feedback. From a usability perspective, the users were able to navigate and interact with the system well, with little prompting required from the developer. In terms of functionality, the client made recommendations which have now been implemented. A set of findings are presented in Figures E.12 to E.16.
Chapter 7

Evaluation

In evaluating the project, the initial objectives and requirements are discussed and the user interface and navigational aspects are subjected to the Nielsen Usability Heuristics. Furthermore, the developed application is evaluated against the clients current solution with tests carried out on the algorithm to determine its success. Finally the limitations are discussed and possible enhancements are suggested for future developments.

7.1 Schedule

The project schedule (found in Appendix G) was largely followed but alterations occurred due to the work required from other University modules. Most significantly, by late January the project was delayed as a result of the examination period which as a consequence meant later completion dates for the stages following System Design. This decision was taken to allow the project to return to schedule through the use of attainable dates which, as a result, did not compromise the projects delivery. The project schedule was structured largely around the USDP. The Requirements Capture stage followed by the Requirements Analysis stage allowed for a deeper understanding of the problem through a meeting with the client and further research. Both phases proved successful in not only capturing the issues specific to the client but also in understanding the general problem of selecting questions from a set of
satisfied pre-conditions. The *System Design* stage stepped through a series of design iterations resulting in the final elicitation algorithm. In developing the algorithm in such an incremental fashion, it led to the most appropriate and efficient solution to come to light. A further meeting was arranged with the client to sign-off the system design before progressing to the next phase. During the *Implementation* stage issues surrounding the development of the algorithm and the use of dynamic controls were apparent. However, these were resolved and the stage was completed ahead of schedule, allowing for *Testing* to start earlier. The feedback gained through client testing was positive with further recommendations made and implemented. On a whole, the initial structure of the project plan was followed but as a result of changing circumstances the schedule had to adapt in order satisfy the workload required.

### 7.2 Meeting Objectives

As a method of evaluating the project, this section outlines how the original objectives outlined in Chapter 1 have been fulfilled.

- *Analyse the current system in place for data migration within CSC.*: Fulfilled. Research into current business practices and the requirements elicitation system was carried out in Chapter 3 through an extensive requirements specification with the aid of UML (found in Appendix B).

- *Decide upon the system architecture (standalone or web based) based upon clients requirements.*: Fulfilled. Chapter 3 addresses the issue of the application architecture based upon the clients requirements and in conjunction with Chapter 4 the proposal for a web based system was put forward.

- *Research similar systems in place, methodologies and technologies that can be used for developing the system.*: Fulfilled. Current applications such as the Bristol Online Survey [2] and SurveyMonkey [5] have been critically analysed, as have proposed techniques put forward by Belani et al [8], in Chapter 3. Research into software development methodologies as well the available technologies have been discussed in Chapter 2.

- *Produce a design for the system.*: Fulfilled. A design for the proposed solution has been outlined in Chapter 4.

- *Build the system using the chosen technology to meet the minimum requirements.*: Fulfilled. The
Unified Software Development Methodology (USDP) has been used accordingly throughout the development of the solution as illustrated by the project schedule and structure of the report.

- *Test the system with suitable test data to highlight any problems.*: Fulfilled. Chapter 6 makes reference to a set of test plans found in Appendix E to standardise the testing phase, using questions provided by the client.

- *Evaluate the solution based upon the clients requirements.*: Fulfilled. The current chapter addresses this objective.

### 7.3 Meeting Requirements

The minimum requirements listed in Chapter 1 have all been delivered (and exceeded) as indicated below. Furthermore, as part of the USDP methodology and in conjunction with the MoSCoW task prioritisation technique further requirements captured in Chapter 3 have been implemented.

- *Implementation of a database suitable for storing question sets, requirements information and user details.*: Delivered. A database was designed and implemented to facilitate requirements elicitation with the ability to store questions, users answers and user login details securely.

- *Functionality for generating dynamic web forms for requirements elicitation.*: Delivered. Functionality to allow the generation of dynamic form controls for requirements elicitation was implemented as part of the application logic layer using ASP.NET and C#.

- *Functionality for generating reports based on the requirements information that has been gathered using the system.*: Delivered. Reports displaying users questions and answers provided for a Q&A Session, for which requirements elicitation data is gathered can be generated and exported by either a limited application user or an administrator.

### 7.4 Usability

During the design of the user interface, the Nielsen Usability Heuristics were considered to ensure that the interface conformed to standard guidelines. To evaluate the interface and navigational components, the web application was subjected to these heuristics by selecting ten users at random, to perform certain
tasks on the system (found in Figures E.12 to E.16). The tasks were simple yet designed to mirror the
day-to-day usage of the application. Upon completion, users were asked questions in relation to how
the web application satisfied the relevant Nielsen Usability Heuristics. Furthermore, users were asked
to comment on the use of predecessors over successors and whether one method was more intuitive than
the other.

7.4.1 Nielsen Usability Heuristics

This section aims to summarise the Nielsen usability feedback (noted in figures F.1 to F.3, Appendix
E) obtained from user testing. Largely users were satisfied with general usability of the application as
indicated in the results. The majority of test users believed the systems dialogue was understandable
and was conveyed in a manner that users could understand. However, there were exceptions to this,
characterised by users 2, 3, 7 and 10 who believed that the error messages were not explained well
enough or were too technical, possibly due to the fact that the system is specific to a particular domain.

A further Nielsen heuristic is that of reducing memory load required of the user, which was agreed
upon by the majority. However this was offset by users 3, 7 and 10 who commented on the fact that
when creating a survey, lots of information such as pre-conditions and options had to be remembered in
order to define conditions. This is an important issue which could provide a basis for further research
by developing a user interface facilitating the creation of a survey but at the same time minimising the
memory load required of a user. All users agreed that the interface was consistent, characterised by
the use of colour, layout and style. The use of error messages were also commented upon with many
users agreeing that messages were appropriate and used in the correct places. Finally, positive feedback
was obtained for the help functionality, with most users being able to locate the help link correctly,
during a Q&A Session. However users 6 and 10 believed the application would benefit from help being
available during the creation of a survey via the Add a Question page. In doing so, this may indeed
counter the issues associated with memory load. In conclusion, the Nielsen Usability Heuristics were
largely satisfied. Although some feedback was negative, it provides a good starting point for further
development and research in producing an efficient yet user friendly environment for creating surveys.

7.4.2 Navigation

In order to evaluate the navigational menu found on the web site, figure F.4 in Appendix F shows a table
indicating the number of mouse clicks required of the user to navigate from one page to another. As can
be seen, the minimum number of clicks required is 1 due to the links found within the static menu. The maximum number of clicks is 3 but such cases only arise when the user has logged out. The number of clicks required in order to navigate to the MyReports.aspx page is dependent upon the number of questions asked to the user. Furthermore, the number of clicks required suggests that the overall navigational time is low, a point raised in section 4.4.3 and noted by the developer during user testing. Lastly, the menu takes up little space on the screen but is immediately identifiable as a navigational instrument, justified from the results obtained through testing, where no users had navigational issues.

### 7.5 Client Evaluation

The client evaluated the system by performing a set of tasks found in Figures E.12 to E.16. The outcome of the tasks were noted and feedback was obtained. On a whole the client was “very happy” [13] with what was achieved. From the tests carried out, few difficulties were encountered with usability issues noted. The ’User Administration’ tasks were completed successfully as were the ’Project Administration’ tasks. However during the creation of a survey, the client and other users suggested the use of Answer Option templates for commonly required options to reduce the amount of user input, which as a result may counter some of the usability points raised in section 7.4.1. Whilst adding questions into the system, the client forgot to specify a Question Type on several occasions which suggests that this functionality be repositioned on the form. Again this may be a consequence of the amount a user has to remember when creating a survey. Finally, when asked the question regarding the intuitiveness of successors over predecessors, the client stated that “the general maintainability of the application is more important than intuitiveness and the most technically efficient should not be compromised” [13]. This is indeed an interesting response, suggesting that the use of predecessors over successors makes little difference.

### 7.6 Comparison with the Clients Existing Solution

In evaluating the solution, it seems appropriate to compare it to the clients existing solution, noted in section 3.2.1. Currently, requirements elicitation is inflexible, taking place via a set of hard-coded questions and precursors and thus making configuration (such as the addition of questions) difficult. Furthermore, in comparison to the Bristol Online Survey [2] and SurveyMonkey [5] systems discussed in section 3.3.1, which are although not used by the clients organisation, do not allow for the use of
pre-conditions when adding questions and creating surveys. In comparison with the existing system, the new application has far more configurable properties, such as being able to add questions, predecessors, answer options and projects, with the ability to store users answers and generate reports. The application itself is web based thereby allowing authorised users to perform requirements elicitation with no geographical boundaries and as a consequence, security measures have been implemented. Furthermore no controls required for the Q&A Session are created manually as they are all generated on the form dynamically, thus cutting down the time required to build a survey. All these capabilities contrast with the current standalone system whereby questions, predecessors, answer options and form controls are added programmatically with no configuration ability and regards to security. Users answers are also not maintained in the clients current system.

In evaluating the solution it was suggested by Shakhlevich & Bennett [25] that a question be added to the new and existing solutions to ascertain the time taken to perform such an action in both systems. As such a single question (with a pre-condition and the answer options, ‘Yes’ and ‘No’) was added into the two applications to observe the required effort and time to perform such a task. The results for adding a question in the developed solution were positive, taking just over 2 minutes from accessing the relevant web page to submitting the new question. Adding the same question in the existing solution took substantially longer as the form controls (DropDownList, Labels and Buttons) had to be created manually and then the program methods had to be written to handle events on the controls, taking approximately 14 minutes. Further time was taken to test that the methods worked correctly. The developed solution is therefore not only more configurable but also far more scalable than the existing solution as the number of questions increase.

### 7.7 Limitations

The initial design of the elicitation algorithm involved the use of disjunctive and conjunctive pre-conditions, in which a question could have the predecessor:

\[
(Q1 = l \lor Q2 = \text{‘Yes’}) \land (Q3 > 0)
\]

Although such a condition did not form part of the initial requirements it was considered more flexible and beneficial if implemented, since it allowed conjunctive and disjunctive predecessors to be assessed. The difficulty encountered when implementing such functionality occurred when the algorithm attempted to find the questions with predecessors (similar to above) that had been satisfied, which
the developer believes is a consequence of the database schema and would require a minor revision to allow for disjunctive pre-conditions. Furthermore in conjunction with time constraints on the project, only disjunctive predecessors were therefore implemented.

A further limitation of the solution is that RadioButton controls were not implemented for boolean answers and as such DropDownList controls were used instead. Whilst it is acceptable for DropDownLists to be used for boolean answers, it seems more appropriate for RadioButtons to be utilised given their common relation to binary selections. RadioButtons were not implemented due to the nature in which event handlers were applied to dynamically created RadioButton controls and the difficulties encountered. Whilst the issue may be possible to overcome, the time spent trying to resolve it prevented further development and as such was abandoned.

7.8 Possible Enhancements

In order to enhance the systems current functionality the points raised in section 7.7 could be considered. As mentioned, difficulties were encountered when implementing the algorithm to consider conjunctive predecessors. Such functionality could be achieved with the addition of a table in the current database schema to store conjunctive and disjunctive predecessors for a question as a single record by relating predecessors within the ELICITATION_PREDECESSOR table. However the main challenge is writing the algorithm (in SQL) to find and update pre-conditions containing disjunctive and conjunctive relations. After this point the process of displaying the correct question is trivial.

The purpose of the QuestionOrder field found within the ELICITATION_QUESTION table (discussed in section 4.3.2) was to facilitate the re-ordering of questions independently of unique identifiers. Yet due to time constraints such functionality was not implemented but could be achieved via a ListBox control containing an ordered list of questions, along with two Buttons to move the selected questions up and down in the list. Whilst moving the question down the list would be unrestricted, moving a question up the list (resulting in a lower question number) would require checks to ensure that the questions precursor is not compromised (i.e. the pre-condition for question a does not refer to question b which comes after a).

In terms of the applications usability a Button to allow users to step back through the Q&A Session could be added and therefore allow users to change a previous answer. Caution however must be taken at this stage as stepping back through a Q&A Session has the potential for changing satisfied predecessors
and hence which questions are successively displayed. Although there will be little impact on the presentation layer since the controls are created dynamically, appropriate steps would be required to ensure correct pre-conditions are marked and unmarked as satisfied.

As a final enhancement, functionality could be placed within the system to allow questions and pre-conditions to be displayed as a tree structure, mapping all possible routes. In doing so it would allow the application administrators to find the most common routes taken by users during a Q&A Session. It may also act as a key maintenance tool and provide evidence for revising the entire question set, if the tree indicates that the current questions are not effective in achieving requirements information. Furthermore the tree may also act as an integrity checker, by highlighting any loops that may occur when users are answering questions.

### 7.9 Conclusion

The premise for the project was to produce a requirements elicitation tool to aid the Child Health data migration process within the Computer Sciences Corporation (CSC). However, at the same it was important to build a generic application by using the CSC problem domain as a platform for developing such a tool.

The main achievement of the project is that a configurable web application has been designed and implemented to allow the creation of dynamic web surveys through the use of pre-conditions. Furthermore, users are able to provide answers to questions which may then go on to determine which further questions are displayed, thus eradicating the risk of being asked irrelevant questions. Such functionality is achieved through the use of dynamic controls, which are created on the web page in run time and allow the users to input an answer. In addition to this, the web application can be applied to almost any field of endeavour, characterised by the generic design of the elicitation algorithm, which itself saw several iterations. Feedback obtained from the client and other users was positive with all minimum requirements and objectives fulfilled (and exceeded in some cases), evident from the testing and evaluation results. The implementation of a widely used development process and the utilisation of various technologies are also achievements in themselves, since the learning of new methods and concepts resulted in a fully functional and successfully deployed web application that was delivered on time.
Bibliography


Appendix A

Personal Reflection

The opportunity for the development of this project came about whilst working for the Computer Sciences Corporation (CSC) during my work placement year. Towards the start of the third academic year, I was keen to work on a project that had potential to make a difference and I therefore feel honoured to have been able to work with an old colleague and my client Dr. William Thirsk–Gaskill of CSC in producing a requirements elicitation tool.

Initially, I had different ideas about how I was going to attempt to solve the problem and when a new idea for the algorithm was proposed, I had difficulties in understanding how it would actually work. However, as I began to develop it further and with the help provided by Brandon it became easier. Upon reflecting on this stage in the project, I believe that I became so engaged with my initial ideas that I forgot about the bigger picture and trying to understand someone else’s approach became much harder, until I sought advice enabling me to understand the problem. My advice to other students in similar predicaments is to take time out and return to the problem later, I did this during the project and it certainly helped. Furthermore, don’t be afraid to ask for advice from supervisors or accessors, I found that discussing issues with them allowed me to focus and concentrate on the issues which mattered.

When selecting a project at the start of the year, my advice to other students is to select one that interests you and that perhaps you might enjoy. Don’t be afraid of a challenge, certainly pick a project that you are able to attain but at the same time don’t be put off by any pre-conceptions you may have. If you have been given the opportunity to develop a solution for a third party, think about how you could
evaluate it as this forms an important of the project.

During the initial stages of the project, ensure that you define realistic minimum requirements as over complicating the project can lead to a limited solution or late delivery.

With regards to the project schedule I found myself working more on the project in the last semester than in the first, therefore my advice is to do as much work as possible before the Christmas break as by the time January comes revision takes over and project work does not resume until late January, thus demonstrating the importance of time management. In scheduling the work make sure you plan it out early, and base your project plan around the chosen methodology as it is then easier to plan the workload and structure the final report. Furthermore do not underestimate the workload especially the time taken for implementing the solution (it really does take time and careful thought), start coding as soon as possible. If you are using a new technology/language ensure that enough time is spent getting to grips with it, avoid learning the language and implementing a solution at the same time as the final product may be limited. When writing up the project, ensure that you have at least a month to do so as planning and writing takes longer than you think. Most importantly, backup your work throughout and keep old copies, then if work gets lost or corrupted it can be restored (remember if lost, 8 months worth of work is gone!).

When dealing with third parties ensure that a professional conduct is maintained throughout (remember you are representing the University and all your peers). Ensure that when meetings are arranged, you are aware of the meeting place and arrive on time. Remember the client is taking time out of their schedule to meet you and if you are late, the meeting will overrun. Before the meeting, make sure you are prepared. My advice is to draft out a set of questions/points you would like to address during the meeting and then cross them off as they are discussed. Also during the process, don’t be afraid to cast your own points or concerns. Furthermore, make notes on what the client says in the meeting, as these will form the requirements of the project. Don’t however pre-empt the client by telling them what they want, instead use your knowledge to convey how such requirements could be achieved.

Finally, the project should be about learning and producing a quality piece of work, so have fun at the same time.
Appendix B

Requirements Capture and Analysis

Figure B.1: UML Implementation Diagram
Figure B.4: UML User Case Diagram: Application Administrator
Figure B.5: UML User Case Diagram: Limited User
Figure B.6: UML Activity Diagram: Add a Question
Figure B.7: UML Activity Diagram: Add a User
<table>
<thead>
<tr>
<th>Priority</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Must Haves</td>
<td>Add User</td>
</tr>
<tr>
<td></td>
<td>Delete User</td>
</tr>
<tr>
<td></td>
<td>Change a Password</td>
</tr>
<tr>
<td></td>
<td>Add Question</td>
</tr>
<tr>
<td></td>
<td>Submit Answer</td>
</tr>
<tr>
<td></td>
<td>Generate Report</td>
</tr>
<tr>
<td></td>
<td>Login</td>
</tr>
<tr>
<td></td>
<td>Logout</td>
</tr>
<tr>
<td></td>
<td>Create Q&amp;A Session</td>
</tr>
<tr>
<td>Should Haves</td>
<td>Export Report</td>
</tr>
<tr>
<td></td>
<td>Add Project</td>
</tr>
<tr>
<td></td>
<td>Get Help</td>
</tr>
<tr>
<td>Could Haves</td>
<td>Modify Question</td>
</tr>
<tr>
<td></td>
<td>Delete Question</td>
</tr>
<tr>
<td>Won’t Haves</td>
<td>Delete Q&amp;A Session</td>
</tr>
</tbody>
</table>
1. How many sets of schedules does your Child Health system currently run?

   CH User Type: Operational
   Exp Data Type: Integer
   Predecessor: N/A

2. Are all these run from the same site?

   CH User Type: Operational
   Exp Data Type: Boolean
   Predecessor: Q1 > 1

3. Does each office or site see exactly the same view of the Child Health system (same software, same data)?

   CH User Type: Technical
   Exp Data Type: Boolean
   Predecessor: Q2 = ‘No’

4. Does your current Child Health system contain ANY patient records from any other care settings (e.g. Community, Mental Health, acute care)?

   CH User Type: Operational
   Exp Data Type: Boolean
   Predecessor: (Q1 = 0) ∨ (Q2 = ‘Yes’) ∨ (Q3 = ‘No’) ∨ (Q3 = ‘Yes’)

5. Please list all the other care settings which have records in the same system as Child Health.

   CH User Type: Operational
   Exp Data Type: String
   Predecessor: Q4 = ‘Yes’
6.) Please describe in as much detail as possible how you tell a Child Health record from those belonging to each of the other care settings.

CH User Type: Operational
Exp Data Type: String
Predecessor: (Q4 = ‘No’) ∨ (Q5 = ‘’) ∨ (Q5 != ‘’)

7.) Please enter the age at which a living patient ceases to be under the care of the Child Health unit.

CH User Type: Operational
Exp Data Type: Integer
Predecessor: (Q6 = ‘’) ∨ (Q6 != ‘’)

8.) Does this age also apply to patients with special needs?

CH User Type: Operational
Exp Data Type: Boolean
Predecessor: Q7 != ‘’

9.) What is the age for patients with special needs?

CH User Type: Operational
Exp Data Type: Integer
Predecessor: Q8 = ‘No’

10.) Please enter the registration status which indicates a birth which is transferred out.

CH User Type: Operational
Exp Data Type: String
Predecessor: (Q9 <= 0) ∨ (Q8 = ‘Yes’)

11.) Please enter the registration status which indicates a patient who has moved out.

CH User Type: Operational
Exp Data Type: String
Predecessor: (Q10 = ‘’ ∨ (Q10 != ‘’)}
12.) *Of all the deceased patients in the current system, do you require any to be EXcluded from SystmOne?*

*CH User Type:* Operational  
*Exp Data Type:* Boolean  
*Predecessor:* \((Q11 = \text{``''}) \lor (Q11 ! = \text{``''})\)

13.) *Is your current system provided by a vendor or in-house?*

*CH User Type:* Technical  
*Exp Data Type:* Boolean  
*Predecessor:* N/A

14.) *What is the name of the vendor?*

*CH User Type:* Technical  
*Exp Data Type:* String  
*Predecessor:* Q13 = ‘Vendor’
Figure B.11: Project Terminology Definitions used in Chapter 3

**Application** – Refers to the computer program under development (i.e. the project deliverable)

**Limited Application User** – Refers to an individual who uses the application being developed for this project.

**Application Administrator** – Refers to an individual who may or may not use the application but has overall control of the system and acts as the governing individual.

**NPfIT** – National Programme for IT (when read aloud it sounds like “enn–pee–fit”. Current project undertaken where by the UK government contract various private-sector companies, to work together along with the NHS to upgrade the current NHS IT infrastructure.

**NHS Trust** – An NHS Trust is an organization which provides healthcare services to a designated area, e.g. North Lincolnshire PCT.

**CH Project** – Child Health Project. This is a term used by CSC as a unique reference to the deployment of the Child Health system to an NHS Trust. E.g. Northeast Lincolnshire.

**NHS CH Users** – Users who work for the National Health Service and use the Child Health System. There are two types of users who use the Child Health system

1. NHS CH Technical User – deal with the technical/DBA side of Child Health.

2. NHS CH Operational User – Use the Child Health system as part of their job.

**CH system** – Child Health System.

**Q&A Session** – Is a time over which an application user uses the system and provides answers to questions. The end of the session is signalled when the user answers the very last question and a report is produced, detailing which questions were asked along with the answers. A Q&A session may occur more than once for a CH Project (described above).

**CH** – Stands for Child Health. The module is part of a software suite called “SystmOne” which serves a number of care settings including GP, Community and Hospice, as well as Child Health.

**Legacy CH System** – An existing Child Health system currently being used by an NHS Trust.
Notes from meeting

The purpose of this document is to provide the reader with an overview of the discussions during the first meeting between myself and the client. Throughout the meeting my client and I discussed what would be required of the proposed system. Due to working with the client during my work placement year, I am currently aware of the fundamentals of what is required of the proposed system and thus during the meeting we were able to concentrate in more specific areas. In the meeting I introduced my client to the MuSCoW system of modelling requirements. This enabled us to break down the project and prioritise certain tasks. My made clear the following functionality which should be apparent:

Must Haves... GENERAL: Database of questions/answers, user roles, user interface, output (in the form of a report) and a list of un-answered questions. SPECIFIC: Ability to add a user, delete and change passwords – use of Login credentials. Create Q&A Sessions and answer questions. Ability for administrators to add questions. Users should be able to generate reports for a project.

Should Haves... GENERAL: None SPECIFIC: Adding projects into the database, for surveys to be carried out against. Help functionality. Export

Could Haves... GENERAL: None SPECIFIC: Ability to change a question, though isnt of utmost importance in this iteration. Would be nice though... Should be able to delete projects.

Won’t Haves... GENERAL: None SPECIFIC: None

Discussion surrounding the users... In terms of the users who will be using the system, the client made it clear that there will be two users of the application. A user and administrator. Each of which will have different rights with regards to the system
User:

- Will have the ability to log in to the system and start the question answering process.
- May view and print out the final report.

Administrator:

- Will be able to add and remove users from the application.
- Will be able to define a user role (see the next page), which will ultimately determine which questions they answer.
- Should have the ability to add or remove questions.
- Change the question ordering... not crucial at this stage.

With respect to the system, client emphasised importance of drawing a distinction between the types of users referred to throughout the duration of the project and in order to make this clear I will now provide descriptions for the types of users. Essentially, there will be users of the application (the proposed system) and users whose field of endeavour we are attempting to get information about. The user of the application can be defined as a normal user and an administrator. This user will then go on to answer questions for their field of endeavour. Clarified below.

Users of the application being developed (the Project)

- Users
- Administrators

Users whose field of endeavour we are attempting to gather information about

- Database Administrators
- Operational Managers
The points above therefore provides us with the types of users who will be using the application (top) and the users professions whom we are gathering information regarding their field of endeavour (bottom). This will then go on to determine which questions the user answers. If they are database administrators, they will answer technical issues. If they are operation managers, they will answer questions relating to the general day-to-day running of the system. However, if the user is a database administrator and operational manager then they will go onto answer questions regarding both topics. These types of user will be added into the database and will also be recorded against the questions, to identify the type of question (technical or non-technical). Security measures were discussed and it was agreed that usernames and passwords will be implemented to ensure that no unauthorised users can access the system.
Appendix C

Design of Proposed Solution

Figure C.1: Mock Up: Interface Design
Figure C.2: Entity Relationship Diagram: Administrative User
Figure C.3: Entity Relationship Diagram: Limited User
Appendix D

Implementation

Figure D.1: Screenshot: Login (Default.aspx)
Welcome to Inquisitor

You are about to be asked a series of questions about your current Child Health system and the requirements for the migration of data into SystemOne. The result of this will be a submitted report which may be read and amended as required. The purpose of the questions is to get specific information about the most important aspects of the migration and to create headings which are relevant to your particular Child Health unit.

To begin a question and answer session please fill the project field and your user type with regards to the Child Health system.

Any additional information which you feel is relevant but which is not requested should be included at the end of the session. It is much better at this stage to provide too much information rather than too little.

Firstly, please select the your role with regards to the Child System and then select the project in which you are affiliated with. Then press 'Continue' to proceed to the questions.

---

User Administration

Here the administrator can make add and delete users of the web site.

Sign Up for Your New Account

Username: [Enter username]
User Role: [Please select]
Password: [Enter password]
Confirm Password: [Confirm password]
E-mail: [Enter email]

Create User

Change User’s Password

Username: [Please select]
New Password: [Enter new password]
Confirm Password: [Confirm new password]

Change

Delete User Account

Username: [Please select]

---
Figure D.4: Screenshot: Add a Project (AddProject.aspx)

Figure D.5: Screenshot: Delete a Project (DeleteProject.aspx)
Add a Question

Here you can add new questions into the system. When all the necessary details for the question have been entered, click on the 'Add Question' button.

Please be aware that there is only 1 user(s) currently online (including yourself) and it is recommended that questions are added when there are no other active users.

Current Questions held in Database

Displayed below is the list of questions (along with their positions) currently held within the database.

1. How many sets of schedules does your current Child Health system run?
2. Are all these run from the same site?
3. Does each office or site have exactly the same view of the Child Health system (same software, same data)?
4. Does your current Child Health system contain any patient records from other care settings (e.g. Community, ...
5. Please list all the other care settings which have records in the same system as Child Health?

Question Details

Here input the details for the question. The 'Control field' refers to the type of form control (e.g. Textbox) that the answer will be input in.

Question Position: ?
Question Type: - Please select -

Answer Options

Here input the selectable answers that will be required for the question.

Option:
Clear  Add >

Pre-Conditions

Tick, if this question requires 'Predecessors' to be set.

The predecessors refer to the questions that must be answered before the question being input is selected. You can add many predecessors.

Predecessor: - Please select -
Operand: - Please select -
Value:
Clear  Add >
Figure D.8: Screenshot: Question and Answer Session in Progress (QuestionAnswerSession.aspx)

Figure D.9: Screenshot: Question and Answer Session completed (QuestionAnswerSession.aspx)
Figure D.10: Screenshot: My Report (MyReport.aspx)

![Screenshot: My Report (MyReport.aspx)](image1)

The table below shows a report containing your answers. To export this to a spreadsheet, click on Export at the bottom.

<table>
<thead>
<tr>
<th>No.</th>
<th>Question</th>
<th>Answer</th>
<th>User</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How many sets of schedules does your current Child Health system run?</td>
<td>2</td>
<td>matthew</td>
<td>04/04/2008</td>
</tr>
<tr>
<td>2</td>
<td>Are all these run from the same site?</td>
<td>Yes</td>
<td>matthew</td>
<td>04/04/2008</td>
</tr>
<tr>
<td>4</td>
<td>Does your current Child Health system contain any patient records from other care settings (e.g. Community, Mental Health, Acute Care)?</td>
<td>No</td>
<td>matthew</td>
<td>04/04/2008</td>
</tr>
<tr>
<td>6</td>
<td>Please describe in as much detail as possible how you tell a Child Health record from those belonging to each of the other care settings?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Export

Copyright © 2008, University of Leeds

---

Figure D.11: Screenshot: Reports (before being run) (Reports.aspx)

![Screenshot: Reports (before being run) (Reports.aspx)](image2)

Use this page to generate a report containing the answers a certain user entered for a particular project. The drop down list below contains all Projects for which 'Question and Answer Sessions' have taken place.

- Please select -

Generate

Copyright © 2008, University of Leeds

78
**Figure D.12: Screenshot: Reports (after being run) (Reports.aspx)**

![Screenshot: Reports (after being run) (Reports.aspx)](image1)

**Inquisitor**
A requirements elicitation tool

Reports

Use this page to generate a report containing the answers a certain user entered for a particular project. The drop down list below contains all Projects for which ‘Question and Answer Sessions’ have taken place.

![Project: Durham](image2)

**Durham Project**

<table>
<thead>
<tr>
<th>No.</th>
<th>Question</th>
<th>Answer</th>
<th>User</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How many sets of schedules does your current Child Health system run?</td>
<td>2</td>
<td>matthew</td>
<td>06/04/2009</td>
</tr>
<tr>
<td>2</td>
<td>Are all these run from the same site?</td>
<td>Yes</td>
<td>matthew</td>
<td>06/04/2009</td>
</tr>
<tr>
<td>4</td>
<td>Does your current Child Health system contain any patient records from other care settings (e.g. Community, Mental Health, Acute Care)?</td>
<td>No</td>
<td>matthew</td>
<td>06/04/2009</td>
</tr>
<tr>
<td>6</td>
<td>Please describe in as much detail as possible how you tell a Child Health record from those belonging to each of the other care settings?</td>
<td>Answer is provided here</td>
<td>matthew</td>
<td>06/04/2009</td>
</tr>
</tbody>
</table>

![Export](image3)

**Figure D.13: Screenshot: Export Report (Reports.aspx)**

![Screenshot: Export Report (Reports.aspx)](image4)

![File Download](image5)

Do you want to open or save this file?

- **Name:** Durham.xlsx
- **Type:** Microsoft Excel Workbook, 1.33 KB
- **From:** localhost

- Open
- Save
- Cancel

While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save the file. What’s next?
Figure D.14: Screenshot: Log Out (with Remember Me (LoggedOut.aspx))

Figure D.15: Screenshot: Log Out (without Remember Me (LoggedOut.aspx))
The table below shows the list of questions along with their pre-conditions that are currently held within the system's database:

<table>
<thead>
<tr>
<th>No.</th>
<th>Question</th>
<th>Pre-Condition</th>
<th>Constraint Value</th>
<th>Question Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How many sets of schedules does your current Child Health system run?</td>
<td>1</td>
<td>Greater than</td>
<td>Operational</td>
</tr>
<tr>
<td>2</td>
<td>Are all these runs from the same site?</td>
<td>1</td>
<td>Greater than</td>
<td>Operational</td>
</tr>
<tr>
<td>3</td>
<td>Does each office or site see exactly the same view of the Child Health system (same software, same data)?</td>
<td>2</td>
<td>Equal to    No</td>
<td>Technical</td>
</tr>
<tr>
<td>4</td>
<td>Does your current Child Health system contain any patient records from other care settings (e.g. Community, Mental Health, Acute Care)?</td>
<td>1</td>
<td>Equal to    0</td>
<td>Operational</td>
</tr>
<tr>
<td>5</td>
<td>Does your current Child Health system contain any patient records from other care settings (e.g. Community, Mental Health, Acute Care)?</td>
<td>1</td>
<td>Equal to    1</td>
<td>Operational</td>
</tr>
<tr>
<td>6</td>
<td>Does your current Child Health system contain any patient records from other care settings (e.g. Community, Mental Health, Acute Care)?</td>
<td>2</td>
<td>Equal to    Yes</td>
<td>Operational</td>
</tr>
<tr>
<td>7</td>
<td>Please list all the other care settings which have records in the same system as Child Health.</td>
<td>4</td>
<td>Equal to    Yes</td>
<td>Operational</td>
</tr>
<tr>
<td>8</td>
<td>Please describe in as much detail as possible how you tell a Child Health record from those belonging to each of the other care settings?</td>
<td>4</td>
<td>Equal to    No</td>
<td>Operational</td>
</tr>
<tr>
<td>9</td>
<td>Please describe in as much detail as possible how you tell a Child Health record from those belonging to each of the other care settings?</td>
<td>5</td>
<td>Equal to    Yes</td>
<td>Operational</td>
</tr>
<tr>
<td>10</td>
<td>Please describe in as much detail as possible how you tell a Child Health record from those belonging to each of the other care settings?</td>
<td>5</td>
<td>Not equal to</td>
<td>Operational</td>
</tr>
</tbody>
</table>
Figure D.17: Stored Procedure: CreateSession

```
ALTER PROCEDURE [dbo].[createSession]
    @ProjectID int,
    @QuestionTypeID int
AS
/* PROCEDURE TO CREATE A NEW Q&A SESSION FOR A PROJECT */
BEGIN TRANSACTION

    /* Insert predecessors into table */
    INSERT INTO elicitation_PredecessorsSatisfied (PredecessorID)
    SELECT PredecessorID FROM elicitation_Predecessor
    WHERE QuestionTypeID = @QuestionTypeID

    /* Now update the table with the ProjectID */
    UPDATE elicitation_PredecessorsSatisfied SET ProjectID = @ProjectID
    WHERE ProjectID IS NULL

    /* Now set the satisfied row to 1 for the opening question */
    UPDATE elicitation_PredecessorsSatisfied SET Satisfied = 1
    WHERE PredecessorID =
        SELECT TOP 1 elicitation_Predecessor.PredecessorID
        FROM elicitation_Question
        INNER JOIN elicitation_Predecessor
            ON elicitation_Question.QuestionID = elicitation_Predecessor.QuestionID
        WHERE elicitation_Predecessor.QuestionTypeID = @QuestionTypeID
    ORDER BY elicitation_Question.QuestionOrder) AND
    ProjectID = @ProjectID

    /* Catch error if raised */
    IF @ERROR <> 0
    BEGIN
        ROLLBACK TRANSACTION
        RETURN
    END

    COMMIT TRANSACTION
```
Figure D.18: Stored Procedure: GetFirstQuestion

```sql
ALTER PROCEDURE [dbo].[getFirstQuestion]
    @ProjectID int,
    @QuestionTypeID int
AS

SELECT elicitation_question.QuestionID,
       elicitation_question.QuestionOrder,
       elicitation_Control.Control,
       elicitation_question.Question,
       elicitation_dataType.Syntax
FROM elicitation_question
INNER JOIN elicitation_Control
    ON elicitation_question.ControlID = elicitation_Control.ControlID
INNER JOIN elicitation_dataType
    ON elicitation_question.DataTypeID = elicitation_dataType.DataTypeID
WHERE QuestionID =
    (SELECT TOP 1 elicitation_Predecessor.QuestionID
     FROM elicitation_Predecessor
     INNER JOIN elicitation_question
         ON elicitation_Predecessor.QuestionID = elicitation_question.QuestionID
     INNER JOIN elicitation_PredecessorsSatisfied
         ON elicitation_Predecessor.PredecessorID = elicitation_PredecessorsSatisfied.PredecessorID
     WHERE elicitation_PredecessorsSatisfied.ProjectID = @ProjectID AND
           elicitation_Predecessor.QuestionTypeID = @QuestionTypeID AND
           elicitation_PredecessorsSatisfied.Satisfied IS NOT NULL AND
           elicitation_Predecessor.QuestionID
     ORDER BY elicitation_question.QuestionOrder)
```
Figure D.19: Stored Procedure: GetNextQuestion

```
ALTER PROCEDURE [dbo].[getNextQuestion]
    @ProjectID int,
    @QuestionTypeID int,
    @CurrentQuestionID int
AS

/* Get the next question to ask */
DECLARE @QuestionToBeAsked int

SET @QuestionToBeAsked =
    (SELECT TOP 1 elicitation_Predecessor.QuestionID
    FROM elicitation_Predecessor
    INNER JOIN elicitation_Question
        ON elicitation_Predecessor.QuestionID = elicitation_Question.QuestionID
    INNER JOIN elicitation_PredecessorsSatisfied
        ON elicitation_Predecessor.PredecessorID = elicitation_PredecessorsSatisfied.PredecessorID
    WHERE elicitation_Predecessor.QuestionID <> @CurrentQuestionID AND
        elicitation_PredecessorsSatisfied.ProjectID = @ProjectID AND
        elicitation_Predecessor.QuestionTypeID = @QuestionTypeID AND
        elicitation_PredecessorsSatisfied.Satisfied IS NOT NULL AND
        elicitation_Predecessor.QuestionID NOT IN
            (SELECT QuestionID FROM elicitation_Answer
            WHERE ProjectID = @ProjectID)
    ORDER BY elicitation_Question.QuestionOrder)

/* How do the necessary joins to get the information for question generation */
SELECT elicitation_Question.QuestionID,
    elicitation_Question.QuestionOrder,
    elicitation_Control.ControlID,
    elicitation_Question.Syntax
FROM elicitation_Question
INNER JOIN elicitation_Control
    ON elicitation_Question.ControlID = elicitation_Control.ControlID
INNER JOIN elicitation_DataType
    ON elicitation_Question.DataTypeID = elicitation_DataType.DataTypeID
WHERE QuestionID = @QuestionToBeAsked
```
Figure D.20: Stored Procedure: InsertAnswer (Page 1)

```sql
ALTER PROCEDURE [dbo].[InsertAnswer]
    @QuestionID int,
    @ProjectID int,
    @Username varchar(200),
    @Answer varchar(250),
    @QuestionTypeID int
AS

    /*--------------------------- SET THE USER ID ------------------------------- */
    DECLARE @DateAdded datetime
    DECLARE @UserID varchar(200)

    SET @UserID = (SELECT UserID FROM aspnet_users
                   WHERE Username = @Username)

    SET @DateAdded = GETDATE()

    /*--------------------------- PHASE 1 ------------------------------- */
    /*1. INSERT THE ANSWER*/
    BEGIN TRANSACTION

    INSERT INTO elicitation_answer VALUES (@QuestionID, @ProjectID, @UserID, @Answer, @DateAdded)

    IF @@error <> 0
        BEGIN
            RAISERROR('Failed to add Answer', 16, 1)
            ROLLBACK TRANSACTION
            RETURN
        END

        COMMIT TRANSACTION

    /*--------------------------- PHASE 2 ------------------------------- */
    /*2. FIND OUT WHICH PREDECESSOR ID(S) HAVE BEEN SATISFIED WITH THIS ANSWER, UPDATE*/
    IF (SELECT TOP 1 PredecessorID FROM elicitation_Predecessor
    ```
Figure D.21: Stored Procedure: InsertAnswer (Page 2)

```sql
WHERE Predecessor = @QuestionID AND
    @Answer < [Value] AND
    OperandID = 2 AND
    QuestionTypeID = @QuestionTypeID) IS NOT NULL

UPDATE elicitation_PredecessorsSatisfied SET Satisfied = 1
WHERE ProjectID = @ProjectID AND PredecessorID IN
(SELECT PredecessorID FROM elicitation_Predecessor
WHERE Predecessor = @QuestionID AND @Answer = [Value]
    AND OperandID = 2 AND QuestionTypeID = @QuestionTypeID)

ELSE IF (SELECT TOP 1 PredecessorID FROM elicitation_Predecessor
WHERE Predecessor = @QuestionID AND @Answer = [Value]
    AND OperandID = 1 AND QuestionTypeID = @QuestionTypeID) IS NOT NULL

UPDATE elicitation_PredecessorsSatisfied SET Satisfied = 1
WHERE ProjectID = @ProjectID AND PredecessorID IN
(SELECT PredecessorID FROM elicitation_Predecessor
WHERE Predecessor = @QuestionID AND @Answer = [Value]
    AND OperandID = 1 AND QuestionTypeID = @QuestionTypeID)

ELSE IF (SELECT TOP 1 PredecessorID FROM elicitation_Predecessor
WHERE Predecessor = @QuestionID AND @Answer > [Value]
    AND OperandID = 3 AND QuestionTypeID = @QuestionTypeID) IS NOT NULL

UPDATE elicitation_PredecessorsSatisfied SET Satisfied = 1
WHERE ProjectID = @ProjectID AND PredecessorID IN
(SELECT PredecessorID FROM elicitation_Predecessor
WHERE Predecessor = @QuestionID AND @Answer > [Value]
    AND OperandID = 3 AND QuestionTypeID = @QuestionTypeID)

/* If the answer is NULL */
ELSE IF (SELECT TOP 1 PredecessorID FROM elicitation_Predecessor
WHERE Predecessor = @QuestionID AND (@Answer IS NULL) AND ([Value] IS NULL)
    AND OperandID = 1 AND QuestionTypeID = @QuestionTypeID) IS NOT NULL
```
Figure D.22: Stored Procedure: InsertAnswer (Page 3)

```sql
UPDATE elicitation_PredecessorsSatisfied SET Satisfied = 1
WHERE ProjectID = @ProjectID AND PredecessorID IN
(SELECT PredecessorID FROM elicitation_Predecessor
WHERE Predecessor = @QuestionID AND (@Answer IS NULL AND [(Value) IS NULL]
AND OperandID = 1 AND QuestionTypeID = @QuestionTypeID)

/* IF the answer IS NOT NULL */
ELSE IF (SELECT TOP 1 PredecessorID FROM elicitation_Predecessor
WHERE Predecessor = @QuestionID AND (@Answer IS NOT NULL AND [(Value) IS NULL]
AND OperandID = 4 AND QuestionTypeID = @QuestionTypeID) IS NOT NULL

UPDATE elicitation_PredecessorsSatisfied SET Satisfied = 1
WHERE ProjectID = @ProjectID AND PredecessorID IN
(SELECT PredecessorID FROM elicitation_Predecessor
WHERE Predecessor = @QuestionID AND (@Answer IS NOT NULL AND [(Value) IS NULL]
AND OperandID = 4 AND QuestionTypeID = @QuestionTypeID)

/*----------------------------- PHASE 3 ----------------------------- */

/*4. Get the next question to ask by calling the 'getNextQuestion' stored procedure*/
EXECUTE getNextQuestion @ProjectID, @QuestionTypeID, @QuestionID
```

Figure D.23: Stored Procedure: GetAnswersForReportQuery

```sql
set ANSI NULLS ON
set QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[getAnswersForReport]
    @ProjectID int
AS

SELECT elicitation_Question.QuestionOrder AS [No.],
    elicitation_Question.Question AS [Question],
    elicitation_Answer.Answer AS [Answer],
    aspnet_Users.UserName AS [User],
    CONVERT (varchar(10), elicitation_Answer.DateAdded, 103) AS [Date]
FROM elicitation_Answer
INNER JOIN elicitation_Project ON elicitation_Answer.ProjectID = elicitation_Project.ProjectID
INNER JOIN elicitation_Question ON elicitation_Answer.QuestionID = elicitation_Question.QuestionID
INNER JOIN aspnet_Users ON elicitation_Answer.UserID = aspnet_Users.UserID
WHERE elicitation_Project.ProjectID = @ProjectID
ORDER BY elicitation_Question.QuestionOrder ASC
```
Figure D.24: Code: Insert New Question (ElicitationDAL.cs)

```csharp
/// <param name="questionOrder">writing</param>
/// <param name="question">int</param>
public static int InsertNewQuestion(int questionOrder, int controlID, string question, int dataTypeID)
{
    try
    {
        OpenConnection();

        SqlCommand cmdInsertNewQuestion = new SqlCommand();
        cmdInsertNewQuestion.Connection = _dbConnection;
        cmdInsertNewQuestion.CommandType = CommandType.StoredProcedure;
        cmdInsertNewQuestion.CommandText = "insertNewQuestion";

        cmdInsertNewQuestion.Parameters.Add(new SqlParameter("@QuestionOrder", questionOrder));
        cmdInsertNewQuestion.Parameters.Add(new SqlParameter("@ControlID", controlID));
        cmdInsertNewQuestion.Parameters.Add(new SqlParameter("@Question", question));

        if (dataTypeID == 0)
        {
            cmdInsertNewQuestion.Parameters.Add(new SqlParameter("@DataTypeID", 4));
        }
        else
        {
            cmdInsertNewQuestion.Parameters.Add(new SqlParameter("@DataTypeID", dataTypeID));
        }

        SqlParameter outputParameter = cmdInsertNewQuestion.Parameters.Add("@QuestionID", SqlDbType.Int);
        outputParameter.Direction = ParameterDirection.Output;

        cmdInsertNewQuestion.ExecuteNonQuery();
        int outputValue = (int)cmdInsertNewQuestion.Parameters["@QuestionID"].Value;

        return outputValue;
    }
    catch (SqlException e)
    {
        throw e;
    }
}
finally
{
    CloseConnection();
}
```
```csharp
public static void InsertNewPredecessor(int questionID, int predecessor, int operandID, string value, int questionTypeID)
{
    try
    {
        OpenConnection();

        SqlCommand cmdInsertNewPredecessor = new SqlCommand();
        cmdInsertNewPredecessor.Connection = _dbConnection;
        cmdInsertNewPredecessor.CommandType = CommandType.StoredProcedure;
        cmdInsertNewPredecessor.CommandText = "insertNewPredecessor";

        cmdInsertNewPredecessor.Parameters.Add(new SqlParameter("@questionID", questionID));
        if (predecessor == 0)
        {
            cmdInsertNewPredecessor.Parameters.Add(new SqlParameter("@predecessor", DBNull.Value));
        }
        else
        {
            cmdInsertNewPredecessor.Parameters.Add(new SqlParameter("@predecessor", predecessor));
        }
        if (operandID == 0)
        {
            cmdInsertNewPredecessor.Parameters.Add(new SqlParameter("@operandID", DBNull.Value));
        }
        else
        {
            cmdInsertNewPredecessor.Parameters.Add(new SqlParameter("@operandID", operandID));
        }
        if (value == "NULL")
        {
            cmdInsertNewPredecessor.Parameters.Add(new SqlParameter("@value", DBNull.Value));
        }
        else
        {
            cmdInsertNewPredecessor.Parameters.Add(new SqlParameter("@value", value));
        }
        cmdInsertNewPredecessor.Parameters.Add(new SqlParameter("@questionTypeID", questionTypeID));
        cmdInsertNewPredecessor.ExecuteNonQuery();
    }
    catch (SqlException e)
    {
    }
}
```
Figure D.26: Code: Insert New Answer Option (ElicitationDAL.cs)

```csharp
/// <summary>
/// Method to insert a new answer option, for a given question ID
/// </summary>
/// <param name="questionID"></param>
/// <param name="answerOption"></param>
public static void InsertNewAnswerOption(int questionID, string answerOption)
{
    try
    {
        OpenConnection();
        SqlCommand cmdInsertNewAnswerOption = new SqlCommand();
        cmdInsertNewAnswerOption.Connection = _dbConnection;
        cmdInsertNewAnswerOption.CommandType = CommandType.StoredProcedure;
        cmdInsertNewAnswerOption.CommandText = "InsertNewAnswerOption";
        cmdInsertNewAnswerOption.Parameters.AddWithValue("@QuestionID", questionID);
        cmdInsertNewAnswerOption.Parameters.AddWithValue("@AnswerOption", answerOption);
        cmdInsertNewAnswerOption.ExecuteNonQuery();
    }
    catch (SqlException e)
    {
        throw e;
    }
    finally
    {
        CloseConnection();
    }
```
    /// <param name="projectID">\</param>
    public static void CreateSession(int projectID, int questionTypeID)
    {
        try
        {
            OpenConnection();

            SqlCommand cmdCreateQASession = new SqlCommand();
            cmdCreateQASession.Connection = _dbConnection;
            cmdCreateQASession.CommandType = CommandType.StoredProcedure;
            cmdCreateQASession.CommandText = "createSession";

            cmdCreateQASession.Parameters.AddWithValue("@ProjectID", projectID);
            cmdCreateQASession.Parameters.AddWithValue("@QuestionTypeID", questionTypeID);

            cmdCreateQASession.ExecuteNonQuery();
        }
        catch (SqlException e)
        {
            throw e;
        }
        finally
        {
            CloseConnection();
        }
    }
```csharp
public static DataSet GetFirstQuestion(int projectId, int questionTypeId)
{
    try
    {
        OpenConnection();

        SqlCommand cmdGetFirstQuestion = new SqlCommand();
        cmdGetFirstQuestion.Connection = _dbConnection;
        cmdGetFirstQuestion.CommandType = CommandType.StoredProcedure;
        cmdGetFirstQuestion.CommandText = "GetFirstQuestion";

        cmdGetFirstQuestion.Parameters.Add(new SqlParameter("@projectId", projectId));
        cmdGetFirstQuestion.Parameters.Add(new SqlParameter("@questionTypeId", questionTypeId));

        SqlDataAdapter dataAdapter = new SqlDataAdapter();
        dataAdapter.SelectCommand = cmdGetFirstQuestion;

        DataSet dataSet = new DataSet();
        dataAdapter.Fill(dataSet);

        return dataSet;
    }
    catch (Exception ex)
    {
        throw ex;
    }
    finally
    {
        CloseConnection();
    }
}
```
```csharp
    public static DataSet InsertAnswer(int questionID, int projectId,
                                      string username, string answer, int questionTypeID)
    {
        try
        {
            OpenConnection();

            SqlCommand cmdInsertAnswer = new SqlCommand();
            cmdInsertAnswer.Connection = _dOConnection;
            cmdInsertAnswer.CommandType = CommandType.StoredProcedure;
            cmdInsertAnswer.CommandText = "insertAnswer";

            cmdInsertAnswer.Parameters.Add(new SqlParameter("@questionID", questionID));
            cmdInsertAnswer.Parameters.Add(new SqlParameter("@projectID", projectId));
            cmdInsertAnswer.Parameters.Add(new SqlParameter("@username", username));

            if (answer == "")
            {
                cmdInsertAnswer.Parameters.Add(new SqlParameter("@answer", DBNull.Value));
            }
            else
            {
                cmdInsertAnswer.Parameters.Add(new SqlParameter("@answer", answer));
            }

            cmdInsertAnswer.Parameters.Add(new SqlParameter("@questionTypeID", questionTypeID));

            SqlDataAdapter dataAdapter = new SqlDataAdapter();
            dataAdapter.SelectCommand = cmdInsertAnswer;

            DataSet dataSet = new DataSet();
            dataAdapter.Fill(dataSet);

            return dataSet;
        }
        catch (SqlException e)
```
Figure D.30: Code: Snippet from the ControlCreator.cs class, to return controls

```csharp
namespace ElicitationTool
{
    public class ControlCreator
    {
        /// <summary>
        /// Method to create a Label
        /// </summary>
        /// <param name="text">Label text</param>
        /// <returns>Label</returns>
        public Label NewLabel(string text)
        {
            Label lblLabel = new Label();
            lblLabel.Text = text;
            lblLabel.EnableViewState = true;
            lblLabel.Font.Name = "Arial";
            lblLabel.Font.Size = 10;
            return lblLabel;
        }

        /// <summary>
        /// Method to create a new textbox
        /// </summary>
        /// <returns>TextBox</returns>
        public TextBox NewTextBox()
        {
            TextBox txtTextBox = new TextBox();
            txtTextBox.EnableViewState = true;
            txtTextBox.ID = "txtTextBox";
            txtTextBox.MaxLength = 250;
            txtTextBox.Width = 332;
            txtTextBox.Height = 50;
            txtTextBox.Font.Name = "Arial";
            txtTextBox.TextMode = TextBoxMode.Multiline
            return txtTextBox;
        }
    }
}
```
Figure D.31: Code: MasterPage.Master file

```xml
<%@ Master language=""C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
<
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="_Head1" runat="server">
<title>Untitled.Title</title>
<link rel="stylesheet" type="text/css" href="/-/style/style.css"/>
</head>
<body>
<form id="Form1" runat="server">
<div class="content">
<table>
<tr>
<td>
<asp:SiteMapDataSource ID="SiteMapDataSource" Runat="server" />
</td>
</tr>
</table>
</div>
</form>
</body>
</html>
```
```csharp
private void CreateComponentTable()
{
    Dispose();
    TableRow row1 = new TableRow();
    TableRow row2 = new TableRow();
    TableRow row3 = new TableRow();

    tblQuestion.Width = 440;
    tblQuestion.Height = 175;

    row1.Height = 30;
    row2.Height = 50;
    row3.Height = 30;

    // If the table already has data in it, clear it out.
    if (tblQuestion.Rows.Count > 0)
    {
        RemoveRows();
    }

    // Set up the FIRST row
                                "center", "lblQuestionNo", 0, false));
    row1.Cells.Add(new BlankCell(5));
    tblQuestion.Rows.Add(row1);

    // Set up the SECOND row
    // Determine the control to add to the form
    switch (ControlRequired)
    {
        case ("TextBox"): row2.Cells.Add(new BlankCell(5));
            row2.Cells.Add(new ControlCell(_ctrlCreator.NewTextBox(), 2));
            tblQuestion.Rows.Add(row2);
            break;

        case ("DropdownList"): datasetDataSet1 = new Dataset();
            datasetDataSet1 = EleStationDAL.GetAnswerOptions(QuestionID);
            row2.Cells.Add(new BlankCell(5));
            row2.Cells.Add(new ControlCell(_ctrlCreator.NewDropDownList(datasetDataSet1), 2));
            tblQuestion.Rows.Add(row2);
            break;

        case ("RadioButton"): datasetDataSet2 = new Dataset();
            datasetDataSet2 = EleStationDAL.GetAnswerOptions(QuestionID);
            int j = 0;
            while (j < datasetDataSet2.Tables[0].Rows.Count)
            {
                row2 = new TableRow();
                row2.Cells.Add(new BlankCell(5));
               _tblQuestion.Table[0, j].Text.ToString();
                RadioButton rdo = new RadioButton();
                rdo = _ctrlCreator.NewRadioButton(text);
                tblQuestion.Rows.Add(row2);
                j++;
            }
            break;

        default: break;
    }
}
```
Figure D.33: Code: Snippet from the Web.Config file

```xml
<authorization>
  <allow users="*"/>
</authorization>

<!-- Membership Provider -->
<membership defaultProvider="MemProv">
  <providers>
      minRequiredNonalphabeticCharacters="1"
      connectionStringName="ConnectionString"
      requiresUniqueEmail="true"
      passwordFormat="Hashed"
      requiresQuestionAndAnswer="false"
      maxInvalidPasswordAttempts="10"
      minRequiredPasswordLength="8"
      passwordStrengthRegularExpression=""/>
  </providers>
</membership>

<!-- Role Provider -->
<roleManager enabled="true"
  defaultProvider="RoleProv">
  <providers>
    <add name="RoleProv" type="System.Web.Security.SqlRoleProvider"
      connectionStringName="ConnectionString"/>
  </providers>
</roleManager>
```
Figure D.34: Code: Method to return a cell containing a TextBox control

```java
private TableCell TextCell([Label label, string alignment, string id, int colspan, bool isBold])
{
    TableCell cell = new TableCell();
    cell.ColumnSpan = colspan;
    cell.EnableViewState = true;
    cell.Font.Name = "Arial";
    label.ID = id;
    label.Font.Bold = isBold;
    cell.Controls.Add(label);

    switch (alignment)
    {
    case "center":
        cell.HorizontalAlignment = HorizontalAlign.Center;
        break;
    case "left":
        cell.HorizontalAlignment = HorizontalAlign.Left;
        break;
    case "right":
        cell.HorizontalAlignment = HorizontalAlign.Right;
        break;
    default:
        break;
    }

    return cell;
}
```
Figure D.35: Code: Snippet from the Web.Sitemap file, showing the hierarchical structure of the menu

```xml
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">

  <siteMapNode url="/pages/Welcome.aspx" title="Home">
    <siteMapNode title="Administration" />
      <siteMapNode title="User Administration" />
        <siteMapNode title="Question Administration" />
          <siteMapNode url="/pages/AddQuestion.aspx" title="Add a Question" />
            <siteMapNode url="/pages/QuestionsWithPreConditions.aspx" title="View Questions with Pre-Conditions" />
          </siteMapNode>
        </siteMapNode>
      </siteMapNode>
    </siteMapNode>
  </siteMapNode>

  <siteMapNode title="Project Administration" />
    <siteMapNode title="Add a Project" />
      <siteMapNode title="Delete a Project" />
    </siteMapNode>
  </siteMapNode>

  <siteMapNode url="/pages/Reports.aspx" title="Reports" />
    <siteMapNode>
      <siteMapNode url="/pages/loggedout.aspx" title="Log Out" />
    </siteMapNode>
  </siteMapNode>

</siteMap>
```
body
{
  background-color: #C83F12;
}

h3
{
  font-family: Arial;
  color: #460254;
  font-size: 12pt;
  text-align: left;
  padding-left: 6px;
}

th
{
  font-family: Arial;
  font-size: 10pt;
  border-style: none;
  font-weight: bold;
}

td
{
  font-family: Arial;
  font-size: 10pt;
  border-style: none;
}

.content
{
  width: 650px;
  margin: auto;
  border-style: solid;
  border-width: thin;
  border-color: #2883B8;
  background-color: White;
  padding: 0px;
}

p
{
  font-family: Arial;
}
Figure D.37: Microsoft SQL Server 2005® database diagram, showing the tables...
Figure D.38: Microsoft Visual Studio 2005® class diagram, showing the classes implemented.
Figure D.39: Screenshot: Hashed User Credentials in the database

<table>
<thead>
<tr>
<th>UserId</th>
<th>Password</th>
<th>PasswordFormat</th>
<th>PasswordSalt</th>
</tr>
</thead>
<tbody>
<tr>
<td>57e550af-6df5-a48a-8e2e-665e625549</td>
<td>k3C2W5PmKzCyC09Bdv6oFyf/5k6y=</td>
<td>1</td>
<td>1p16GkayFDHGHMOfFlA==</td>
</tr>
<tr>
<td>9f0256</td>
<td>039045e-cddf-3cf5-e25b-21ed7276550bab</td>
<td>PAPPASDU81U9166670SHY30=</td>
<td>1</td>
</tr>
<tr>
<td>9f0256</td>
<td>5f58710b-6a45-1495-8b14-f894ae135ce3</td>
<td>F0vR9Q297a9F83Qzmb77+p+p+p=</td>
<td>1</td>
</tr>
<tr>
<td>9f0256</td>
<td>5b4b6d57-020c-4c8b-655c-d8e22dd8d23b</td>
<td>786N7b2xue25nffPAPR4q6jS6l=</td>
<td>1</td>
</tr>
</tbody>
</table>

null | null | null | null |
Appendix E

Testing

The test plan can be found on the following pages.
## Use Case Testing

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>Test Item (use case)</th>
<th>Required Input Specifications</th>
<th>Required Output Specifications</th>
<th>Environmental needs</th>
<th>Special Procedural requirements</th>
<th>Inter-case Dependencies</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1. Login</td>
<td>1.) A valid username and password supplied with Remember Me checked. 2.) Invalid username and password supplied with Remember Me checked. 3.) Attempted SQL injection using values 'OR true' = true.</td>
<td>1.) Successful login, response under 2 seconds. 2.) Unsuccessful login, response under 2 seconds. 3.) Should not login in to the system.</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1.) Successful. The valid credentials were accepted. 2.) Successful. The invalid credentials were not accepted and were asked to try again. 3.) Attempted SQL injection proved unsuccessful. The application rejected it as incorrect login credentials. Overall response rates well under 2 seconds.</td>
<td></td>
</tr>
<tr>
<td>U2. Logout</td>
<td>Click on the Log Out menu link.</td>
<td>If the user checked Remember Me during login, a page should be displayed offering to log the user out properly. If the user did not check Remember Me, a page confirming log out should be displayed. Should take no longer than 2 seconds.</td>
<td>N/A</td>
<td>Output is determined by whether Remember Me was checked during login.</td>
<td>- Login</td>
<td>Successful. When the application was asked to remember the user, Log Out occurred but asked if they wanted to log out fully. When Remember Me was not checked, Log Out page was displayed normally. Response rates well under 2 seconds.</td>
<td></td>
</tr>
<tr>
<td>U3. Add User</td>
<td>1.) Enter the username 'Paul', with the Role Limited User. Provide a password of 'CaFe4e5', with email address <a href="mailto:Paul@Work.com">Paul@Work.com</a> 2.) Enter the username 'Paul2', with the Role Limited</td>
<td>1.) User Paul should not be created and an error message stating Password length minimum: 8, Non-alphanumeric characters required 1 should be displayed. Should take no</td>
<td>N/A</td>
<td>- User clicks on the User Administration link on the menu  - User must be an Administrator to access User Administration</td>
<td>- Login</td>
<td>1.) Successful. The suggested error message appeared, password not accepted. 2.) Successful. This time the password was accepted.</td>
<td></td>
</tr>
<tr>
<td>Use Case</td>
<td>Description</td>
<td>Pass/Fail Criteria</td>
<td>Result</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U4. Change Password</td>
<td>User provides a password of 'CaSe52!'</td>
<td>Password should not be changed. Error message: 'The length of parameter 'newPassword' needs to be greater or equal to 8. Should take no longer than 2 seconds.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U5. Delete User</td>
<td>Select the username 'Paul2' from the Delete User drop-down list, click Delete button</td>
<td>The username 'Paul2' should be deleted from the database, confirmed in message. Should take no longer than 2 seconds.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U6. Add Project</td>
<td>On the Add a Project webpage, the user enters a blank Project name into the textbox provided.</td>
<td>JavaScript is invoked to display an error message saying: 'Please enter a Project to add to the database'.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U8. Add Question</td>
<td>On the Add a Question page, the user inputs no details and clicks the Add Question button.</td>
<td>JavaScript is invoked informing the user of the fields that need filling out. No question should be added.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use Case</td>
<td>Test Case Details</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| U9 Create Q&A Session | 1. The user selects from the Home page (Welcome page) no CH User Type (Child Health User Type).  
2. The user selects from the CH User Type drop down list 'Operational'. The Project selected should be 'Leeds'. The user then presses Continue.  
3. The user selects from the CH User Type drop down list 'Technical'. The Project selected should be 'Leeds'. The user then presses Continue.  
4. JavaScript is invoked which displays a message informing:  
- Please select a Project.  
- Please select your ‘Child Health User Type’.  
5. The Q&A Session begins, by asking the user the opening question of the survey for that particular CH User Type (Operational).  
6. The Q&A Session begins, by asking the user the opening question of the survey for that particular CH User Type (Technical). |
| N/A | User is on the Home Page. |
| 1. Successful. JavaScript is invoked, displaying the message outlined in the Required Output Specifications column.  
2. Successful. The Q&A Session was created for the selected User Type (Operational). Operational questions were then asked.  
3. Successful. The Q&A Session was created for the selected User Type (technical). Technical questions were then asked. |
| U10 Submit Answer | 1. The user provides an answer of type integer for a question requiring text.  
2. The user provides an answer of type text for a question requiring an integer.  
3. The user provides an answer of the required data type. |
| N/A | N/A |
| 1. The answer provided is not accepted. JavaScript is invoked, with the message: Input Error. This question requires int. Control is then returned to the question.  
2. The answer provided is not accepted. JavaScript is invoked, with the message: Input Error. This question requires int. Control is then returned to the question.  
3. The answer is accepted and the next question is displayed. |
| N/A | N/A |
| 1. Successful. The integer input into the textbox was not accepted. The suggested message appeared. The answer was not submitted.  
2. Successful. The text input into the textbox was not accepted. The suggested message appeared. The answer was not submitted.  
3. Successful. The answer provided was accepted.  
(More specific testing for this use case is explained in detail in the ‘Question Answering Testing’ table) |
<table>
<thead>
<tr>
<th>Test Case</th>
<th>Description</th>
<th>Pre-conditions</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>U11</td>
<td>Exit Q&amp;A Session</td>
<td>During the Q&amp;A Session, the Exit button is pressed.</td>
<td>All answers provided for that Q&amp;A Session are deleted.</td>
</tr>
<tr>
<td>U12</td>
<td>Generate Report</td>
<td>Within the system, reports are created in two ways: 1) By the Limited User by clicking on the View My Report link after completion of a Q&amp;A Session. 2) By the Administrator from the Reports page, by selecting the Project name from the drop down list and clicking on Generate.</td>
<td>1) After clicking on the link, a report will be generated consisting of all the questions and the answers that the user has provided during the Q&amp;A Session. Should take no longer than 2 seconds. 2) A report should be generated consisting of all the answers provided during the Q&amp;A Session for a Project.</td>
</tr>
<tr>
<td>U13</td>
<td>Get Help</td>
<td>1) During a Q&amp;A Session, click the Show Help link to display help to the users. 2) After help is displayed, click the Hide Help link.</td>
<td>1) Help to text (for the current question) appears at the bottom of the page. 2) Help to text (for the current question) disappears from the bottom of the page.</td>
</tr>
<tr>
<td>U14</td>
<td>Export Report</td>
<td>Below the report, an Export button is displayed. The user clicks the Export button.</td>
<td>The report of questions and answers for a Q&amp;A Session should be exported to the local spreadsheet application.</td>
</tr>
<tr>
<td>UI5</td>
<td>User Session</td>
<td>Attempt to access an administrative area whilst logged in as a Limited User.</td>
<td>Should permit access to the administrative area.</td>
</tr>
<tr>
<td>-----</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>UI5</td>
<td>Web Browser Testing</td>
<td>Load the web site in: 1) Microsoft Internet Explorer 2) Mozilla Firefox 3) Opera</td>
<td>The web pages displayed should not undergo any major differences, in presentation and/or functionality.</td>
</tr>
</tbody>
</table>

Successful. Logged in as a Limited User, the user was unable to access administrative areas. When tried, the page just refreshed.

Largely successful. There are no major alterations in functionality or presentation. However, when a report is run, results are aligned to the left in the table when using Mozilla Firefox, yet results are aligned to the right when using Microsoft Internet Explorer and Opera.

Results for the web browser test can be seen in Appendix x, which shows the Add a Question web page.
<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>Test Item</th>
<th>Required Input Specifications</th>
<th>Required Output Specifications</th>
<th>Environmental needs</th>
<th>Special procedural requirements</th>
<th>Interact dependencies</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>01</td>
<td>Question: &quot;How many sets of schedules does your current Child Health System run?&quot; Question Type: Operational Control Required: Text Box, Help Text Question 1 help. Expected Data: Integer Answer Options: N/A, Predecessors: N/A</td>
<td>Question data should be accepted and inserted into the relevant tables within the database. In the ELICITATION, PRECEDES, OR, table, there should be a NULL predecessor for the question.</td>
<td>N/A</td>
<td>User role must be Administrator</td>
<td>Login</td>
<td>Successful. The question was added into the database successfully.</td>
</tr>
<tr>
<td>Q2</td>
<td>02</td>
<td>Question: &quot;Are all these run from the same site?&quot; Question Type: Operational Control Required: Drop Down List, Help Text Question 2 help. Expected Data: N/A, Answer Options: Yes and No, Predecessors: Q1 &gt; 1</td>
<td>Question data should be accepted and inserted into the relevant tables within the database.</td>
<td>N/A</td>
<td>User role must be Administrator</td>
<td>Login</td>
<td>Successful. The question was added into the database successfully.</td>
</tr>
<tr>
<td>Q3</td>
<td>03</td>
<td>Question: &quot;Does each office or site see exactly the same view of the Child Health system (same software, same data)?&quot; Question Type: Technical Control Required: Drop Down List, Help Text Question 3 help. Expected Data: N/A, Answer Options: Yes and No, Predecessors: Q1 &gt; 1</td>
<td>Question data should be accepted and inserted into the relevant tables within the database.</td>
<td>N/A</td>
<td>User role must be Administrator</td>
<td>Login</td>
<td>Successful. The question was added into the database successfully. From a usability point of view, the user forgot to select a Question Type, an error which was caught by JavaScript.</td>
</tr>
<tr>
<td>Q4</td>
<td>04</td>
<td>Question: &quot;Does your current Child Health system contain ANY patient records from any other care settings like Community, Mental Health, ...&quot;</td>
<td>Question data should be accepted and inserted into the relevant tables within the database.</td>
<td>N/A</td>
<td>User role must be Administrator</td>
<td>Login</td>
<td>Successful. The question was added into the database successfully.</td>
</tr>
<tr>
<td>Q5</td>
<td>Q5</td>
<td>Question: &quot;Please list all the other care settings which gave records in the same system as Child Health.&quot;</td>
<td>Question data should be accepted and inserted into the relevant tables within the database.</td>
<td>N/A</td>
<td>User role must be Administrator</td>
<td>Login</td>
<td>Successful. The question was added into the database successfully.</td>
</tr>
<tr>
<td>Q6</td>
<td>Q6</td>
<td>Question: &quot;Please describe in as much detail as possible how you tell a Child Health record from those belonging to each of the other settings.&quot;</td>
<td>Question data should be accepted and inserted into the relevant tables within the database.</td>
<td>N/A</td>
<td>User role must be Administrator</td>
<td>Login</td>
<td>Successful. The question was added into the database successfully.</td>
</tr>
<tr>
<td>Q7</td>
<td>Q7</td>
<td>Question: &quot;Please enter the age at which a long patient ceases to be under the care of the Child Health unit.&quot;</td>
<td>Question data should be accepted and inserted into the relevant tables within the database.</td>
<td>N/A</td>
<td>User role must be Administrator</td>
<td>Login</td>
<td>Successful. The question was added into the database successfully.</td>
</tr>
<tr>
<td>Question</td>
<td>Question Type</td>
<td>Expected Data</td>
<td>Answer Options</td>
<td>User Role</td>
<td>Result</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>---------------</td>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C8: C8</td>
<td>Question: &quot;Does this age apply to patients with special needs?&quot;</td>
<td>Integer</td>
<td>N/A</td>
<td>Administrator</td>
<td>Successful. The question was added into the database successfully.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C9: C9</td>
<td>Question: &quot;What is the age for patients with special needs?&quot;</td>
<td>Integer</td>
<td>N/A</td>
<td>Administrator</td>
<td>Successful. The question was added into the database successfully.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C10: C10</td>
<td>Question: &quot;Please enter the registration status which indicates a birth which is transferred out?&quot;</td>
<td>String</td>
<td>N/A</td>
<td>Administrator</td>
<td>Successful. The question was added into the database successfully.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Question Answering Testing

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>Test Item</th>
<th>Required Input Specifications</th>
<th>Required Output Specifications</th>
<th>Environmental needs</th>
<th>Special Procedural Requirements</th>
<th>Interoperability Dependences</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Q1</td>
<td>1) Provide an answer of 0 to Q1 2) Provide an answer of 1 to Q1 3) Provide an answer of 2 to Q1 4) Attempt to enter a string</td>
<td>1) Answer accepted, Q4 is displayed 2) Answer accepted, Q4 is displayed 3) Answer accepted, Q2 is displayed 4) Answer not accepted, JavaScript message invoked informing user of incorrect data type</td>
<td>N/A</td>
<td>The user selects a Project and sets the CR User Type to 'Operational', thus proceeding to an operational session.</td>
<td>N/A</td>
<td>1) Successful, Answer was accepted and Q4 was displayed 2) Successful, Answer was accepted and Q4 was displayed 3) Successful, Answer was accepted and Q2 was displayed 4) Successful, Message box appeared informing user to input an integer</td>
</tr>
<tr>
<td>A2</td>
<td>Q2</td>
<td>1) Provide an answer of 'Yes' to Q2 2) Provide an answer of 'No' to Q2</td>
<td>1) Answer accepted, Q4 is displayed 2) Answer accepted, Q4 is displayed</td>
<td>N/A</td>
<td>The user selects a Project and sets the CR User Type to 'Operational', thus proceeding to an operational session.</td>
<td>N/A</td>
<td>1) Successful, Answer was accepted and Q4 was displayed 2) Successful, Answer was accepted and Q4 was displayed 3) Successful, Answer was accepted and Q4 was displayed 4) Successful, Message box appeared informing user to input an integer</td>
</tr>
<tr>
<td>A3</td>
<td>Q3</td>
<td>1) Provide an answer of 'Yes' to Q3 2) Provide an answer of 'No' to Q3</td>
<td>1) Q4 predecessor is satisfied but Q4 is not displayed since it is not a technical question 2) Q4 predecessor is satisfied but Q4 is not displayed since it is not a technical question</td>
<td>N/A</td>
<td>The user selects a Project and sets the CR User Type to 'Technical', thus proceeding to a technical session.</td>
<td>N/A</td>
<td>1) Successful, Answer was accepted, survey ended 2) Successful, Answer was accepted, survey ended 3) Successful, Answer was accepted, survey ended 4) Successful, Answer was accepted, survey ended 5) However, as the question has a predecessor of an operational question, the question was still asked, even when the predecessor was not fulfilled.</td>
</tr>
<tr>
<td>A4</td>
<td>Q4</td>
<td>1) Provide an answer of 'Yes'</td>
<td>1) Answer accepted, Q5 is</td>
<td>N/A</td>
<td>The user selects a</td>
<td>Login</td>
<td>1) Successful, Answer</td>
</tr>
</tbody>
</table>
| A5 | G5 | 1.) Provide an answer of nothing to G6  
2.) Provide a non-null answer to G5  
3.) Enter an integer (18) as the answer to G7  
4.) Provide a floating-point answer to G7 | 1.) Answer accepted, G6 is displayed  
2.) Answer accepted, G6 is displayed | N/A | Project and sets the CH User Type to 'Operational', thus processing to an operational session. | Add a Project  
Add a Question | 1.) Successful Answer was accepted and G6 was displayed  
2.) Successful Answer was accepted and G6 was displayed  
3.) Successful Answer was accepted and G6 was displayed  
4.) Successful Answer was accepted and G6 was displayed |
| A6 | G6 | 1.) Provide an answer of nothing to G6  
2.) Provide a non-null answer to G6 | 1.) Answer accepted, G7 is displayed  
2.) Answer accepted, G7 is displayed | N/A | The user selects a Project and sets the CH User Type to 'Operational', thus processing to an operational session. | Add a Project  
Add a Question | 1.) Successful Answer was accepted and G7 was displayed  
2.) Successful Answer was accepted and G7 was displayed |
| A7 | G7 | 1.) Provide a string of text as an answer to G7  
2.) Provide a null answer for G7  
3.) Enter an integer (18) as the answer to G7  
4.) Provide a floating-point answer to G7 | 1.) Answer not accepted, JavaScript message invoked, informing the user of incorrect data type  
2.) Answer accepted, G10 is displayed  
3.) Answer accepted, G10 is displayed  
4.) Answer not accepted, JavaScript message invoked, informing the user of incorrect data type | N/A | The user selects a Project and sets the CH User Type to 'Operational', thus processing to an operational session. | Add a Project  
Add a Question | 1.) Successful Answer was not accepted and a message appeared informing the user to enter an integer  
2.) Successful Answer was accepted, G10 was displayed  
3.) Successful Answer was accepted and G10 was displayed  
4.) Successful Answer was not accepted and a message appeared informing the user to enter an integer |
| A8 | G8 | 1.) Provide an answer of 'Yes' to G8  
2.) Provide an answer of 'No' to G8 | 1.) Answer accepted, G10 is displayed  
2.) Answer accepted, G10 is displayed | N/A | The user selects a Project and sets the CH User Type to 'Operational', thus processing to an operational session. | Add a Project  
Add a Question | 1.) Successful Answer was accepted and G10 was displayed  
2.) Successful Answer was accepted and G10 was displayed  
3.) Successful Answer was accepted and G10 was displayed  
4.) Successful Answer was accepted and G10 was displayed |
| A9 | Q9 | 1. Enter a string as the answer to Q9  
    2. Enter a floating-point number as the answer to Q9  
    3. Provide an integer as the answer to Q9 | 1. Answer not accepted, JavaScript message invoked, informing the user of incorrect data type  
    2. Answer not accepted, JavaScript message invoked, informing the user of incorrect data type  
    3. Answer accepted, Q10 is displayed | N/A | The user selects a Project and sets the Q9 User Type to 'Operational', thus proceeding to an operational session | Login Add a Project Add a Question | 1. Successful, Answer was accepted, Q9 was displayed |
| A10 | Q10 | 1. Enter an integer as the answer to Q10  
    2. Enter a string as the answer to Q10 | 1. Answer not accepted, JavaScript message invoked, informing the user of incorrect data type  
    2. Answer accepted, survey finished | N/A | The user selects a Project and sets the Q9 User Type to 'Operational', thus proceeding to an operational session | Login Add a Project Add a Question | 1. Successful, Answer was not accepted and a message appeared informing the user to enter an integer  
   2. Successful, Answer was accepted and Q9 was displayed |
Figure E.12: User Acceptance Test Plan (Page 1)

User Testing Session

Administrator Tasks

1. User administration

The following tasks are designed to allow an administrator to create a new user, change their password and delete that user in the system. Whilst carrying out the tasks the administrator is asked to provide feedback to the developer (sitting alongside) regarding the tasks and any points they wish to raise about the processes in the table at the end of this document. Furthermore, please report any bugs or errors you encounter whilst performing these tasks.

Please perform the following actions:

- Add a new limited user into the system. Provide them with a relevant password and email address.
- Now change that user’s password.
- Now delete that user.

Feedback table

<table>
<thead>
<tr>
<th>Task</th>
<th>Comments</th>
<th>Errors/Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation</td>
<td>The client could find the User Administration screen with ease.</td>
<td>None.</td>
</tr>
<tr>
<td>Adding a user</td>
<td>- From usability point of view, the user did not realise the continue button should be pressed - before carrying on. However, the client successfully added a user, after being prompted.</td>
<td>None.</td>
</tr>
<tr>
<td>Changing a user's password</td>
<td>The client successfully changed a password with no issues.</td>
<td>None.</td>
</tr>
<tr>
<td>Deleting a user</td>
<td>The client successfully deleted a user account with no issues.</td>
<td>None.</td>
</tr>
</tbody>
</table>

2. Project Administration

The following tasks are designed to allow an administrator to add and delete projects from the system. Once again, whilst carrying out the tasks the user is asked to provide any comments to the developer (sitting alongside) regarding the process (navigation, interface, usability etc) or errors encountered.
Figure E.13: User Acceptance Test Plan (Page 2)

- Add a new project into the system
- Now delete this project from the system

Feedback table

<table>
<thead>
<tr>
<th>Task</th>
<th>Comments</th>
<th>Errors/Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation</td>
<td>No issues in finding the project administration menu</td>
<td>None</td>
</tr>
<tr>
<td>Adding a Project</td>
<td>The client successfully added a project with no issues.</td>
<td>None</td>
</tr>
<tr>
<td>Deleting a Project</td>
<td>The client successfully deleted a project with no issues.</td>
<td>None</td>
</tr>
</tbody>
</table>

3. Question Administration

The following tasks are designed to allow the user to enter a set of questions (below), including (if required) answer options and predecessors. Whilst performing these tasks, the user is expected to provide comments/feedback to the developer (sitting alongside) on each task regarding its process, usability, or the general interface itself. Further to this, the user may provide other comments which they feel need to be made.

Please enter the following questions into the system, along with their details (outlined underneath).
(N.B. That these questions are specific to the data migration process of the Child Health system deployed by CSC)

<table>
<thead>
<tr>
<th>No.</th>
<th>Question Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>How many sets of schedules does your current Child Health system currently run?</td>
</tr>
<tr>
<td></td>
<td>Question Type: Operational</td>
</tr>
<tr>
<td></td>
<td>Control Required: Textbox</td>
</tr>
<tr>
<td></td>
<td>Expected Data: Integer</td>
</tr>
<tr>
<td></td>
<td>Answer Options: N/A</td>
</tr>
<tr>
<td></td>
<td>Predecessors: This question does not require predecessors</td>
</tr>
<tr>
<td>2.</td>
<td>Are all these run from the same site?</td>
</tr>
<tr>
<td></td>
<td>Question Type: Operational</td>
</tr>
<tr>
<td></td>
<td>Control Required: Drop down list</td>
</tr>
<tr>
<td></td>
<td>Expected Data: N/A</td>
</tr>
</tbody>
</table>
Figure E.14: User Acceptance Test Plan (Page 3)

<table>
<thead>
<tr>
<th>Question</th>
<th>Type</th>
<th>Control Required</th>
<th>Expected Data</th>
<th>Answer Options</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Does each office or site see exactly the same view of the Child Health system (same software, same data)?</td>
<td>Technical</td>
<td>Drop down list</td>
<td>N/A</td>
<td>Yes and No</td>
<td>Q1 &gt; 1</td>
</tr>
<tr>
<td>4. Does your current Child Health system contain any patient records from any other care settings (e.g. Community, Mental Health, Acute Care)?</td>
<td>Operational</td>
<td>Drop down list</td>
<td>N/A</td>
<td>Yes and No</td>
<td>Q1 = 0 OR Q1 = 1 OR Q2 = &quot;Yes&quot;</td>
</tr>
<tr>
<td>5. Please list all the other care settings which have records in the same system as Child Health.</td>
<td>Operational</td>
<td>Textbox</td>
<td>String</td>
<td>N/A</td>
<td>Q4 = &quot;Yes&quot;</td>
</tr>
</tbody>
</table>

Feedback table

<table>
<thead>
<tr>
<th>Task</th>
<th>Comments</th>
<th>Errors/Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation</td>
<td>The user could find the &quot;Add a Question&quot; page easily.</td>
<td>None</td>
</tr>
<tr>
<td>Creating a new question in the system. Includes selecting the question type, selecting the required control and the expected data.</td>
<td>- No major issues regarding usability - Fact that predecessors need not be required could be made more obvious - Sometimes easy to miss the Question Type drop down list</td>
<td>None</td>
</tr>
<tr>
<td>Process of assigning answer options to a question.</td>
<td>- Client thought that a control to automatically add a &quot;Yes/No&quot; answer option should be placed on the form.</td>
<td>None</td>
</tr>
<tr>
<td>Process of assigning the question a single or a</td>
<td>- No issues with assigning predecessors - User managed to enter the same</td>
<td>None</td>
</tr>
</tbody>
</table>
4. Report generation

The following tasks will result in a generated report, which may be exported to a spreadsheet. Please generate a report for a project (where questions have been answered) and attempt to export it to a spreadsheet, using the export feature. As with previous tasks, please provide feedback to the developer (sitting alongside) regarding the report generation process, including any information that may be missing from the report, its format or any bugs/errors which arise.

- On the menu go to Administration → Reports
- Select a project for which questions have been answered and click Generate.

Feedback table

<table>
<thead>
<tr>
<th>Task</th>
<th>Comments</th>
<th>Errors/Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation</td>
<td>The client could find the Report menu item with ease.</td>
<td>None</td>
</tr>
<tr>
<td>Report issues (missing information?)</td>
<td>No issues with the report, client mentioned all the information required was apparent.</td>
<td>None</td>
</tr>
</tbody>
</table>
Limited User Tasks

1. **Starting a Q&A Session**

The proceeding tasks will allow the user to test and evaluate the question and answering process handled by the system. Following from the previous section, where an administrator entered questions into the system, this section deals with answering these questions.

Whilst answering these questions, please provide appropriate feedback to the developer (sitting alongside), regarding the design of the interface or any bugs which are encountered.

To begin the process, start a Q&A Session by navigating to the Home page and selecting a CH User type (the users' affiliation with the Child Health system) and the Project in which the user is working on.

**Feedback table**

<table>
<thead>
<tr>
<th>Task</th>
<th>User comments</th>
<th>Errors/Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation</td>
<td>No real navigational issues. After time to read and think, the client was able to start a Q&amp;A Session.</td>
<td>None</td>
</tr>
<tr>
<td>Creating a Q&amp;A session</td>
<td>No issues in starting a Q&amp;A Session</td>
<td>None</td>
</tr>
<tr>
<td>Answering questions</td>
<td>No issues were found when answering the questions.</td>
<td>None</td>
</tr>
<tr>
<td>Getting help</td>
<td>The client found the &quot;Show Help&quot; link easily and said they were happy with the help text's position.</td>
<td>None</td>
</tr>
<tr>
<td>Viewing the report and exporting it</td>
<td>Client mentioned no issues with creating and exporting the report.</td>
<td>None</td>
</tr>
</tbody>
</table>
Appendix F

Evaluation

Listed on the following pages are a set of users whom after testing the application were asked by the developer to comment on the Nielsen Usability Heuristics found in [21].
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>Yes — good introductory information.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes — good use of colour and layout.</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>User 2</td>
<td>Generally yes — but give the user instructions for password conditions.</td>
<td>Mostly — Error message regarding incorrect input for a question could be more helpful in terms of what it wants.</td>
<td>Yes</td>
<td>Yes — Very consistent</td>
<td>N/A</td>
<td>Yes — clearly</td>
<td>N/A</td>
<td>Mostly see previous point regarding “Speaking Users Language”</td>
<td>N/A</td>
<td>Yes — easy access to help</td>
</tr>
<tr>
<td>User 3</td>
<td>Yes — reasonably clear dialogue on every page.</td>
<td>Yes — though error messages could be more user friendly</td>
<td>Mostly — found there was a lot to bear in mind when adding questions.</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes — Log out on the menu and Exit button during the question and answer session.</td>
<td>N/A</td>
<td>Yes — seem to be used in the right places.</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>User 4</td>
<td>Yes — but give the user instructions for password conditions.</td>
<td>Yes — but not necessarily relevant. Still explanatory.</td>
<td>Yes — easy to use</td>
<td>N/A</td>
<td>User did not notice the log out button on the menu.</td>
<td>N/A</td>
<td>Didn’t get any</td>
<td>N/A</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>User 5</td>
<td>Yes – but give the user instructions for password conditions.</td>
<td>Yes – but think this is determined by the capability of the user, some areas suggest you need a good knowledge and others not so.</td>
<td>Yes – Fresh feel doesn’t come across as difficult to use.</td>
<td>N/A</td>
<td>User didn’t think this applied.</td>
<td>N/A</td>
<td>Mostly – Error message regarding incorrect input for a question could be more helpful in terms of what it wants.</td>
<td>N/A</td>
<td>Yes.</td>
<td></td>
</tr>
<tr>
<td>User 6</td>
<td>Yes – no real issue</td>
<td>Yes – not great masses of information to remember</td>
<td>Yes – nice look, consistent throughout</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes – Messages boxes which appeared were largely useful but the message informing the user to input a string could be clearer.</td>
<td>N/A</td>
<td>The help for the question was adequate. However, help would be useful when creating a survey (Add a question)</td>
<td></td>
</tr>
<tr>
<td>User 7</td>
<td>Yes – simple to read. Largely yes but see – “Good Use of error messages column”</td>
<td>To an extent, though there was a reasonable amount to consider when adding a question: Question, Predecessors, options etc.</td>
<td>Yes very much so.</td>
<td>N/A</td>
<td>Yes – menu was clear</td>
<td>N/A</td>
<td>Yes – useful though some were bit too technical – e.g. “Input Error: This question requires a string”. What’s a string?</td>
<td>N/A</td>
<td>Yes – clearly marked</td>
<td></td>
</tr>
<tr>
<td>User 8</td>
<td>There was no irrelevant information. Yes – All understandable but is not much to remember.</td>
<td></td>
<td>Yes</td>
<td>N/A</td>
<td>Yes – able to exit a question</td>
<td>N/A</td>
<td>Only got one and that was informative</td>
<td>N/A</td>
<td>Yes – clearly marked</td>
<td></td>
</tr>
<tr>
<td>User 9</td>
<td>dependent upon the user's knowledge</td>
<td>Yes</td>
<td>Yes – mostly</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
<td>found it easy to find</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------</td>
<td>-----</td>
<td>---------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------------------</td>
</tr>
<tr>
<td>User 10</td>
<td>Parts are too technical for someone setting up</td>
<td>Not really – found some of the test too specific to the system.</td>
<td>Get used to where things are after one or two goes</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes – not many encountered though</td>
<td>N/A</td>
</tr>
<tr>
<td>Default (Login)</td>
<td>Welcome</td>
<td>Add Question</td>
<td>Add Project</td>
<td>User Admin</td>
<td>Delete Project</td>
<td>Reports</td>
<td>My Report</td>
<td>Question and Answer Session</td>
<td>Questions with Pre-Conditions</td>
<td>Log Out</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------------</td>
<td>---------</td>
<td>-----------</td>
<td>-----------------------------</td>
<td>-------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Default (Login)</td>
<td>N/A</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>Dependent on number of questions</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Welcome</td>
<td>2</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Dependent on number of questions</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Add Question</td>
<td>2</td>
<td>1</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Dependent on number of questions</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Add Project</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Dependent on number of questions</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>User Admin</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>Dependent on number of questions</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Delete Project</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
<td>1</td>
<td>Dependent on number of questions</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Reports</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
<td>Dependent on number of questions</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>My Report</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Question and Answer Session</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Dependent on number of questions</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>Questions with Pre-Conditions</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Dependent on number of questions</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>Log Out</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>Dependent on number of questions</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Appendix G

Project Management

The project schedule can be found on the following page
Figure G.1: Project Schedule (Microsoft Project 2005®)

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Physical % Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. PB02: Final Year Project</td>
<td>444 days</td>
<td>Wed 24/10/07</td>
<td>Wed 23/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.1. Requirements Capture</td>
<td>13 days</td>
<td>Wed 24/10/07</td>
<td>Thu 08/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.1.1. Meet with client to discuss requirements</td>
<td>1 day</td>
<td>Mon 29/10/07</td>
<td>Mon 29/10/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.1.2. Methodology Research Completed</td>
<td>8 days</td>
<td>Wed 24/10/07</td>
<td>Wed 31/10/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.1.3. Technology Research Completed</td>
<td>8 days</td>
<td>Wed 24/10/07</td>
<td>Wed 31/10/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.4. Research Current Solution Completed</td>
<td>3 days</td>
<td>Wed 31/10/07</td>
<td>Fri 02/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.5. Write up requirements Completed</td>
<td>5 days</td>
<td>Sat 03/11/07</td>
<td>Thu 09/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.2. Requirements Analysis</td>
<td>7 days</td>
<td>Mon 05/11/07</td>
<td>Mon 12/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.2.1. Use Cases Completed</td>
<td>1 day</td>
<td>Mon 05/11/07</td>
<td>Mon 05/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.2.2. Activity Diagram Completed</td>
<td>1 day</td>
<td>Tue 06/11/07</td>
<td>Tue 06/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.2.3. Must have Use Cases Completed</td>
<td>1 day</td>
<td>Wed 07/11/07</td>
<td>Wed 07/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.2.4. Should have Use Cases Completed</td>
<td>1 day</td>
<td>Thu 08/11/07</td>
<td>Thu 08/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.2.5. Could have Use Cases Completed</td>
<td>1 day</td>
<td>Sat 10/11/07</td>
<td>Sat 10/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.2.6. Would have Use Cases Completed</td>
<td>1 day</td>
<td>Mon 12/11/07</td>
<td>Mon 12/11/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.3. System Design</td>
<td>37 days</td>
<td>Sat 03/12/07</td>
<td>Tue 22/01/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.3.1. Elicitation Algorithm Design Iteration 1 Completed</td>
<td>3 days</td>
<td>Sat 03/12/07</td>
<td>Tue 11/12/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.3.2. Elicitation Algorithm Design Iteration 2 Completed</td>
<td>5 days</td>
<td>Sat 11/12/07</td>
<td>Sat 18/12/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.3.3. Elicitation Algorithm Design Iteration 3 Completed</td>
<td>3 days</td>
<td>Sat 18/12/07</td>
<td>Sat 25/12/07</td>
<td>100%</td>
</tr>
<tr>
<td>1.3.4. Database Design Completed</td>
<td>2 days</td>
<td>Thu 10/01/08</td>
<td>Fri 17/01/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.3.5. Entity Relationship Diagram Completed</td>
<td>2 days</td>
<td>Thu 10/01/08</td>
<td>Fri 17/01/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.3.6. Interface Prototype Completed</td>
<td>1 day</td>
<td>Tue 22/01/08</td>
<td>Tue 22/01/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.3.7. Meet with Client to Sign-off design</td>
<td>1 day</td>
<td>Tue 22/01/08</td>
<td>Tue 22/01/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.4. Implementation</td>
<td>59 days</td>
<td>Wed 23/01/08</td>
<td>Fri 11/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.4.1. Solid Database Completed</td>
<td>1 day</td>
<td>Wed 23/01/08</td>
<td>Wed 25/01/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.4.2. Solid SQL Overview Completed</td>
<td>9 days</td>
<td>Wed 23/01/08</td>
<td>Mon 04/02/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.4.3. Code User Interface Completed</td>
<td>30 days</td>
<td>Wed 08/02/08</td>
<td>Tue 15/03/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.4.4. Code Entity Classes Completed</td>
<td>49 days</td>
<td>Wed 08/02/08</td>
<td>Fri 13/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.5. Testing</td>
<td>7 days</td>
<td>Thu 10/04/08</td>
<td>Fri 18/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.5.1. Developer testing Completed</td>
<td>6 days</td>
<td>Fri 11/04/08</td>
<td>Fri 18/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.5.2. Client Acceptance testing Completed</td>
<td>1 day</td>
<td>Thu 10/04/08</td>
<td>Thu 10/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.6. Report Write-Up</td>
<td>30 days</td>
<td>Mon 17/03/08</td>
<td>Wed 23/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.6.1. Chapter Introduction Completed</td>
<td>3 days</td>
<td>Mon 17/03/08</td>
<td>Wed 19/03/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.6.2. Chapter Methodology and Technology Completed</td>
<td>6 days</td>
<td>Mon 17/03/08</td>
<td>Mon 24/03/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.6.3. Chapter System Design Completed</td>
<td>7 days</td>
<td>Mon 24/03/08</td>
<td>Mon 31/03/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.6.4. Chapter Implementation Completed</td>
<td>5 days</td>
<td>Tue 01/04/08</td>
<td>Mon 07/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.6.5. Chapter Testing Completed</td>
<td>5 days</td>
<td>Mon 07/04/08</td>
<td>Mon 14/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.6.6. Chapter: Evaluation Completed</td>
<td>5 days</td>
<td>Tue 15/04/08</td>
<td>Sat 19/04/08</td>
<td>100%</td>
</tr>
<tr>
<td>1.6.7. Chapter: Appendices and Read-through completed</td>
<td>30 days</td>
<td>Mon 17/03/08</td>
<td>Wed 25/04/08</td>
<td>100%</td>
</tr>
</tbody>
</table>