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 objective of this project is to create a system that displays information relating to the performance of the School of Computing. Initially, a list of statistics that are appropriate to display in the system must be constructed. Historical information, held within various electronic files for various statistics, must also be incorporated with current data in the system. Output should be in the form of dynamically generated images where appropriate. The development is intended to be an extension to the current system in use on the School of Computing Intranet, and may be used by any member of the department.

A web-based system has been implemented and incorporates statistics that reflect upon a variety of areas of School activity. Historical data has been fully integrated and is retrieved alongside current data. The display of dynamically generated images, constructed from the retrieved data, is also managed by the system. As a result of this, the implemented system meets the original objectives of the project and surpasses the minimum requirements. Functionality is added to the system through selectable date ranges available, where appropriate, on statistics.
Acknowledgements

Primarily, I must thank my supervisor Dr. Les Proll whose guidance and help with any problems encountered has been invaluable. The technical assistance and information received from Jonathan Ainsworth has also proven helpful and saved a great deal of time during the implementation phase. The completion of this project owes a great deal to these two people.

I have to thank my parents for their support throughout the duration of this project. Further, I must acknowledge the hours my mother spent proof reading this report.

Additionally, I must thank Dr. John Davy and Dr. Kevin McEvoy for the help and assistance they gave during the period of my illness and on my return to University. I hope I did not create too much extra work for you!

Finally, I must thank the many friends whose support has been a great source of strength to me. Predominantly Neil Rickus and Chris O’Hara, whose untiring assistance this year has been incalculable and I only hope I was able to return the help!
# Contents

Summary...................................................................................................................................i
Acknowledgements..................................................................................................................ii
Contents..................................................................................................................................iii

**Chapter 1 – Introduction** ...................................................................................................... 1
1.1 Statement Of The Problem ............................................................................................... 1
1.2 Aims & Objectives ............................................................................................................. 1
1.3 Minimum Requirements ................................................................................................. 2
1.4 Evaluation Criteria............................................................................................................. 2
1.5 Project Schedule ............................................................................................................... 2

**Chapter 2 – Research And Analysis** ................................................................................... 5
2.1 Introduction ....................................................................................................................... 5
2.2 Past Project Analysis ........................................................................................................ 5
2.3 Methodology ..................................................................................................................... 6
2.4 Implementation Tools ........................................................................................................ 7
   2.4.1 System Front-End ...................................................................................................... 8
   2.4.2 Incorporating Historical Data .................................................................................... 10
2.5 Users And Their Requirements ....................................................................................... 10
   2.5.1 Must Have Requirements ......................................................................................... 11
   2.5.2 Should Have Requirements ....................................................................................... 11
   2.5.3 Could Have Requirements ....................................................................................... 11
2.6 Statistics To Display ........................................................................................................ 11
   2.6.1 Research Students .................................................................................................. 12
   2.6.2 Taught Students ....................................................................................................... 12
   2.6.3 Publications .............................................................................................................. 12
   2.6.4 Research Contract ................................................................................................... 13
   2.6.5 Full Time Equivalents .............................................................................................. 13

**Chapter 3 – System Design** ................................................................................................ 15
3.1 Introduction ..................................................................................................................... 15
3.2 Database Design ............................................................................................................. 15
   3.2.1 Graduate Table ........................................................................................................ 15
   3.2.2 Degree Table ........................................................................................................... 16
   3.2.3 Full Time Equivalents Table .................................................................................. 17
3.3 Interface Design .............................................................................................................. 18
3.3.1 Human-Computer Interaction Issues .................................................. 18
3.3.2 Time-Sequence Diagram .................................................................. 19
3.3.3 Prototype Interfaces ....................................................................... 20
3.3.4 Script Execution ............................................................................. 21
3.4 Statistic Design ................................................................................... 22
3.4.1 Research Students .......................................................................... 22
3.4.2 Taught Students ............................................................................. 22
3.4.3 Publications .................................................................................... 24
3.4.4 Research Contract .......................................................................... 24
3.4.5 Full Time Equivalents ..................................................................... 25

Chapter 4 – Implementation ...................................................................... 27
4.1 Introduction ........................................................................................ 27
4.2 Database Implementation ................................................................... 27
  4.2.1 Data Cleansing .............................................................................. 27
  4.2.2 Table Creation ............................................................................... 27
  4.2.3 Importing Data ............................................................................ 28
4.3 Statistic Implementation ...................................................................... 28
  4.3.1 Script Security .............................................................................. 28
  4.3.2 Research Students ....................................................................... 29
  4.3.3 Statistics Menu Interface ............................................................... 31
  4.3.4 Taught Students .......................................................................... 31
  4.3.5 Publications .................................................................................. 34
  4.3.6 Research Contract ....................................................................... 35
  4.3.7 Full Time Equivalents .................................................................. 37

Chapter 5 – Testing .................................................................................. 39
5.1 Introduction ........................................................................................ 39
5.2 Validation ........................................................................................... 39
  5.2.1 Query Testing ................................................................................. 39
  5.2.2 Black-box Testing ......................................................................... 40
  5.2.3 Browser Testing ........................................................................... 41
5.3 Verification .......................................................................................... 41

Chapter 6 – Evaluation .............................................................................. 43
6.1 Introduction ........................................................................................ 43
6.2 Requirements Evaluation .................................................................... 43
6.3 Usability Evaluation ........................................................................... 45
  6.3.1 Usability Testing ......................................................................... 45
Chapter 1 – Introduction

1.1 Statement Of The Problem

The School of Computing is a large department within the University of Leeds. The School holds a considerable amount of information that indicates many aspects of its academic performance levels over many years. This information is not, however, made available to either staff or students in any form and is currently held implicitly in the School’s Information System or in static files such as spreadsheets.

The School Information System (known as SIS) is “based on a central, powerful, relational database,” Ainsworth, (2002). The database is implemented using the PostgreSQL database management system. Access to the database is mainly through a web interface, restricted to users of the School of Computing Intranet. Also, possible uses of the database are restricted further as data that is either personal or considered sensitive is held within it. SIS supports the management of School staff, taught students, research students, research activities and room booking. It also drives much of the School’s Internet and Intranet sites.

1.2 Aims & Objectives

The aim of this project is to implement a system that extends the current School of Computing Intranet website to incorporate certain performance indicators. The indicators cover a variety of criteria that reflect on the School’s performance. Additionally, where appropriate, statistics are presented in a graphical format. The statistics are collated at runtime and all system output is generated dynamically.

The following objectives are identified for this project:

1. Identify a set of performance indicators, which would be appropriate to display on the School of Computing Intranet.
2. Implement Perl scripts that dynamically collate data for the statistics and display it in the appropriate form.
3. Extend those scripts to gather historical data from static files, involving possible extension to the SIS database.
1.3 Minimum Requirements

The minimum requirements agreed for this project are:

1. To develop a system that gathers statistics in a textual form on a variety of criteria relating to research and teaching within the School of Computing.
2. Extend the developed system to dynamically present those statistics graphically on the School Intranet where appropriate.
3. Provide a mechanism to incorporate historical data held in a variety of electronic forms.

1.4 Evaluation Criteria

Evaluation will take place on both the developed software and the project as a whole. Basic evaluation criteria for the software, in the form of some simple questions, are outlined here. These criteria were formulated early in the project using the project objectives and the minimum requirements. This enables the progress of the system to be constantly assessed during development.

- Does the system work without producing errors and on the different web browser software available within the School of Computing and the University of Leeds?
- Is the choice of language appropriate?
- Does the system take account of any security issues with the development language?
- Does the system return the correct figures for each statistic and is historical data fully integrated and returned when requested? Incorrect figures indicate errors within the program.
- Are any graphs produced by the system clear and easy to read?
- Do users find it easy and intuitive to use the system? No user manual is available for web-based systems, so this is essential.
- Is the system code written in a way that makes it easily readable and extendable in the future?

1.5 Project Schedule

The Gantt chart found in Appendix B shows the project schedule in greater detail, for instance holidays and the semester 1 examination period (where no project work is completed) are shown. The project starts in October, 2002, and is divided into a number of
phases, each of which is listed, along with the number of weeks it is intended to take, below (some phases run concurrently which is shown on the Gantt chart, along with the milestones identified in the project).

Project Planning and Analysis (3 weeks)
This is the beginning phase of the project and concentrates on producing a realistic plan for completion. Background research is primarily required on development methodologies for use in the project, although upon completion investigation into development tools and the performance criteria appropriate for collating and displaying by the system can be performed.

Requirements Analysis and Definition (3 weeks)
This phase concentrates on defining the user requirements of the proposed system. The presentation of these requirements relies on the methodology research performed in the previous phase.

System Design (4 weeks)
This phase concentrates on designing the interface for the system, any tables that historical data are to be read into and the graphs that are produced for each statistic. As part of the interface design research into human-computer interaction is also required to ensure the design of the interface and the various graphs are of a high quality.

Implementation (8 weeks)
This phase concentrates on implementing the system designed in the previous phase of the project. Extra time is included in this phase to allow for any unfamiliarity with the tools being used during development.

Testing (3 weeks)
This phase concentrates on testing the developed system, both for errors and also for user acceptance. It runs approximately concurrent with the implementation phase because the developed system can be tested as it is implemented. Once implementation is completed some final tests can be run on the system, such as altering the data to ensure any changes are reflected in the collated statistics.
Evaluation (2 weeks)
This phase concentrates on assessing whether the implemented system actually meets the objectives and minimum requirements of the project and the evaluation criteria outlined in this chapter of the report.

Project Write-up (6 weeks)
This phase concentrates on the production of the final project report. Work on this phase of the project can begin when the implementation and testing phases are nearing completion and the quantity of work required in these phases decreases.
Chapter 2 – Research And Analysis

2.1 Introduction

The aim of this project is to develop a system to gather and display statistics reflecting the performance of the School of Computing. In order for this to be achieved first an appropriate methodology needs to be selected and followed, and the actual tools to be used need to be selected. This chapter presents the information gathered in order to achieve this.

2.2 Past Project Analysis

Perhaps one of the most important questions to be asked when attempting the development of any software is “has anything similar been developed before, and can I learn anything from it?” Unfortunately a development of this nature has not previously been attempted in the School. There are, however, a number of projects from the previous two years that are of interest. Although not directly related to this project, each has some relevance to what is being undertaken. This is described below.

Fiddy, (2001), has the most relevance to this project in that access was required to SIS and a scripting language was used to dynamically retrieve (and store) data and generate web pages. Although a very different system was produced, exactly the same tools were used therefore descriptions of any problems encountered prove useful.

McGuinness, (2001), deals with the use of information systems within the School. This project is two years old and the information contained within it regarding SIS is now out of date and inaccurate, thus limiting the project report’s use. However the suggested improvements in the use of information systems, such as increased information sharing, are an aid to understanding the motivation for solving the current problem.

Kelly, (2002), deals with creating a web-based system similar to Xfaces, in use within the School, for other departments within the University of Leeds. The final system created is completely unrelated. However, the research undertaken proves very useful in indicating where to start researching a solution to the problem in this project.
2.3 Methodology

Many development methodologies exist, each with their own advantages and disadvantages for different types of project. The importance of adopting a methodology is highlighted in Avison and Fitzgerald, (2003), where it is argued that the adoption leads to a better product and a better development process. There are several obvious methodologies that could be used in this project. These are outlined in Table 1 below, along with the advantages and disadvantages found with each one.

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘Waterfall’, Avgerou and Cornford, (1998).</td>
<td>Strictly follows the software development lifecycle.</td>
<td>Each stage must be complete before proceeding to the next and then does not allow a return to the preceding stage. Problems are harder to fix the later they are discovered. Not suited to larger projects.</td>
</tr>
<tr>
<td>‘Rapid Application Development’, Avison and Fitzgerald, (2003).</td>
<td>Iterative technique and development of prototypes that are tested by the user provides a definitive list of requirements.</td>
<td>Requires high level of user contact to assess the prototypes. Can be used in projects of any size.</td>
</tr>
</tbody>
</table>

The arguments presented in Avgerou and Cornford, (1998), coupled with the fact that unanticipated changes in the requirements of the system are hard to incorporate, suggest that the Waterfall methodology is inappropriate to use. The fact that the ‘Rapid Application Development’ methodology requires a high level of user contact throughout the project and the iterative nature also proves it to be unsuitable. In their favour, however, both methodologies are suited to projects of this size (small).
Both the ‘Modified Waterfall’ and the ‘Spiral Model’ are appropriate for use in this project; however, the major drawback of the ‘Spiral Model’ is its unsuitability for use in smaller sized projects. This leaves the ‘Modified Waterfall’ methodology as the most suitable and therefore the one that is followed. The strict nature of the development lifecycle should help progress and the allowance for a return to the preceding step should allow new information to be incorporated into the project easily. The project has been broken down into five stages and the methodology is shown in figure 1.

![Figure 1 – Project Methodology](image)

The first stage is been added as this is seen as an important part of the completion of this project. The final stage is removed, as the developed system will not be deployed but passed to the SIS Manager. Additionally the requirements definition section will use the ‘MoSCoW rules’, detailed in section 2.5, for stating requirements, as discussed in Avison and Fitzgerald, (2003). This is because the planned development is an extension of an existing system, so it is felt that an extensive requirements document is not needed. The idea of user evaluation of prototypes in the ‘Rapid Application Development’ methodology, Avison and Fitzgerald, (2003), is seen as useful and a modified use of this is made throughout the implementation phase. Each statistic is implemented individually and then demonstrated to a potential user, to try and limit the amount of alteration required as a result of the final user acceptance testing on the whole system.

### 2.4 Implementation Tools

There are two areas of the project that require investigation into the tools available. They are creation of the system front-end, required to interact with SIS and generate HTML and / or images, and incorporating the historical data into the system.
2.4.1 System Front-End

The required system has to take input from a web page, perform a query on a PostgreSQL database and then format any data that is returned graphically. Since the data is to be retrieved from the database and its formatting and display created dynamically it is preferable to use a server-side scripting language. This is the most logical solution, because the server only communicates with the database (higher security) and also, it will minimise communication between the client and server. The particular disadvantage with this is that it does place an increased load on the server because it is doing all the processing involved.

Many server-side scripting languages exist, examples being Microsoft ASP, PHP3, Perl and Java Servlets, each with its own advantages and disadvantages. However, Perl has been used for all other developments in SIS so for ease of integration, maintenance, and to not violate any security constraints in place, its usage is a specific requirement of this project. A description of the language and the parts of it that will prove especially useful follows.

Perl is a programming language that was created in the early 1980’s that has been "accepted as the de facto standard for writing common gateway interface (CGI) programs on the World Wide Web," Descartes and Bunce, (2000). Saltzman, (2002), gives a list of uses of Perl that are currently made. This includes uses such as CGI applications, file and text processing and database retrievals, all of which are essential in implementing a solution to the problem. Finally Perl is a platform independent language, a feature that is essential due to the variety of Internet browser software available.

The Perl Database Interface (DBI) “is a package that allows you to access relational databases from within a Perl script,” Saltzman, (2002). It is used by loading the correct driver for the database and supports every major database management system from PostgreSQL to SQL Server 2000. Access to a database is achieved by the execution of SQL queries defined within a script, Saltzman, (2002). Once the data is retrieved it can be manipulated in any way desired.

Perl has the ability to dynamically generate web graphics through the use of one of many graphics modules written for it. GD is one of these graphics modules highlighted in Wallace, (1999). Upon investigation it was found that this module is installed on School of Computing machines and is available for use. During this investigation another module, GDGraph, was discovered and is also available for use. Upon checking the documentation for this module, Verbruggen, (1999), the conclusion was reached that its usage would be much simpler. This
is because the structure of various graphs has already been coded, with the developer only having to specify the required data and settings.

The GDGraph module provides classes for eight different types of graph. The classes are for vertical and horizontal bar graphs, line, point and line with point graphs, area graphs, mixed graphs and pie charts. Of most interest to this project are the classes for a vertical bar graph and for a pie chart, although the type of graph used for each statistic is not decided until the design phase. Many different settings can be specified, some of which are common to all types while others are specifically for one type of graph. For example labels, the title and the required colours can be specified for each type of graph. The size of axes and spacing of elements on the x-axis are specific for graphs with axes. Pie charts are different in that settings are available to control the start angle and whether the chart appears raised.

The module requires data to be in a certain format for it to be able to draw a graph. That format for a bar graph is a two dimensional array. The values in the first dimension are a list of the labels for the graph. The values in the second dimension specify the size of the data for each label (e.g. the height of the bars for a bar graph). Subsequent dimensions can be specified for most graphs with either the elements being drawn for each label, or stacked for each label.

The GDGraph module supports the creation of images in the two major standards for image files on the Internet, the Graphic Interchange Format (GIF) and Portable Network Graphics (PNG). A comparison of the two file formats, discussed by Wheeler, (2000), is summarised in Table 2, below.

<table>
<thead>
<tr>
<th>File Format</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIF</td>
<td>Lossless compression.</td>
<td>Patented compression algorithm. 8-bit colour depth.</td>
</tr>
<tr>
<td>PNG</td>
<td>Open specification format with a non-proprietary compression algorithm. Lossless compression. High colour depths supported.</td>
<td>May not be supported by older versions of internet browsers.</td>
</tr>
</tbody>
</table>

Table 2 – Comparison Of Image Formats
The Portable Network Graphics (PNG) file format is used for dynamically generated images, because of its suitability to the task and also the non-proprietary nature of its compression algorithm.

The dynamic generation of web pages and images is not actually directly possible in a single script, Kew, (2000). The reason for this is because of the different header content code required by dynamic web pages and images in CGI scripts. To overcome this the script is divided into ‘subs’ that are individual sections of code that concentrate on performing a single task. A command variable is used to determine which sections of code are called and when that is.

2.4.2 Incorporating Historical Data

Historical data is currently held in Microsoft Excel spreadsheet files. Perl is a particularly good language for parsing text files; however, having a script access a text file every time it is executed would be vastly inefficient and insecure. The simplest solution is to create new tables within the SIS database, fill them with the historical data and to access that data through SQL queries.

Tables are created in PostgreSQL through the use of simple ‘CREATE TABLE’ syntax, Lockhart, (2000). Data held in files can be read into those tables with the ‘\copy’ command, Eckermann, (2001). This command allows a delimiter (such as a comma) to be specified that separates the data for each field in a row of the table in the file. Excel has a built in function that allows files to be saved in a comma-separated format.

2.5 Users And Their Requirements

In order to identify the user requirements of the intended development, the potential users must first be recognized. The system is to be used on the School Intranet; therefore the potential users are limited to its members who have a username and password allowing them access to it. This includes the department’s members of staff and students (including groups such as single honours and joint honours undergraduates, taught and research masters and postgraduates). Everybody will use the system in the same way so it is treated as having one user group.
As stated previously the ‘MoSCoW rules’ of requirements definition, as described in Avison and Fitzgerald, (2003), will be used from the Rapid Application Development methodology. This is where the requirements are sorted into four categories based on their importance. The categories are: ‘the Must haves,’ without which the project is pointless; ‘the Should haves,’ those requirements that do not directly influence a project’s success but will add the maximum benefit; ‘the Could haves,’ the requirements that can be met if time allows but can be left out; and ‘the Won’t haves,’ requirements that simply will not be met. This method of requirements definition is used because the development is an extension to an existing system and other definition methods would be too comprehensive.

The requirements, identified after careful analysis, are listed below.

### 2.5.1 Must Have Requirements

- The system must query the SIS database and return the relevant data formatted in a web page.
- The system must operate the same on the many different browsers available on the School of Computing and University of Leeds machines.
- The system must allow concurrent access between users.

### 2.5.2 Should Have Requirements

- The system should format the data graphically.

### 2.5.3 Could Have Requirements

- The system could have a clear and easy to use interface allowing the user to choose which statistic to view.

The ‘Won’t have’ requirements have been omitted, as successfully meeting the minimum requirements of the project will also meet the user requirements listed above.

### 2.6 Statistics To Display

The following list of statistics to gather and display via the developed system is compiled after initial talks with Dr. L. Proll (SIS Manager) and a meeting with Professor P. Jimack.
(Director of Research). In SIS there are four main areas of interest from which statistics can be gathered: publications, research contracts, research students and taught students. Within each area there are many potential statistics that may be gathered and displayed. Additionally, information about the number of full time equivalents can be generated from student data. Each area of interest, the statistics to be gathered and their data source are highlighted below.

2.6.1 Research Students
This area covers students performing research within the School with the aim of achieving a further qualification.

The statistics to be gathered for this area are:

- Number of PhD’s awarded each year.

The data for this statistic is held entirely within SIS.

2.6.2 Taught Students
This area covers students taught within the School.

The statistics to be gathered for this area are:

- Total number of graduates each year.
- Breakdown of classifications (I, Iii, III, II, Ordinary, Fail) for a single honours degree programme for a year.
- Comparison of the above breakdowns, per year and per degree programme.

Data relating to all statistics within this area is held within SIS from the 1996/1997 session. Historical data relating to the number of graduates and the classifications for single honours degree programmes is held in two spreadsheets.

2.6.3 Publications
Many types of publication come from the School each year, details of which are held within SIS. Those specifically of interest are ‘Conference Contributions: refereed’, ‘Edited Works: contribution’ and ‘Academic Journal Papers’. The authors of those publications should be
members of the School of Computing (that is either Staff, Students, or both in the case of joint-authorship).

The statistics to be gathered for this area are:

- Total for types of publication and author by year.

The data for this statistic is held entirely within SIS.

2.6.4 Research Contract

This area covers research undertaken in the School that is funded with a contract. Each contract is recorded with the source and a percentage. Primarily of interest are those contracts issued by research councils. The percentage records the amount of the total value of the contract that is awarded to the School.

The statistics to be gathered for this area are:

- Total value of new contracts in the School of Computing per year.
- Total value of new contracts from ‘Research Councils’ per year.
- Total value of ongoing contracts from ‘Research Councils’ per year.

The data for these statistics is held entirely within SIS.

2.6.5 Full Time Equivalents

This is the number of full time equivalents registered with the School in a session. It is broken down into a number of groups. The groups that comprise the full time equivalents for the School are single honours and joint honours students, service students (that is students for whom the School is not the parent department or joint honours students for whom studies in the School of Computing are not a principle component of their degree), masters students, postgraduate students and the number of staff involved in teaching and support. To calculate the full time equivalents the total credits studied by a group is counted and divided by the credits studied by a single student in a session. Additionally the staff-student ratio can be calculated from the total of individual group counts divided by the staff count.
The statistics to be gathered for this area are:

- Total number of Full Time Equivalents per year.
- Staff-student ratio per year.

The University releases official figures for the full time equivalents within the School each summer. Currently, this data is stored in a spreadsheet and not explicitly within SIS. Figures can be calculated from the 1996/1997 session onwards for most groups, however, there may be some slight discrepancy between the calculated and official figures (particularly in the service group where data for ACOM modules is not held within SIS). In light of this historical data, it is held in a new table, which should be updated as new figures become available each year. Figures for the current session, not yet issued by the University, can be calculated from SIS.
Chapter 3 – System Design

3.1 Introduction

This chapter will describe the design of the system produced for the third phase of the methodology being used. This incorporates both simple database design for the tables that contain the historical data and human-computer interaction issues with the web front-end. A discussion of each statistic, the required query and the associated graph, follows this.

3.2 Database Design

Historical data is held in Microsoft Excel spreadsheets, which can be read into PostgreSQL tables by saving them as a comma-delimited text file and using the ‘\copy’ command, discussed in section 2.4.2. In order for this to be achieved the tables that are to contain the data need to be designed before being implemented. There are three files containing historical data, each requiring an individual table. The tables are named according to the convention currently employed in SIS – that is with initials corresponding to the area the table is related to, e.g. ‘ts_’ for taught students, followed by the name of the spreadsheet file appended with ‘_history’. The columns in each table correspond to the majority of columns present in the spreadsheet; however where there are differences between the two the reasons for this are highlighted.

The new tables are unrelated to other tables in the database and they are completely static in the sense that they will not be updated (or at least only updated annually). Considering these facts leads to the conclusion that the tables need not have primary keys and can remain in first normal form, that is all attributes are atomic, Elmasri & Nevathe, (2000).

3.2.1 Graduate Table

The file ‘graduates.xls’ contains data relating to the number of graduates from each degree programme offered by the School. This incorporates both single and joint honours, which introduces the single issue with this data. An extra row is used within the spreadsheet to distinguish whether it is single or joint honours because the programme initials are the same. This extra row cannot be used within a database table so the simplest solution is to rename some of the columns so that they are all unique. All joint honours programme initials are
appended with a ‘W’ to indicate they are ‘With’ another programme. Also present is a column concerned with graduates from ‘Computer Arts’. Since this is also a joint honours programme this column will be renamed with the initials ‘JHA’.

Some programmes did not run in a session, e.g. Computing was not introduced until the 1998/1999 session so had no graduates until the 2000/2001 session. Converting to a comma-separated file will simply result in two commas to indicate there was no data in a field. When read into the database a zero value will be inserted into the relevant field, which is the desired result so no action needs to be taken. The design can be seen in Table 3, below.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session</td>
<td>Character(9)</td>
<td>The session of the data</td>
</tr>
<tr>
<td>CG</td>
<td>Integer</td>
<td>Single honours Cognitive Science</td>
</tr>
<tr>
<td>CS</td>
<td>Integer</td>
<td>Single honours Computer Science</td>
</tr>
<tr>
<td>CT</td>
<td>Integer</td>
<td>Single honours Computing</td>
</tr>
<tr>
<td>IS</td>
<td>Integer</td>
<td>Single honours Information Systems</td>
</tr>
<tr>
<td>OR</td>
<td>Integer</td>
<td>Single honours Operational Research</td>
</tr>
<tr>
<td>AIW</td>
<td>Integer</td>
<td>Joint honours Artificial Intelligence</td>
</tr>
<tr>
<td>CGW</td>
<td>Integer</td>
<td>Joint honours Cognitive Science</td>
</tr>
<tr>
<td>CSW</td>
<td>Integer</td>
<td>Joint honours Computer Science</td>
</tr>
<tr>
<td>CTW</td>
<td>Integer</td>
<td>Joint honours Computing</td>
</tr>
<tr>
<td>ISW</td>
<td>Integer</td>
<td>Joint honours Information Systems</td>
</tr>
<tr>
<td>JHA</td>
<td>Integer</td>
<td>Joint honours Computer Arts</td>
</tr>
<tr>
<td>ORW</td>
<td>Integer</td>
<td>Joint honours Operational Research</td>
</tr>
</tbody>
</table>

**Table 3** – ‘ts_graduate_history’ Table Definition

### 3.2.2 Degree Table

The file ‘degrees.xls’ contains data relating to the breakdown of the graduates from single honours degree programmes offered by the School into individual counts for each classification. There are a number of columns in the spreadsheet that are not needed in the database. The ‘Total’ and ‘Graduates’ columns are not needed as these figures are calculated within the file, a feature that can be repeated by any script reading the data. The ‘Pending’ and ‘DipHE’ columns are not needed either as they are not related to the act of graduating.
There are three single honour programmes for which there is historical data – Cognitive Science, Computer Science and Information Systems. To record this fact some data redundancy is introduced by repeating the session for each programme. A further column, ‘Programme’, is added to distinguish the programme a repeated session refers to. The design can be seen in Table 4, below.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session</td>
<td>Character(9)</td>
<td>The session of the data</td>
</tr>
<tr>
<td>Programme</td>
<td>Character(19)</td>
<td>The programme of the data</td>
</tr>
<tr>
<td>I</td>
<td>Integer</td>
<td>First degree classification count</td>
</tr>
<tr>
<td>II_i</td>
<td>Integer</td>
<td>Upper Second degree classification count</td>
</tr>
<tr>
<td>II_ii</td>
<td>Integer</td>
<td>Lower Second degree classification count</td>
</tr>
<tr>
<td>III</td>
<td>Integer</td>
<td>Third degree classification count</td>
</tr>
<tr>
<td>Pass</td>
<td>Integer</td>
<td>Pass degree classification count</td>
</tr>
<tr>
<td>Ordinary</td>
<td>Integer</td>
<td>Ordinary degree classification count</td>
</tr>
<tr>
<td>Fail</td>
<td>Integer</td>
<td>Fail degree classification count</td>
</tr>
</tbody>
</table>

Table 4 – ‘ts_degree_history’ Table Definition

### 3.2.3 Full Time Equivalents Table

The file ‘ftes.xls’ contains data relating to the number of full time equivalents in the School of Computing per session. Two columns, ‘Total’ and ‘Staff/Student Ratio’, found in the spreadsheet are not required in the table as they are calculated from the other figures, a feature that can be repeated by any script reading the data. Figures for the ‘Masters’ section are included in those for ‘PG Research’ before the 1999/2000 session. A statement saying this will simply be placed on the statistics page. This table will be updated once per year as the University releases official figures for the full time equivalents in the department. Figures for the current session will be estimated from data in the live database. The design can be seen in Table 5.
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session</td>
<td>Character(9)</td>
<td>The session of the data</td>
</tr>
<tr>
<td>Single_subject</td>
<td>Double Precision</td>
<td>FTE figures for single subject</td>
</tr>
<tr>
<td>Joint</td>
<td>Double Precision</td>
<td>FTE figures for joint honours</td>
</tr>
<tr>
<td>Service</td>
<td>Double Precision</td>
<td>FTE figures for electives</td>
</tr>
<tr>
<td>Masters</td>
<td>Double Precision</td>
<td>FTE figures for masters courses</td>
</tr>
<tr>
<td>PG_research</td>
<td>Double Precision</td>
<td>FTE figures for PhD courses</td>
</tr>
<tr>
<td>Staff</td>
<td>Double Precision</td>
<td>FTE figures for staff</td>
</tr>
</tbody>
</table>

Table 5 – ‘fte_history’ Table Definition

### 3.3 Interface Design

The interface to the developed system is entirely web-based. This means that there will be no help files, or manuals, available to the user so the interface has to be clear and intuitive to use. That said, the user-base of the development is limited to members of the School so a certain degree of computer / Internet literacy can be assumed. It is still very important to consider the research undertaken in the field of human-computer interaction and apply it in the design of the interface. It is important to do this because it is widely recognised that this area has been overlooked in the past and some systems have truly awful interfaces, rendering them effectively useless. Using the findings of this research and a time-sequence diagram a basic interface is designed.

#### 3.3.1 Human-Computer Interaction Issues

The most important fact to consider is consistency. This may be considered common sense but is surprisingly overlooked all too often. A system with different interfaces for different functions severely limits usability. According to Dix et al, (1998), consistency in the interfaces can improve usability considerably. This means that all the interfaces of the developed system should not only have a consistent style among themselves; but also that style should be consistent with what currently exists in SIS.

The recommended font size for text is between 9 and 12 points, Dix et al (1998). Many websites have text that is smaller than the recommended point size range, making them unnecessarily hard to use. Of course if the user finds text too small to read on a webpage they have the ability to increase its size via the browser software. This is still a slight inconvenience to the user, so to avoid this the recommendation will be followed.
A major issue with the Internet is bandwidth and download speed. System response speeds of fewer than ten seconds are needed to keep the users’ attention according to Smith, (1997). The dynamic generation of an image is a processor-intensive task so, in order to keep the system response time at a minimum, the size of the image will be limited. The variety of resolutions in use on different computer monitors also influences this. The minimum resolution available is six hundred and forty by four hundred pixels. The computers in use in the School have a minimum resolution of one thousand and twenty four by seven hundred and sixty eight pixels. Vertical scrolling is fine but to remove the need for horizontal scrolling the maximum width of an image is eight hundred pixels.

Effective use can be made of colour, especially in the differentiation of different categories in the columns of a graph. It is important to note that, “around 8% of males and 1% of females suffer from colour blindness... commonly being unable to discriminate between red and green,” Dix et al, (1998). To overcome this it is suggested that colour should not be the only indicator of any difference present. In light of this, each statistic will not only be returned in the form of a graph, but also presented in a textual form in a table underneath the graph.

### 3.3.2 Time-Sequence Diagram

A time-sequence diagram is used to visualize data and control passing an interface between two objects, Yeates et al, (1994). The diagram has boundaries between objects and an arrow crossing this represents the passing of data and control between the objects, as seen in Figure 2. The boundary also represents time passing in the system. This type of diagram is more suitable for use here, rather than a state-transition or data-flow diagram, because the system is relatively simple in operation.
From this diagram it is clear to see that the system requires two interfaces; one for the menu page and one for the statistics page. Also it can be inferred from the diagram that the Perl script developed has three functions: to format a menu of the available statistics, upon the selection take the appropriate input, request the relevant data, format that data and generate a graph from it.

### 3.3.3 Prototype Interfaces

Using the information discovered in research into the field of human-computer interaction and the required interfaces discovered through use of a time-sequence diagram prototype designs for the menu interface, Figure 3, and the statistic interface, Figure 4, are constructed. Consistency is maintained through the use of the standard SIS header and footer.
3.3.4 Script Execution

The execution sequence of the script is shown in Figure 1; however, the execution sequence of the dynamic table and image generation code is unclear. As discussed in section 2.4.1 the Perl script is unable to deal with dynamic web page and image generation in a single script.
Upon selection of a statistic from the menu the first thing to be generated by the script is the results page with the table. The image 'sub' is then called recursively by a HTML image tag within the results page.

### 3.4 Statistic Design

Each statistic is designed in this section. This includes a description of the query used to retrieve the required data, any error checking that is required as a result of user input, and a decision on the type of graph most suited to that statistic. As discussed in section 3.3.1 each graph will have a table of results below it.

In order to learn about SIS Mr. J. Ainsworth, the School of Computing Information Officer, provides an ‘anonymised’ copy of the database. He also provides a quick overview of the structure of the tables and the information contained within. More comprehensive information is gathered from several meetings with the SIS Manager, Dr. L. Proll.

#### 3.4.1 Research Students

- Number of PhD’s awarded each year

Two research student tables are required, joined on student ID so that the status can be taken as graduated. The type of course (PhD) also has to be specified in the conditions of the query.

A vertical bar graph is the most appropriate type of graph to use here for clarity in representing the data. This comes through the height of each bar representing the count. A line graph is inappropriate because it would mislead the user through the continuous line between years. A point graph could be used but a user could easily overlook small points.

#### 3.4.2 Taught Students

- Total number of graduates each year.

The student table contains records for master’s students as well as undergraduate students, which needs to be joined to another table so that only undergraduate degree programmes can be specified in the query. The query also needs to contain a statement to exclude
current students (those who have no degree result recorded). Once the historical data is read into a table it can be retrieved with a simple query.

A vertical bar graph seems the obvious choice as the query involves many years. The bar graph could be drawn with columns for each degree programme per year, however this would take up a large amount of space. A bar graph with stacked colour segments for each degree programme is more suitable, as it does not require so much of the available image width per year and the height of the stack represents the total. This feature is not as pronounced with the other types of graph available.

- Breakdown of classifications (I, III, IIi, III, Ordinary, Fail) for a single honours degree programme for a year.

Data for this statistic is retrieved in a similar way to that for the total graduates. The same joins are made but this time the programme and owner of that programme are used to restrict the results to those that are required. Once the historical data is read into a table it can be retrieved using a simple query.

Error checking is required on the user input. The degree programme is selectable from a drop-down menu so no error can occur here. The year is the only input that needs to be checked for validity.

A pie chart is the most appropriate type of graph to use here, as it allows direct comparison of the different classifications achieved in a year. A vertical bar graph would contain a single bar with different bands or five bars, one for each classification, and would not allow the easy comparison available in a pie chart. Again point and line graphs are not suitable for reasons discussed in section 3.4.1.

- Comparison of the above breakdowns, per year and per degree programme.

The query used in the previous statistic can be reused here, only executed twice with different years / degree programmes depending on the request.

In the case of comparing years the two inputs need to be checked for validity. In the case of comparing programmes each selection is made from a drop-down menu. Although the contents of each menu are, by default, correct the selection of different degree programmes
needs to be validated. Also the year input by the user needs to be checked for validity. In the
case of comparing two years for one programme the two years need to be validated.

The simplest way to allow comparison between two years is to place two pie charts,
produced in the same way as for the previous statistic, for the different years / programmes
on the same page. This is better than a vertical bar graph as it allows a better view of the
performance both each year and between the two years / programmes.

### 3.4.3 Publications

- Total for types of publication and author by year.

The authors of and types of publication are linked in another table that is itself linked to the
main publications table. In order to retrieve the required data all tables need to be joined
together and the conditions outlined in section 2.6.2 specified.

A vertical bar graph seems the obvious choice as the query involves many years. The bar
graph could be drawn with three columns per year – one for each type of publication, however, a bar graph with segments for each type of publication is preferable, as it does not
require so much of the available image width per year and the height of the stack represents
the total. The segments cannot be broken down further into authors because joint authors
cannot be distinguished from the database. Point and line graphs are completely unsuitable
for this statistic for reasons outlined in section 3.4.1 and also it is hard to read and compare
stacked values.

### 3.4.4 Research Contract

- Total value of new contracts in the School of Computing per year.

Using the source and project tables joined together on the source id any internal contracts
can be discarded and the total value calculated (taking the percentage into account).

- Total value of new contracts from ‘Research Councils’ per year.

A similar query to that in the previous statistic is required, simply altering the join condition to
account for the different source.
• Total value of ongoing contracts from ‘Research Councils’ per year.

Some difficulty is introduced by the fact that a contract from x years ago may still be running in the year currently being assessed. The problem is determining how far back in time to look. In order to overcome this, first a query is issued to find the maximum duration of a project, which is converted from months to years and used to calculate the start year in the query. The query is otherwise exactly the same as the one used in the previous statistic.

A vertical bar graph is the most appropriate type of graph for these statistics. A pie chart simply does not present the information correctly. The point and line types of graph are unsuitable for reasons outlined in section 3.4.1. Also with a continuous line the user may infer values between years, something that is not shown on the graph and is an incorrect view.

### 3.4.5 Full Time Equivalents

• Total number of full time equivalents per year.

Currently data for sessions since 1979 is held in a file. Upon reading the figures into a table a simple query retrieves the required data. Approximate figures for the current session can be generated from data held in SIS. For single honours, joint honours, service and masters this requires a count of the credits students are registered for. This is achieved through linking the taught student tables and using the degree type (e.g. ‘SS’ for single subject). The count is divided by one hundred and twenty (the number of credits studied per year) to give a figure for the full time equivalents. The figure for service, calculated in a similar fashion, is not accurate, for reasons discussed in section 2.6.1. The figure for postgraduate students is calculated from the attendance of the individual students (with a value of one assigned to full attendance and a half to anything less). The figure for staff is approximated from staff posting. This is not accurate either as the percentage of teaching work assigned to each staff member is not available.

A vertical bar graph, with segments for each of the areas that comprises the total full time equivalents, is the most suitable type of graph. A pie chart would be suitable if one session were being viewed, however the graph will have data for many sessions. The point and line types of graph are unsuitable for use because it is hard to read stacked values.
- Staff-student ratio per year.

This figure is calculated from the total full time equivalents divided by the staff count. Again the figure for the current session will be an approximation.

This graph will appear below the vertical bar graph for the full time equivalents, so to introduce some variance in the display, a horizontal bar graph is the most suitable type of graph to do this. Again point and line graphs are unsuitable for reasons outlined in section 3.4.1. A pie chart simply does not represent the information suitably.
Chapter 4 – Implementation

4.1 Introduction

This chapter will describe the implementation of the system based on the design produced in the previous phase of the project. This incorporates the creation and population of historical tables in SIS, the implementation of each statistic with a dynamically produced image and the subsequent user demonstration (all demonstrations are with Dr. L. Proll).

4.2 Database Implementation

Before the historical data stored in files can be imported it needs to undergo some cleansing so that it is in the correct format. The tables to contain the data also need to be created.

4.2.1 Data Cleansing

Each spreadsheet requires some cleansing before the data is ready to be read into the database. The first step in the cleansing process is the removal of columns of data that are surplus to requirements, outlined in section 3.2.1 – 3.2.3. After the removal of these columns the session column in each spreadsheet must be reformatted. Initially each session is in the form ‘98/99’ or ‘1998/99’. This is renamed in the form ‘1998/1999’ with the aim of making future queries simpler. Figures for sessions post ‘1996/1997’ are retrieved from the database so can be deleted from the files. Finally an extra column is added to the degree classification data to distinguish between different programmes where the session is repeated. After performing the cleansing on each spreadsheet it is saved as a comma-separated file.

4.2.2 Table Creation

Using the simple ‘CREATE TABLE’ syntax, discussed in section 2.4.2, the full time equivalents and the degree table are created following the designs from sections 3.2.2 and 3.2.3. There are, however, some minor problems with the design for the graduate table in that some column titles use reserved SQL syntax (namely ‘IS’ and ‘OR’). To overcome this ‘IS’ is renamed to ‘InfS’ and ‘OR’ to ‘OpR’. To maintain consistency in column naming, similar joint honours programmes are renamed in the same way, simply appended with a ‘W’. This can be seen in Figure 5.
### 4.2.3 Importing Data

Once the files are placed in the correct directory the `psql` client software is invoked. The `copy` command is accompanied by the target table name, the filename and the delimiter used in the file. On entering the command an error is received related to the differences in text file structure between Microsoft Windows and Linux. The data item at the end of a line in the text file is misread because Windows uses an extra carriage return character at the end of a line. This is overcome by using the simple `dos2unix` program, Lin & Wuebben, (1995), to alter the line formatting in each text file. Each file is simply read into the relevant table after this.

### 4.3 Statistic Implementation

At the beginning of the project Dr. L. Proll provided two sample scripts that gave some assistance in learning the Perl syntax. Further to this, Mr. J. Ainsworth provided copies of the Perl module ‘SOCweb.pm’ and the School of Computing Intranet cascading style sheet. The module contains functions to display the standard SIS header and footer. These can be used in the developed script with the addition of two simple lines of code. The use of the cascading style sheet allows the formatting of the interface of the developed system to be consistent with the rest of the SIS Intranet pages without the need for extra code.

#### 4.3.1 Script Security

In order to access the database the Perl DBI code in the script contains a username and a password. There is no security here as the script file is world readable. In order to maintain a...
level of security higher than this Mr. J. Ainsworth provided a compiled ‘wrapper’. This file is world readable and is renamed to be the same as the script. The script itself is placed in a private directory. The ‘wrapper’ is a simple program written in ‘C’ that calls the ‘setuid’ function to allow the browser access to the script.

To further increase the security of the script the strict and warnings pragmas and taint checking are specified at the beginning of the code, Saltzman, (2002). The strict pragma checks the script for use of unsafe code. It also disallows the use of any bare words, such as strings without quotation marks, and forces all variables to be declared as scoped variables. The warnings pragma writes warnings to the server error log when problems are detected. Taint checking is performed on any user input, which is regarded as tainted data, a tag that is passed on to data it interacts with. Tainted data is unable to affect anything external to the actual script.

The web server error log is checked throughout the implementation of the system, as it aids the development process. When execution of the script fails, the entries in the log usually point to the location of the problem in the script. Even when the script is executed properly the log is still checked, as there may be some warnings related to particular sections of code. The aim of doing this is to reduce the number of problems discovered in the final testing of the script.

A requirement of the system is that it works on the different Internet browser software available. In order to achieve this the system is tested throughout development on the two major versions of such software – Microsoft Internet Explorer and Netscape Navigator. Further testing on other platforms is covered in the next chapter.

### 4.3.2 Research Students

Statistics to implement:
- Number of PhD’s awarded each year.

Initially the query to retrieve the data is repeated in both the statistic and graph ‘sub’. A command variable is used to determine which of the available ‘subs’ in the script are executed. The construction of the statistic page is fairly simple. The standard header and footer are inserted with the addition of two lines of code. A connection to the database is specified and the data retrieved. The results of the research student query are returned to
In order to output the results in the form of a table the data is sorted into an array with two dimensions. This is achieved through the use of the ‘grep’ command, shown in Figure 6, to extract the values in the odd and even positions in the list. A simple ‘for’ loop is then used to construct a table by iterating through the results and printing them. This method of table construction is employed in all subsequent statistics ‘subs’.

$$\begin{array}{ll}
\text{my $ctr$ = '0';} \\
\text{my @number = grep ($ctr++ % 2) @data;} \\
\text{my $ctr-1 = '0';} \\
\text{my @year = grep (++$ctr1 % 2) @data;} \\
\text{my @results = ([@year], [@number]);}
\end{array}$$

Figure 6 – ‘Grep’ Usage

The same method of sorting the data into the required format is used in the graph ‘sub’. Upon creation of a blank canvas, the most important settings made are those for the height of the y-axis and the spacing of ticks. Improper specification of these settings would limit the usability of the graph in reading and comparing the data. The y-axis height is hard-coded to a value just above the maximum bar height. The labels on the x-axis overlap each other and are hard to read. To resolve this they are placed vertically instead of horizontally.

In order to call the graph from the statistic page code, the ‘sub’ calls the script recursively from an image tag, but specifies the graph command instead of the statistic command. This method of embedding dynamically generated images is employed in all statistic ‘subs’. The call passes any required parameters in the form of attribute/value pairs in the URL string.

The main comment to come from the demonstration is that the user may only be interested in data for a particular range of years. The start and end year of the range should be selectable. This idea is extended to all statistics that cover a number of years. The addition of this requirement is at an early stage in the implementation and does not require extensive revision of code; rather minor alterations to it and also to the design of the query for each statistic (to incorporate dates). It is considered to be a ‘Should have’ requirement.

To accommodate the selectable year range, the research student query is extended with greater than and less than statements, and also two variables for the start and end year are declared globally. The code to call the script for the image is extended to pass the required parameters between invocations of the script. Error checking is added, so that the two user input years are validated. The conditions checked for are: invalid start / end years, encompassing the start year being before the first year that data is held, and the end year being greater than the current year, and if the end year is before the start year. These are the standard set of conditions tested on all statistics that incorporate a range of years. In
order to improve code efficiency, the research student query is declared at the beginning of the script and called in the appropriate ‘subs’.

### 4.3.3 Statistics Menu Interface

At this point the menu interface is implemented, extending the design to incorporate user selectable start and end years, where required. Each statistic is described in a simple form with fields for the user input and a ‘Submit’ button. Another query is added to retrieve the current year, so that this is the default entry in the end year.

**Research Students**

PhD’s Awarded Per Year

Click ‘Submit’ to view data for the last 5 years. Alternatively, select the range of years you wish to view (data is held from 1960 onwards).

Start Year: [1997] End Year: [2002] Submit

*Figure 7 – Menu Interface*

Originally the default entry in the start year is hard-coded as the first year that data is held for the statistic. However, on user demonstration of this, it is recommended that this should be changed so that by default the range is just the previous five years. A sentence, explaining the year that data is held from, is placed above the fields. This can be seen in Figure 7.

### 4.3.4 Taught Students

Statistics to implement:

- Total number of graduates each year.
- Breakdown of classifications (I, IIi, IIIi, III, Ordinary, Fail) for a single honours degree programme for a year.
- Comparison of the above breakdowns, per year and per degree programme.

There are two ‘subs’ for each of the statistics in this area, one for the statistics page and one for the graph generation. All statistics incorporate data from the live database and also newly created historical tables.
The statistics page ‘sub’ for the total number of graduates, after error checking is performed on the start and end years input by the user (as described in section 4.3.2), begins by defining an array with entries for each year in the required range. To overcome the possibility (remote as it is) that there may be no graduates in a certain year, a results array is defined with dimensions for each year and values of zero in each entry. The graduate query is executed; raw data compiled in a list and the array of years is used for sorting the raw data into the correct position in the final results array. When the position for a data element is found the default entry of zero is overwritten by that data element. If historical data is required, the first thing to do is construct the start and end sessions, as these are stored instead of years. These are constructed from the start year and the last year that historical data is held. The historical graduate query is issued and the year array defined previously is used for sorting the raw data into the final results array. Finally the table is constructed from this array.

The statistics page ‘sub’ for the breakdown of the classifications for a single honours degree programme, after error checking is performed on the validity of the year input by the user, begins by defining a list of classes in the correct order. This is needed because raw data returned from the database is in alphabetical order, which puts a third (‘III’) before the second classifications (‘Iii’, ‘IIii’). Upon issuing the classification query, the results are checked and if no data was retrieved an error message is given. Returned data is sorted using the classification list, shown in Figure 8. To keep the data in alphabetical order, it is pushed onto the final results array in the order that the ‘if’ statement evaluates true. The final results array is then used to output the results in a table. If a historical year is entered, the newly created table is simply used in the classification query, otherwise the sorting and display process is the same.

```bash
# List of classifications used to sort the data into the correct order
@classifications = "I", "III", "Ii", "IIi", "III", "Ordinary";
$ctr = scalar(@classifications);
# Sort the raw data into the results array
$ctr1 = scalar(@record);
for ($i = 0; $i < $ctr; $i++) {
    for ($j = 0; $j < $ctr1; $j++) {
        if (@record[$j] eq @classifications[$i]) {
            push @ug_results, @record[$j];
            push @ug_results, @record[$j+1];
        }
    }
}

Figure 8 – Sorting Using A Pre-Defined List
```
The comparison of breakdowns of classifications uses exactly the same method as for the breakdown for a single year, except repeats the code to produce two tables. The command variable used to determine which ‘sub’ is executed is re-used to decided which code inside the multiple ‘sub’ is processed. The correct classification query is selected based on the two years input by the user.

In order to draw the total undergraduate graph with different coloured segments for each degree programme, the data retrieval method has to be slightly different. An array of years and a final results array are defined in the same way, however the final results array now contains dimensions for each programme. A list of the School codes for each programme (e.g. ‘CT’ for ‘Computing’) is defined and used within a loop to retrieve the figures for the required range of years, which is then sorted using the year array. If historical data is required, the newly created table is queried in the same way and raw data is sorted into the final results array. A legend for the graph is specified to allow the colour used for each programme to be distinguished.

The same method for data retrieval and sorting with a pre-defined list of classifications is used in both the single and multiple graph ‘subs’. The data for use in a pie chart has to be in exactly two dimensions; the first for the segment labels and the second for the size of each segment. In order to achieve this the ‘grep’ command is used in the same way as explained in section 4.3.2. The available settings for a pie chart are slightly different, allowing the start angle to be specified and also if the chart is to appear flat or slightly raised. An example of the pie chart produced can be seen in Figure 9.

The main comment to come from the demonstration is that, in the classifications statistics, the table should be limited to just figures for first, upper and lower second, third and ordinary classes. This excludes results such as ‘DipHE’ and ‘Fail’ as these are not classed as graduating. Also, for the breakdown of classifications for two years / programmes, the pie charts should be next to one another so that they are both displayed at the same time and removes the need for scrolling the page.
Three simple alterations are required in order to make the changes suggested in the demonstration. The lists in the two ‘subs’ used for sorting classifications into order are simply reduced to contain the five classes listed. A temporary variable is used to count the number of classifications discarded from the raw data. This is subtracted from the main counter used in the construction of the table so that the correct number of rows is output. In order to output the graphs side by side they are placed within a table with one row and two columns.

### 4.3.5 Publications

Statistics to implement:

- Total for types of publication and author by year.

```plaintext
# Put the results in the correct position
my $count3 = scalar(@data);
for ($j = 0; $j <$count3; $j = $j + 2) {
    for ($i = 0; $i <$count2; $i++) {
        if ($data[$j] eq $years[$i]) {
            $pub_results[1][$i] = $data[$j + 1];
        }
    }
}
```

**Figure 10 – Nested ‘For’ Loop Structure**

The statistics ‘sub’ generates a table with the year and the total for the three types of publication interested in. Before the main code is executed, error checking is performed on the start and end years input by the user (as described in section 4.3.2). The year and totals for the publications are found through the publication query and the raw data is read into a list array in the form ‘year-figure’. Another array is defined containing entries for each year in the required range. A nested ‘for’ loop, Figure 10, is used to compare the years from the raw data, and from the array of years, to sort the data into a final results array. This final results array is used in the construction of the table.

The graph ‘sub’ is similar in that it uses a raw data array and an array of the required years. In order to draw the graph, with segments for each type, the publication query is executed in a loop which is invoked the number of times as types of publication. The results array from which the graph is drawn contains dimensions for each type of publication, and the loop count is used to sort the raw data into the correct position. An extra setting, to stack the values in corresponding array positions, is specified in the graph settings. Finally a legend is specified to allow the user to distinguish the different segments. The GDGraph module automatically maintains similar colours between the graph segments and the associated key in the legend.
The main feedback to come from the demonstration is that the table should contain figures for each type of publication in place of a single total. Other than this the page and graph are fine.

The method used to read the values for each type of publication in the generation of the graph does exactly what is now required in the statistics ‘sub’. The results array is extended to include the same code from the graph ‘sub’. Finally the code to deal with the table construction is extended to account for the extra figures in the results array.

4.3.6 Research Contract

Statistics to implement:

- Total value of new contracts in the School of Computing per year.
- Total value of new contracts from ‘Research Councils’ per year.
- Total value of ongoing contracts from ‘Research Councils’ per year.

To maintain a high level of code efficiency the statistics page and graph ‘subs’ for each statistic are compiled into two ‘subs’. The command variable used for determining the function of the script is reused to determine the function of each sub.

The first two statistics are very similar, in that only the source of the research contract varies, which can be specified within the query. After the required error checking on the start and end year input by the user (as described in section 4.3.2), the relevant research contract query is executed. Retrieved raw data is stored in a list with the form ‘start-amount-percentage’. A ‘for’ loop is used in conjunction with an array with entries for each year in the required range to sort the data. When the start year of the contract matches a year in the range, rather than just copy values into the results array, the amount is multiplied by the percentage to calculate the value of the contract awarded to the School, shown in Figure 11. This figure is added to that already in the results array so that on completion of the loop the total is left for each of the required years.
For ongoing contracts, the calculation of how much falls within a year is more complex. The contracts are retrieved along with another attribute – the contract duration (recorded in months). To begin with, the value of each contract awarded to the School of Computing is calculated. Each contract is checked to see if it falls within the target year by adding the duration (converted to years) to the start year and testing whether it is greater than or equal to the target year. A final ‘for’ loop takes the start month of a contract and iterates through the duration to calculate whether a month is within the target year and, if so, adds the monthly value of the contract to the total. On completion of the loop, the total value of ongoing research contracts is held for the required year in the final results array. This process is repeated for the number of years in the range requested by the user.

The final results array, regardless of which statistic it has come from, is used to construct and print the table. Figures dealing with the money value for a year are only required to two decimal places. In order to achieve this a ‘sprintf’ statement is used.

Exactly the same methods are used to compile and sort the data in the graph ‘sub’ for all three statistics, as described above. The command variable is re-used here to select the correct data-retrieval method; to determine the settings for the graph title and to select the hard-coded value of the maximum for the y-axis.

The main feedback to come from the demonstration is that the table should contain a count of how many contracts make up the value stated for each year. Other than this the page and the graph are fine.
The inclusion of a count of the number of contracts in a year is a simple addition to make. For the first two statistics each time the ‘for’ loop is evaluated true, and the value of a contract is added to the total, a counter is incremented. Once all the contracts have been evaluated for a particular year, the counter is copied into the final results array and reset for next year. For the ongoing value statistic the code is executed for each year. In order to record a count the length of the raw data list of contracts that run in the target year is divided by the number of data entries per year. This value is copied into the final results array.

4.3.7 Full Time Equivalents

Statistics to implement:

- Total number of Full Time Equivalents per year.
- Staff-student ratio per year.

The data for this statistic for previous sessions is easy to retrieve as it is held centrally in one table. After the required error checking on the start and end year input by the user (as described in section 4.3.2), the full time equivalents query is executed. Retrieved raw data is stored in a list. An empty final results array is initialised, with a dimension for each category within the umbrella term full time equivalents (single/joint honours, masters etc). An array of years in the range is also defined and used in sorting the raw data. Calculations for the total and staff-student ratio for each year are also performed during the data sorting.

If the range of years selected includes the current session the final results array contains an extra dimension, which is filled with further queries issued to the live database. For taught students a query is issued in a loop, with different degree types, each time it is invoked. A check is added if a count of zero is returned, because this would move the results up a position in the raw data list, which is an undesired activity. Different queries need to be issued to find the figures for postgraduate research and staff, as detailed in section 3.4.5, along with the count conditions placed on each category. The required calculations for the total and staff-student ratio are performed and these figures are read into the final results array, which is then used in final table construction.

Two separate graphs are drawn – one showing the full time equivalents and one showing the staff-student ratio. The graph ‘sub’ is called twice from embedded image tags on the statistics page. Different command variables are used in order to instruct the graph ‘sub’ on which code to execute.
In order to construct the final results array for use in the full time equivalent graph the same method of data retrieval described above is used. The figures for the total and staff-student ratio are not required, so not calculated. Colour is used to distinguish the separate segments that make up each bar, so a legend is specified, the appropriate settings made and the graph drawn, shown in Figure 12. For the staff-student graph again the same method of data retrieval is used, however only the calculated figure of the total divided by the staff is stored in the final results array. To draw the horizontal bar graph instead of a vertical bar graph the specification of the blank canvas is altered from ‘bars’ to ‘hbars’.

The main feedback to come from the demonstration is that the table should display data in descending session order. This should be applied to all statistics rather than just this one individually to maintain consistency in the display of each statistic.

Making the revisions to the code based on the feedback of the demonstration requires the alteration of the ‘for’ loop used in the construction of the tables. The counter used to iterate through the final results array, printing each value, is initially set to zero and increments to the size of that array. In order to facilitate this change, the counter is altered to equal the size of the array and decrement to zero.
Chapter 5 – Testing

5.1 Introduction

After the implementation stage, the next step in the methodology being used in the project is testing. The motivation for this step is, firstly, to ensure the system operates correctly and does not crash, or contain bugs, and, secondly, to ensure that the system is acceptable to its users and will be used in the future. This is also known as validation and verification, Sommerville, (2001). This chapter details the two types of testing undertaken.

5.2 Validation

There are three areas of validation that need to be completed on the required system. They are: query testing to ensure they retrieve the correct data; black-box testing, or defect testing, which aims to detect any program faults or deficiencies; and Internet browser testing on the system to ensure its interoperability.

5.2.1 Query Testing

Each query in the script is tested to ensure it retrieves the correct data. For some statistics figures are held in the files, so the comparison between them and the retrieved figures is simple. For others this is not as straightforward and requires inspection of the actual tables and manual recording of the required data. This is the case for statistics in the areas of research students and research contracts. Comparison is made between data recorded manually in a file and the figures displayed on the web page. They are identical, proving the queries to be correct.

In the area of publications the School actually maintains a separate ‘Publications’ database alongside SIS. Comparison between the figures from that database and those displayed on the web page reveal that they are not identical, but only have an average difference of four in the total. According to Dr. L. Proll this is within acceptable bounds.

Figures for taught students for sessions retrieved from the database are also held within the historical files. When comparing the figures from the files to those displayed on the web page some discrepancy is discovered which indicates the queries are incorrect. Manually
checking the relevant tables in the database contradicts this view however. The query for the total number of graduates issued on the live database is found to be correct. Further investigation carried out by Dr. L. Proll uncovers the reason for the discrepancy between the live database and the copy being used. The copy is ‘anonymised’ to deal with data protection issues and part of the ‘anonymisation’ criteria is that student records are removed where only a few are registered for a degree programme (making them easier to identify).

Figures for the full time equivalents are stored in the historical file and are used in comparison with those displayed on the web page. The only difference occurs in the figures for the current session. Although estimates, due to reasons identified in section 3.4.5, the figures are not wildly inaccurate. The ‘anonymisation’ issue also affects this query for the single and joint honours full time equivalents in the current session.

In order to overcome the problem introduced by the ‘anonymisation’ of the data, the affected queries are sent to Dr. L. Proll to enable them to be tested on the live database. Two minor problems are discovered in the process of testing; the total undergraduate query picks up extraneous students, and the types are incorrectly specified in the full time equivalents query. To correct these problems, a minor revision is made in the conditions of each query.

5.2.2 Black-box Testing

Throughout the implementation of the system it is continually tested to ensure that this area of validation is relatively straightforward. In order to ensure the completeness of this type of testing a systematic approach is used focusing on the input, outputs and the function of each unit of the software, Peters & Pedrycz, (2000). Each area is treated as a unique unit.

The inputs to each unit are the years and / or programmes entered by the user on the statistics menu page, incorporating boundary condition testing. Boundary condition testing is also extended to cover the overlap in use of live and historical tables in some statistics. The output and functionality can be grouped, demonstrated by the statistics page being formatted correctly and the graph reflecting the data (i.e. there are no obvious differences between the data in the table and the graph). Data is also added to each table to test whether the additions are reflected accurately in the table and graph.

The full test plan, constructed using the gathered information, with expected and actual results, is in Appendix C. In summary the whole system generally performed as expected
with one minor error and one bug being identified. The error is concerned with the error checking and recovery in the total graduate statistic due to an 'if' statement not being correctly evaluated, which is easily corrected.

The bug is concerned with the overlap between historical and live tables in the comparison of the breakdown of classifications for degree programmes, and leads to a very strange pie chart being drawn. It originates from a term not being reset, leading to the incorrect sorting of the raw data; the addition of a simple line of code rectifies this. Two problems are identified with the drawing of the graphs; the closing vertical line on all bar graphs is intermittently drawn and on one occasion a slice remains uncoloured on a pie chart. Due to the intermittent nature of these problems the conclusion that they are a result of an error in the GDGraph module is reached. A solution to these minor problems is not found.

5.2.3 Browser Testing

In order to validate the interoperability of the system it is tested on the different Internet browsers available within the School. Older versions of the two leaders in the browser software market are in use on the University machines so this requires some further testing. The browsers used to test the system are: Microsoft Internet Explorer 5.0 and 5.5, Netscape Navigator 4.7 and 7.0, Opera 5.12, Mozilla 1.0 and Konqueror 3.0.0-12.

In order to test the system it is invoked in each browser and some statistics selected with random inputs. The display of each page is validated and the server error log is also checked for any additions that indicate problems. The system operates correctly, with only slight differences in the default display sizes for the font used, on all of the available browsers – a widely acknowledged difference.

5.3 Verification

In order to complete verification of the developed system a final user test is conducted for acceptance. The interactive role of the users throughout the implementation of the system, detailed in the previous chapter, has created a partial level of acceptance prior to this final testing.

The test consists of asking the user to experiment with each of the statistics in each area in turn. The role of the tester is to observe the user and to seek and record their opinion and
comments at each stage. The user in this case is Dr. L. Proll and the full results of the test can be seen in Table 6, below. A summary of the results and any actions taken follows this table.

<table>
<thead>
<tr>
<th>Statistic Area</th>
<th>User Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistic menu page and interface in general.</td>
<td>Felt that the use is intuitive but would benefit from some sort of menu listing the areas covered at the top of the page.</td>
</tr>
<tr>
<td></td>
<td>Each statistics page should have a link to the front-page so as not to rely on the ‘Back’ button.</td>
</tr>
<tr>
<td></td>
<td>The first graph on each page should fit entirely on the screen.</td>
</tr>
<tr>
<td>Full Time Equivalents.</td>
<td>The text in this, and all, tables should be right aligned to make reading it easier.</td>
</tr>
<tr>
<td>Publications.</td>
<td>‘Academic journals’ should appear first in the graph.</td>
</tr>
<tr>
<td></td>
<td>The table should be extended to contain counts for each type of publication.</td>
</tr>
<tr>
<td>Research Contract.</td>
<td>Taking suggested alterations into account the three pages in this area are fine.</td>
</tr>
<tr>
<td>Research Student.</td>
<td>Taking suggested alterations into account the page in this area are fine.</td>
</tr>
<tr>
<td>Taught Student.</td>
<td>The total graduates table should be extended to contain separate counts for single / joint honours.</td>
</tr>
<tr>
<td></td>
<td>Each classification should have a set colour to enable easier comparison to be made.</td>
</tr>
</tbody>
</table>

**Table 6 – User Acceptance Test Results**

In summary the user was satisfied with the system. Through observing and listening to the users comments, the system seems intuitive and straightforward to use. All the comments made are responded to by altering the script accordingly, barring the menu of areas on the statistics menu page. It is felt that a menu is unnecessary at this stage as the system is still relatively basic. Upon making the alterations to the script, the black-box test plan is run through once again to ensure there are no undesired effects of the changes.
Chapter 6 – Evaluation

6.1 Introduction
After the testing stage, the final step in the methodology being used in the project is evaluation. The aim of this phase is to assess whether the developed system has met its requirements and can be considered a success. This is achieved through the application of the criteria outlined in section 1.4. Further to this, the usability of the system is evaluated, with the aid of additional user testing, as this is considered to be an important feature of the system. Any additional comments that are received as a result of performing this additional testing are stated here. Finally, possible future enhancements to the developed system are discussed.

6.2 Requirements Evaluation
Using the criteria outlined in section 1.4 the developed system is evaluated as follows:

*Does the system work without producing errors and on the different web browser software available within the School Of Computing and the University Of Leeds?*

The answer to this criterion is yes. The system is thoroughly tested for errors on a single browser during development and also on all the available browsers. A high level of interoperability is achieved through the successful testing of the system on all of the available web browsers within the School and University.

*Is the choice of language appropriate?*

The answer to this criterion is a definite yes. Perl is perfectly suited to the task and the platform independence is essential in providing a high level of interoperability without requiring additional development. The high processing requirements of graph generation would be present whatever language is used for development and is only affected by the server specification. The script being written in the same language as the rest of SIS allows it to be easily incorporated with the existing system, with only a few alterations required. The dynamic nature of data retrieval requires little or no maintenance.
Does the system take account of any security issues with the development language?
Again the answer to this criterion is yes. Security within the script is maintained with the use of the warnings pragma, tainted data and the strict pragma. Through the process of error checking, any problems identified by the three security measures are observed and corrected. The additional use of a wrapper program improves the external script security. This is the same system that is presently in use within SIS, so no radical alterations to the script are required as a result of security measures.

Does the system return the correct figures for each statistic and is historical data fully integrated and returned when requested? Incorrect figures indicate errors within the program.
The answer to this criterion is also yes. The system output is thoroughly checked for the accuracy of the returned figures. Also, the dynamic ability of the system to cope with new data is proven, with additions made to the relevant tables and being reflected on the web page. This ability to cope with new data should limit any future maintenance costs of the system. Historical data is incorporated in the relevant statistic areas and is retrieved with no discernable difference to the user.

Are any graphs produced by the system clear and easy to read?
The answer to this criterion is explored in detail in the next section. From comments received during the implementation phase the initial answer is yes. The accuracy of the graphs is not covered by this criterion, but also must be evaluated. Each graph, when compared to the table of figures on the same page, is an accurate reflection of the data.

Do users find it easy and intuitive to use the system? No user manual is available for web-based systems, so this is essential.
Again the answer to this criterion is explored in detail in the next section.

Is the system code written in a way that makes it easily readable and extendable in the future?
The answer to this final criterion is also yes. The design and implementation of the script is undertaken with expandability in mind. To add further statistics into the system two 'subs' for each one simply need to be added to the script, one for the actual page with the table and one for the production of the graph. The existing code is fully commented to ensure that it is not only readable but, more importantly, understandable.
6.3 Usability Evaluation

In order to fully evaluate the usability of the developed system additional user testing is conducted with Dr. J Davy, Director of Learning and Teaching, and Professor P. Jimack, Director of Research. Ultimately the statistics compiled by the system fall into teaching or research activity within the School of Computing, which are the roles performed by the two members of staff. Each member’s opinion is sought for the statistics covered by their role.

On its own the term usability can have a variety of meanings specific to different types of system. In order to assess the usability of a system, criteria have to be identified that allow a quantifiable measure to be taken, Lindgaard, (1994). This allows not only the presence of a criterion to be detected, but also the level of success attained for that criterion. The criteria used in the evaluation of the system are:

- **Learnability** – this refers to the ability of the user to complete tasks with no help given by the tester.
- **Effectiveness** – this refers to the performance of the user, quantified by speed or accuracy, in completing a given task.

Further to these two criteria, the general opinion of each user is also sought as this may highlight suggestions for further improvement.

6.3.1 Usability Testing

With no previous experience of the system the learnability and effectiveness of the system is heavily tested. This is very useful because with a web-based system no introduction and very little help can be given.

The learnability of the system is demonstrated by the user’s ability to successfully make use of it. This also assesses the intuitiveness of the system, as there will not be any support available in the event of difficulty. Using this fact a statement that assesses the learnability is, ‘can the system be used without needing explanation?’

In both tests the answer to this question is a definite yes. The main comment received from Dr. J. Davy is that the front screen needs a menu, regardless of the number of statistics present, to avoid the need for scrolling to get a view of the available statistics. The main
comment received from Professor P. Jimack is that the start year of data stored for each statistic needs to be clearer.

The effectiveness of the system can be seen as the user being able to complete a task in a given amount of time. The measure of speed is chosen over accuracy because there is a low probability of user error in the system. Using this fact a statement that assesses the effectiveness is, ‘can the system be used successfully within ten minutes?’

Again in both tests the answer to this question is a definite yes. The main comment received from Professor P. Jimack is that the input boxes on the front screen gave the impression of a drop-down menu.

In general, Dr. J. Davy expresses the opinion that the graphs are a little small. Also for the total number of graduates the table should contain figures for each segment shown in the graph. Finally, for aesthetic purposes, the figures in the table on the web page should be more spaced. Professor P. Jimack expresses the opinion that the graphs are complementary to the tables of data. The research student statistic should be more clearly labelled so that the user knows that it is the number of PhD’s awarded per year. Finally, when returning to the front page from a statistics page, the system should return to the point of departure on the front page rather than the top.

The last question posed at the end of the test is, ‘would you use this system if it were incorporated within the SIS Intranet pages?’ Both members of staff respond with a yes with Professor P. Jimack stating that the system would prove useful for publicity reasons and also in his meetings with the Faculty Research Dean.

Although some people may consider the graphs to be unnecessary alongside the table of figures, other people appear to find it easier to gain an overall view of any trends in the data.

### 6.4 Project Evaluation

In completing this project a methodology and particular development tools are used. Each of these is evaluated in this section.

The ‘Modified Waterfall’ methodology is used in this project. The choice of this methodology is appropriate and it is strictly followed. The change in requirements that occurs at the
beginning of the implementation phase is managed well, something that would not be possible with the ‘Waterfall Methodology’. One criticism comes from the fact that each stage must be fully completed before proceeding to the next, limiting the concurrency that can occur between stages, however, overall this is the best methodology to use.

Perl is the development tool used for the implementation of the system. Despite not encountering the language before, a working system is produced, within the time limit, which meets its requirements. Other tools do not integrate the security and image generation available in Perl, so this is the best tool to use.

6.5 Future Enhancements

There is a very wide scope for future enhancements that could be incorporated into the developed system, ranging from minor alterations, which could not be made due to time restrictions, to extension of the system. Enhancements that have been identified are:

- Incorporate some sort of menu on the statistics front page. This could either take the form of a menu at the top of the page, with internal links, or a preceding page, with links for each area and the selectable statistics on another page.
- Use drop-down menus for user selectable year ranges on the front page. This would actually be an improved method for collecting input, as items in the menu are, by default, correct and the need for input validation would be removed. Also, this would remove the issue of making the start year for data clearer.
- Dynamically set the graph parameters to improve their readability. The importance of this is highlighted in a research contracts graph, where an extreme value for one year means the scale of the y-axis is high. This inhibits the size of the other bars, thus restricting any meaningful comparison. Instead the maximum value of the y-axis could be dynamically set to just above the maximum data value.
- Include more statistics in the system. With a working system developed it may be easier to identify further statistics (‘I will know it when I see it’ requirements).
- Improve the code efficiency in the script. As the project has progressed and experience with the development tools has increased, several areas have been identified where the efficiency of the code could be improved.

This list of suggestions is not definitive and there are many more additions that could be made to the system.
6.6 Conclusion

In conclusion the system can be viewed as a success, in that it meets the original project objectives and the requirements derived from these. Through undertaking the testing phase, the system is proved to be correct and robust. The evaluation allows the system to be viewed as useful and, with the many possible future enhancements, it is very extendable.
References

Ainsworth, Jonathan, (2002), *Introduction To SIS*, URL:http://www.comp.leeds.ac.uk/internal/sis/ [27.11.02]


Verbruggen, Martien, (1999), *Graph Module Documentation*, URL: http://theoryx5.uwinnipeg.ca/CPAN/data/GDGraph/Graph.html [01.04.2003]


Appendix A – Personal Reflection

It is fair to say that I approached my final year with a great deal of trepidation, specifically towards my ability to complete such a project. Although I have found the process a challenge, I have enjoyed the project and am satisfied with the way it has turned out. I have learned many valuable lessons and gained plenty of experience as a result of completing it.

Undeniably, this has not only been the largest software engineering task I have undertaken, but also the largest single piece of work I have ever attempted. Looking back, I feel that the time spent in constructing a realistic schedule was invaluable in distributing the workload evenly. The ability to construct my own work plan, suited to my own pace, was refreshingly different to the rather rigid nature of the traditional coursework schedule.

The research conducted during the analysis phase did not fully uncover all the facts that were required during implementation, suggesting that it was somewhat lacking in depth. This may be explained by my setting unsuitable goals that were too easily within reach. Another explanation for this may be my inexperience conducting interviews, something that has improved during the project, but still requires attention. I am, however, pleased with the selection of the correct methodology, as it did allow the inclusion of altered or new ideas.

With regard to the actual programming involved in the project, my confidence in my ability to adapt to a new language and write code from an early stage has grown immeasurably. I feel that the need to actually plan what to code, rather than charge ahead, has been reinforced by the experience of a development on this scale.

Given the opportunity to undertake this, or a similar project, in the future the experience gained would definitely assist me in avoiding a repetition of the problems described above. I would offer the following advice to any future students undertaking such a project:

- Choose a project that you are interested in and will enjoy doing – it does last for an entire session after all. Finding the motivation to complete a project that you are not enjoying cannot be understated.
- Construct a meaningful and realistic schedule. Organisation is the key to success and scheduling the work can save a lot of time in coping with unforeseen problems.
- Conduct the analysis of the problem as thoroughly as possible. Fully understanding the problem should remove the extra work required if alteration is needed.
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
<th>Duration</th>
<th>2002</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1/1</td>
</tr>
<tr>
<td>1</td>
<td>Project Planning And Analysis</td>
<td>14/10/2002</td>
<td>03/11/2002</td>
<td>3w</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Requirements Analysis And Definition</td>
<td>04/11/2002</td>
<td>24/11/2002</td>
<td>3w</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Milestone 1: Requirements Complete</td>
<td>24/11/2002</td>
<td>24/11/2002</td>
<td>0w</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>System Design</td>
<td>25/11/2002</td>
<td>23/12/2002</td>
<td>4w 1d</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Mid-project Report</td>
<td>29/11/2002</td>
<td>12/12/2002</td>
<td>2w</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Milestone 2: Delivery Of Mid-project Report</td>
<td>12/12/2002</td>
<td>12/12/2002</td>
<td>0w</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Christmas Break</td>
<td>23/12/2002</td>
<td>01/01/2003</td>
<td>1w 3d</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Revision And Exam Period</td>
<td>02/01/2003</td>
<td>20/04/2003</td>
<td>3w 4d</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Implementation</td>
<td>27/01/2003</td>
<td>23/03/2003</td>
<td>8w</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Compile Draft Chapter And Table Of Contents</td>
<td>08/03/2003</td>
<td>14/03/2003</td>
<td>1w</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Milestone 3: Delivery Of Draft Chapter And Contents</td>
<td>14/03/2003</td>
<td>14/03/2003</td>
<td>0w</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Testing</td>
<td>10/03/2003</td>
<td>30/03/2003</td>
<td>3w</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Milestone 4: Implementation Complete</td>
<td>23/03/2003</td>
<td>23/03/2003</td>
<td>0w</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Evaluation</td>
<td>24/03/2003</td>
<td>06/04/2003</td>
<td>2w</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Project Write-up</td>
<td>24/03/2003</td>
<td>02/05/2003</td>
<td>3w 5d</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Milestone 5: Delivery Of Project Report</td>
<td>02/05/2003</td>
<td>02/05/2003</td>
<td>0w</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C – Test Results

C.1 Research Students

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct values for start and end years input, table and graph display the same results.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Start year input is greater than the end year input.</td>
<td>Error message displayed on menu screen. Message highlights the cause.</td>
</tr>
<tr>
<td>Start year input is less than the first year that data is held for.</td>
<td>Error message displayed on menu screen. Message highlights the cause.</td>
</tr>
<tr>
<td>End year input is greater than the current year.</td>
<td>Error message displayed on menu screen. Message highlights the cause.</td>
</tr>
<tr>
<td>Additions to the database are reflected accurately in the graph and table.</td>
<td>Graph and table are displayed and are accurate.</td>
</tr>
</tbody>
</table>

C.2 Taught Students

- Total number of graduates each year.

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct values for start and end years input, table and graph display the same results.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Start year input is greater than the end year input.</td>
<td>Error message displayed on menu screen. Message highlights the cause.</td>
</tr>
<tr>
<td>Start year input is less than the first year that data is held for.</td>
<td>Error message displayed on menu screen. Message highlights the cause.</td>
</tr>
<tr>
<td>End year input is greater than the current year.</td>
<td>Error message displayed on menu screen. Message highlights the cause.</td>
</tr>
<tr>
<td>Range of years input is historical only.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Range of years input is live only.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Additions to the database are reflected accurately in the graph and table.</td>
<td>Graph and table are displayed and are accurate.</td>
</tr>
</tbody>
</table>
• Breakdown of classifications (I, Ii, Iii, III, Ordinary, Fail) for a single honours degree programme for a year.

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct value for the year input, table and graph display the same results.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Year input is less than the first year that data is held for.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>Year input is greater than the current year.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>No data is held for the year and course selected.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>Additions to the database are reflected accurately in the graph and table.</td>
<td>Graph and table are displayed and are accurate.</td>
</tr>
</tbody>
</table>

• Comparison of the above breakdowns, per year.

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct values for years 1 and 2 input, table and graph display the same results.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Year 1 input is less than the first year that data is held for.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>Year 2 input is greater than the current year.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>Year 1 is equal to year 2.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>Two years input use historical and live source tables.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Two years input use live tables only.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Two years input use historical tables only.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Additions to the database are reflected accurately in the graph and table.</td>
<td>Graph and table are displayed and are accurate.</td>
</tr>
</tbody>
</table>
• Comparison of the above breakdowns, per degree programme.

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct value for the year input, table and graph display the same</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>results.</td>
<td></td>
</tr>
<tr>
<td>Year input is less than the first year that data is held for.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>Year input is greater than the current year.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>No data is held for the year and first programme selected.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>No data is held for the year and second programme selected.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>First programme is equal to the second programme.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>Additions to the database are reflected accurately in the graph and</td>
<td>Graph and table are displayed and are accurate.</td>
</tr>
<tr>
<td>table.</td>
<td></td>
</tr>
</tbody>
</table>

C.3 Publications

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct values for start and end years input, table and graph display</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>the same results.</td>
<td></td>
</tr>
<tr>
<td>Start year input is greater than the end year input.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>Start year input is less than the first year that data is held for.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>End year input is greater than the current year.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td></td>
<td>Message highlights the cause.</td>
</tr>
<tr>
<td>Additions to the database are reflected accurately in the graph and</td>
<td>Graph and table are displayed and are accurate.</td>
</tr>
<tr>
<td>table.</td>
<td></td>
</tr>
</tbody>
</table>
### C.4 Research Contracts

This test plan is used for all three of the statistics within this area.

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct values for start and end years input, table and graph display the same results.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Start year input is greater than the end year input.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td>Start year input is less than the first year that data is held for.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td>End year input is greater than the current year.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td>Additions to the database are reflected accurately in the graph and table.</td>
<td>Graph and table are displayed and are accurate.</td>
</tr>
</tbody>
</table>

### C.5 Full Time Equivalents

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct values for start and end years input, table and graph display the same results.</td>
<td>Graph and table correctly displayed.</td>
</tr>
<tr>
<td>Start year input is greater than the end year input.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td>Start year input is less than the first year that data is held for.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td>End year input is greater than the current year.</td>
<td>Error message displayed on menu screen.</td>
</tr>
<tr>
<td>Additions to the database are reflected accurately in the graph and table.</td>
<td>Graph and table are displayed and are accurate.</td>
</tr>
</tbody>
</table>