On-line support for MSc Project Administration

Ian Sugden
BSc Information Systems with Industry
2001 - 2002

Summary

This report documents the undertaking and progression of a project to create an Online Management and Monitoring System for the MSc Project process.

The project follows the Rational Unified Process though Business Analysis, Requirements Analysis, Design, Implementation, Testing and Evaluation, delivering a prototype that provides a centralised record of the MSc Students' progress through the project cycle, monitoring the flow of documents, deadlines, and the project and supervisor allocations.

The conclusion of the project is the prototype, which implements the monitoring functionality and gives an insight into how the submission of deliverables by the students can be handled by the system. The prototype is the result of the first iteration of the Design and Implementation process and, after another iteration, it is intended to be installed and used by the MSc Project Coordinators, Project Supervisors and Assessors and the MSc Students.

Table of Contents

Chapter Page

Introduction

Project Aim

The aim of the project is to provide an online support system for the management and monitoring of the MSc Project process, within the School of Computing.

The MSc Project process is supported by 3 members of staff (2 Project Coordinators and 1 Administrator) and currently there is no formal information structure. The information required is stored on paper and in spreadsheets, with each member of staff maintaining their own copies. This is very inefficient and open to discrepancies in the data.

This project proposes to find a solution to the problem by providing a central storage repository, with distributed access, so that any of the members of the team can view or update the information held from their own office. The system will also provide integrity checks, to make sure that the data stored makes sense, and management reports, summarising the information and presenting it in the most suitable way.

Personal Learning Objectives

The Personal Learning Objectives are what I hope to learn by completing this project. I hope to gain experience in the development of a system to fulfil the needs of real Users. This experience will include the implementation of a development methodology, defining User requirements, doing analysis of the requirements, class design, database design, implementation of the prototype, and technical and User testing. I also hope to gain knowledge of web-client and database development, specifically on Linux, using Apache web-server and a Postgress database.

Project Objectives

The Project Objectives are a list of goals that are required to be met for the project to achieve its main aim. By defining them at the start of the project, it is hoped that they will help guide the project towards completion, and also give some criteria to evaluate the success of the project at the end.

The objectives are:

- To understand the requirements of the MSc project process, for both Users and systems.

- To produce the design which meets these requirements and will fit into the School's Information Systems infrastructure.
- To produce a prototype of the designed system, to be reviewed by the Users.
- To evaluate the success of the system in terms of the usefulness of the prototype at meeting the Users requirements.

Minimum Requirements

This is a set of minimum requirements that the project must meet to achieve its objectives. As well as including the required work, they also state the functionality that the system will have to offer. By listing them at the beginning of the project, they provide a checklist for the final evaluation.

The minimum requirements of the project are:

- To produce an outline of the management and monitoring activities for the MSc Project process and summarise the requirements.
- To create a prototype system which will run within the School's IT infrastructure (web-client→ Linux, Apache, Postgress).

The prototype will provide the following functionality:

- Ability to record the "progress" of a student project by logging the flow of forms and reports from/to supervisor, assessor, project coordinator, and external supervisors (if appropriate), throughout the project cycle.
- Ability to generate a report from the information stored in the system.
- Ability to report the "progress" of individual students.

Approach

This is the approach that will be adopted in the completion of this project.

The planned approach is to follow a development methodology to produce a prototype of the solution, and then refine this prototype following the feedback from the Users. Development methodologies provide a framework for system developers to follow, to help ensure that the delivered system meets the User's requirements, is completed on time and within budget. Because a development methodology defines a set of documents that should be produced during the development process, it provides an audit trail for the project. By doing this, the project management can highlight any potential delays, new people recruited to the

development team can follow the work that has already been done, and it makes sure that no steps are accidentally missed out.

Therefore, one of the first decisions that has to be made, before the project can truly begin, is the choice of which development methodology to employ.

There are two main factors that will influence the choice of the development methodology, the first one being that of choosing a methodology that is appropriate to the project, and the second is to gain experience of using a methodology that is widely used in industry. According to topics covered in the IN33 module – "People Centred Information Systems", taught by Tom Gough, the most important feature that must be included in development methodologies is User involvement (Gough 2001). Therefore this must also be emphasised the methodology chosen. This makes sense, because the Users are the ones who are going to have to work with the system when it is completed; it seems appropriate that their needs should be the main focus of the development.

Most methodologies follow the Systems Development Life-Cycle model, which consists of defining User requirements, designing the solution, implementation, testing and then deployment, but giving a different focus on each of the different stages and different methods of completing them. Some methodologies include iterations of the steps (or the whole cycle), allowing backward movements, and some enforce evaluation of each stage by all stakeholders before moving on to the next stage. This helps the evolution of a system that best meets the User's needs, and also can help identify problems early on in the development process.

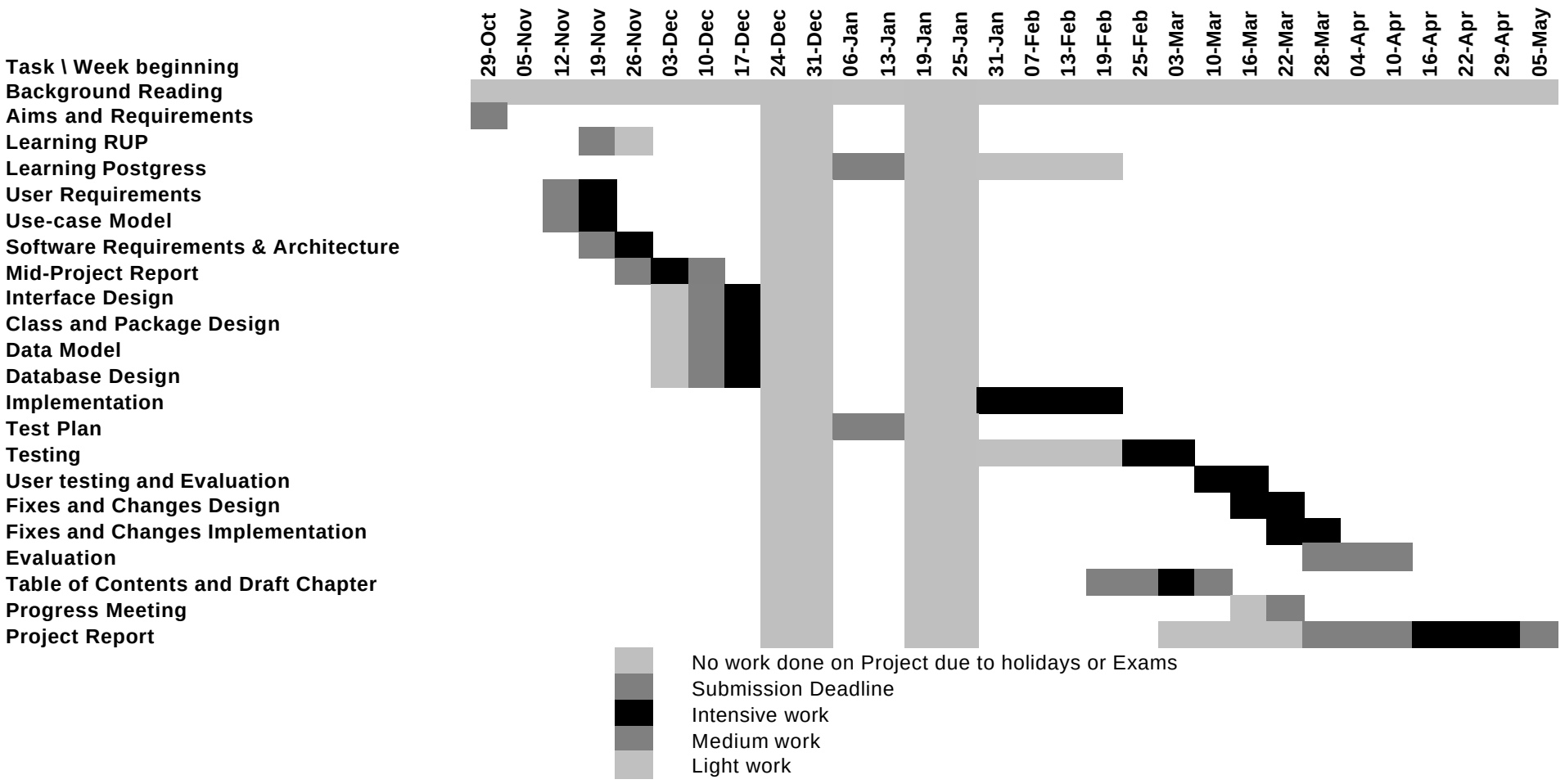
The classic Waterfall Life Cycle methodology was rejected for this development because of its linear progression through the stages, without the possibility of iterations or enforced evaluations at each stage. This means that flaws with the early stages of requirements' analysis and design are unlikely to be noticed until the end of the development process, where rectifying them will involve the massive time cost of redoing all the subsequent stages following the one with the flaw (Kruchten 1998 p6).

The chosen methodology for this project is the Rational Unified Process (RUP). This was chosen because it meets all criteria. It is a methodology that is widely used in industry, and gives focus to the User requirements and analysis stages of the development life cycle. The methodology also includes a package of supporting tools in the form of the Rational Rose Suite. This suite includes programs to aid the drawing of, and provide a template for, the various pieces of documentation that are required in a systems development process.

RUP is a cyclic model to allow iterations of the stages and the process, which fits in with the prototyping approach being undertaken. Most importantly, this methodology is flexible to the scale of the project. Within each of the major stages, there are numerous minor stages, which can be included or omitted in the implementation, depending on the scale of the project and if they are relevant. As this is a small project, only the relevant minor stages will be included.

This project will implement the following stages of the Rational Unified Process: User Requirements, Use-Case Modelling, Software Requirements and Architecture, Data Modelling, Interface Design, Class and Package Design, Database Design, Implementation, Testing, and Evaluation

Schedule


Task \ Week beginning
Background Reading
Aims and Requirements
Learning RUP
Learning Postgress
User Requirements
Use-case Model
Software Requirements & Architecture
Mid-Project Report
Interface Design
Class and Package Design
Data Model
Database Design
Implementation
Test Plan
Testing
User testing and Evaluation
Fixes and Changes Design
Fixes and Changes Implementation
Evaluation
Table of Contents and Draft Chapter
Progress Meeting
Project Report
29-Oct
05-Nov
12-Nov
19-Nov
26-Nov
03-Dec
10-Dec
17-Dec
24-Dec
31-Dec
06-Jan
13-Jan
19-Jan
25-Jan
31-Jan
07-Feb
13-Feb
19-Feb
25-Feb
03-Mar
10-Mar
16-Mar
22-Mar
28-Mar
04-Apr
10-Apr
16-Apr
22-Apr
29-Apr
05-May
No work done on Project due to holidays or Exams
Submission Deadline
Intensive work
Medium work
Light work

Plan vs. Practice

In practice this approach worked successfully with the only change being the dropping of the implementation of the second prototype. This was necessary because of an under estimation of the time involved in the design and implementation stages in the schedule. This under estimation meant that the design took 2 weeks longer that scheduled and the implementation took 4 weeks longer. These overruns led to the project being 4 weeks behind schedule (see Appendix B for the Schedule In Practice) at the end of the Implementation stage, which meant that there was no longer time for the second iteration of the development cycle. Because of the way the development methodology works, there is nothing to stop this work being done at a later date by following this documentation.

Report Structure

This report will follow the structure of the Rational Unified Process. A chapter in this report will represent each of the major stages of the methodology - Requirements Analysis, Design, Implementation and Testing. A section within a chapter will represent each of the documents produced for RUP for that stage. There will be a chapter at the end of the report that will draw the conclusions of the project, evaluate the solution and identify work that still needs to be completed.

Requirements Analysis

This chapter ties together the stages of Business Modelling and Requirements Analysis. These two separate stages of the Rational Unified Process are joined together here because the Business Modelling stage is not required or recommended for every development project, but a small subset of this stage will add value to the Requirements Analysis of this project, because it helps give the high level view of the process that is going to be 'managed'.

The Requirements are gathered by holding interviews with all the stakeholders in the project. An interview with the MSc Course Director, Lydia Lau, was held first, to identify what the main objectives of the project are and who the other stakeholders are. Then interviews were held with each of these stakeholders to gather the specific User Requirements. Once these requirements were documented they were shown to the Users to make sure that they were an accurate record of what the Users wanted.

2.1 Business Model

The purpose of Business Modelling in the RUP is to understand the structure of the organisation, and to ensure that the customers, end Users and developers have a common understanding of the organisation (Kruchten 1998 p129). Therefore, this stage will include a definition of the Business Process, a definition of the stakeholders in the project and a glossary of business terms used in this report.

2.1.1 Business Process

The Business Process is the actions that characterise the main activities of the business. In the MSc Project Process, these are the chain of events carried out by the students, the course director and the project coordinators. The process is as follows:

- In early December, the MSc Course Director and the Project Coordinators draw up a schedule for the year's projects. This schedule outlines the deadline date for all the stages of the project. The stages are completed in order.
- At the start of the academic year (September), a draft of the people within the School, who will be project supervisors and assessors, will be drawn up. This will be finalised by January.
- In December/January, the project coordinators contact the companies who have provided external projects in the past, inviting them to do so again.
- In January, the list of project suggestions is published on news. This combines all the replies from external companies and suggestions from the members of staff. This list

also includes interest areas, within which the students could devise new projects of their own.

- Early February, the students submit their preference forms; these include 3 preferences for projects and the reasons for their choices. They must include no more than 1 project from an external company, and must include an area of interest. Shortly after, the students are allocated their project, supervisor and assessor. The supervisor will be the member of staff who suggested the project, if appropriate, and supervisor and assessor will not be the same person.
- The project website is updated in February, to contain the current year's information, guidelines and schedule. Also handbooks are handed out to students, internal and external supervisors, and assessors.
- Also in February, legal forms need to be completed by students working with external companies. These need signing by the students, the company and the School. If these are not completed, the student cannot work on the project and must be reassigned to another, or the problems addressed.
- In March, the students submit the Project Aims and Objectives. The project coordinators, supervisors and assessors check this to make sure that the project is of sufficient difficulty and depth. Comments can be returned to the students and amendments made.
- In April/May, the students submit the interim report. Copies are sent to the supervisor and assessor of the project, to mark and comment upon.
- In the summer, the students are given the chance to provide feedback on the project experience. Any issues arising from this meeting need to be addressed and properly documented.
- In late August/early September, before the final report is submitted, the student has to present their projects to their supervisor and assessor. The times and dates for these are negotiated by the students and their supervisor and assessor. In some circumstances it is possible for different people to stand in for either the supervisor or assessor. This needs to be recorded.
- In early September the final Project Report is submitted in duplicate, and copies are sent to the supervisor and assessor for marking.
- Late September/early October the supervisors and assessors return the marked reports.
- In October the students are sent their provisional marks for the project, to their Out-of-School address. Students can raise issues/comments by contacting the MSc Course Director, or the Project Coordinators.

- The Exam Board meets and decides the final grades for all the projects. Issues arising from the previous step are finalised before, or at, this meeting.
- Cycle starts again.

2.1.2 Stakeholder Analysis

This is not specifically part of the RUP, although mentioned in the documentation of the Business Modelling stage. The purpose of identifying the stakeholders is to name everyone who will affected by the proposed development, and the nature of their involvement (Gough 2001 p44).

The people involved in the MSc Project process are:

- **MSc Course Director** This is the person in charge of the whole of the MSc courses offered by the School of Computing, and oversees the projects. Lydia Lau currently fills this role. She is the *Problem Owner* who commissioned this project.
- **Project Coordinator** The people involved in the management and monitoring of the MSc project process. There are currently two people in this role, Eric Atwell and Ann Roberts. They are *Primary Users* of the system.
- **Project Supervisor** The person in the School of Computing who supervises students undertaking a project. They are also responsible for marking the projects of the students they supervise. There are approximately 8 supervisors each year and they are selected from all the lecturers who teach MSc modules at the start of each year. They are *Primary Users* of the system.
- **Project Assessor** The person in the School of Computing, who is responsible for marking the projects of the students to whom they are assigned. There are approximately 8 assessors each year and they are selected from all the lecturers who teach MSc modules at the start of each year. They are *Primary Users* of the system.
- **Project Administrator** This person is responsible for all the administration involved in managing the process. This role is filled by Yasmin. She is a *Primary User* of the system.
- **MSc Students** These people are undertaking the MSc Project and are the people being 'managed and monitored' by the system. There are approx. 50 students each year. They are P*rimary Users* of the system.
- **External Supervisor** The person in the external company, who is responsible for supervising the student doing the project in that company. They will be *Tertiary Users*, because they will have no direct contact with the system.

2.1.3 Glossary Of Terms

This section is designed to meet the RUP Business Modelling goal of ensuring that all the stakeholders have a common understanding of the organisation. This is a list of business terms and their meanings, as used in the context of the project and within this report.

- **School of Computing (School, SCS)** This a department of the Faculty of Engineering in the University of Leeds, offering undergraduate and postgraduate courses in computer related subjects.
- **MSc Project** Every student studying a taught MSc degree in the School has to do a 40-credit project.
- **External Project** A project carried out for, and in, a company external to the School of Computing.
- **Internal Project** A project being carried out within the School of Computing.
- **Project Aims and Objectives (Project Scope)** A document that is submitted by the student, which will be part of the final report, which outlines the aims and objectives of the project.
- **Interim Report** A submitted document, which contains background reading and preliminary work done by the student on the project.
- **Feedback Forms** Documents which are sent out to the students to make sure that they are making progress on their projects and that they are happy with the way things are going.
- **Project Demonstration** The students demonstrate their progress on the project to their supervisors and assessors, to confirm that it is their own work, and to be given advice on how to document it.
- **Final Project Report** The final draft of the Project Report.
- **Out-of-School Address** The contact address for the student when they don't have to be in the University. Some students go home to different countries, some go travelling etc.
- **Provisional Marks** The marks given by the supervisor and assessor, pending any intervention by the Exam Board.
- **Exam Board** The committee that endorses and standardises all the examining and assessments within the University.

2.2 User Requirements

The purpose of the User Requirements in the RUP is reach an agreement and understanding between the Users and the developers, defining the functionality of the system (Kruchten 1998 p138). This is achieved by stating a specification of functional and non-functional requirements, which are checked by the Users to make sure that they are accurate and complete. The User Requirements are numbered so they can be referenced throughout the project to make sure that what is being developed is what the Users want, and also at the end of the project to evaluate how well the system meets them.

"*A requirement is a condition or capability to which the system must conform*" (Kruchten 1998 p138).

2.2.1 Functional Requirements

Functional Requirements are used define how the system will behave, specifying input and output conditions (Kruchten 1998 p138). The system is required to provide the following; in each list the order of the requirements signifies their importance, and requirements in *italic* are functions that would be an asset, but are not critical.

The system must provide:

CENTRALISED STORAGE of all the data and information relating to the MSc Project process, So that everyone knows where they can find any information relating to the process that they may require. This includes:

- **R01 -** Contact details for companies who supply external projects.
- **R02 -** The schedule for the current year's deadlines.
- **R03 -** Lists of the people who will be project supervisors.
- **R04 -** Lists of the people who will be project assessors.
- **R05 -** Lists of the students who are on MSc courses, including their preferred e-mail, their course, and the URL of their log website.
- **R06 -** Lists of all the project suggestions (external and internal).
- **R07 -** The MSc Project guidelines and handbook.
- **R08 -** Copies of the Students' Preference Forms.
- **R09 -** Allocations of the students, projects, supervisors and assessors.

- **R10 -** Copies of the Legal Forms, filled in by the external project students and the companies they are working for, and the various stages of completion (student, School, and company to sign).
- **R11 -** Copies of the Project Aims and Objectives.
- **R12 -** Copies of the Interim Report and the comments made by the supervisor and assessor.
- **R13 -** Any issues/actions, from student feedback.
- **R14 -** Details of the Project Presentations, date and who attended etc.
- **R15 -** Copies of the Final Reports, in Microsoft Word format and/or Post Script.
- **R16 -** Copies of the comments and provisional marks given by the supervisors and assessors.
- **R17 -** Contact details for the students when they are out of School.
- **R18 -** *Issues and Resolutions of queries regarding the provisional marks.*

DECENRALISED ACCESS to the information, via the intranet (or internet), for the MSc Course Director, Project Coordinators, Administrators, Supervisors and Assessors. Other important members of staff in the School might want read only access (i.e. John Davy, Sarah Fores, Bill Whyte). Students would need access to input data into the system, for example the submission of the forms electronically. The reports output by the system at the appropriate times will include:

- **R19 -** Publication of the lists of Project suggestions.
- **R20 -** Publication of the deadline schedule.
- **R21 -** Monitoring of who has/hasn't submitted the Preferences Forms.
- **R22 -** All the information required for making the student-project-supervisor-assessor allocations. For example, project suggestions, lists of supervisors, students' preferences and students' exam performance.
- **R23 -** Publication of the Project Allocations.
- **R24 -** Publication of the MSc Project guidelines and handbook.
- **R25 -** Monitoring students who have/haven't submitted the legal Forms, and who was required to.
- **R26 -** Monitoring students who have/haven't submitted the Project Scope.
- **R27 -** Monitoring students who have/haven't submitted the Interim Report.
- **R28 -** Monitoring students who have/haven't submitted the Feedback Form.

- **R29 -** Monitoring students who have/haven't submitted the Final Report, and details of any extensions.
- **R30 –** Monitoring staff who have/haven't returned the marked Reports (supervisors and assessors).
- **R31 -** Out-of-School contact addresses for the students, to distribute the provisional marks.
- **R32 -** *Issues raised via the Feedback Form and their solutions.*
- **R33 -** *Issues raised about the provisional marks and their solutions.*
- **R34 -** *Report for the Exam Board's final meeting.*
- **R35 -** *Automated letter writing to the companies for external projects.*

ADDITIONAL REPORTS. Also at anytime, the system will be required to produce the following reports:

- **R36 -** Lists of students on external projects.
- **R37 -** The student information – detailing their progress, what was submitted and when, their project, supervisor, assessor and external company if appropriate. Ordered/viewed by student, supervisor, assessor, external projects, or whole year.

SECURITY. The system should provide the necessary security so only the people who should have access to, and be able to change data, can do so. The project coordinators and course director should be able to see and edit all the data in the system. Other members of staff in the School should be able to see all the data, but not edit it. Anyone outside the School shouldn't have any access. Students should be able to submit personal on-line forms etc, but shouldn't be able to see other student's information. Students should only be able to see the following information:

- **R38 -** The schedule for the current year's deadlines.
- **R39 -** Lists of all the project suggestions (external and internal).
- **R40 -** The MSc Project guidelines and handbook.
- **R41 -** Their own Project Preference Forms.
- **R42 -** Allocations of their projects, supervisors and assessors.
- **R43 -** Copies of the Legal Forms filled in by the external project students and the companies they are working for, and the various stages of completion (student, School, and company to sign).
- **R44 -** Copy of their Project Aims and Objectives.

- **R45 -** Copy of their own Interim Report and the comments made by the supervisor and assessor.
- **R46 -** Details of the Project Presentations, date and who attended etc.
- **R47 -** Copies of the Final Reports.
- **R48 -** Comments and provisional marks given by the supervisors and assessors on their own project.
- **R49 -** Contact details for the students when they are out-of-School.
- **R50 -** *Any subsequent issues/actions on student feedback*
- **R51 -** *Issues and Resolutions of queries regarding the provisional marks.*

2.2.2 Non-Functional Requirements

The Non-Functional Requirements are used to define the system specification in terms of Usability, Reliability, Performance, and Supportability (Kruchten 1998 p138-9). These are very subjective requirements, therefore it is not so easy to identify when they have been met.

The most important Non-Functional Requirement of the system is that it shouldn't be any harder or more hassle to use than the current system. This is because the people that will have to put extra effort into using the system (i.e. the assessors and supervisors) aren't necessarily the ones who will see the functional benefits.

Reliability is very important for the same reason; people will not be persuaded to use the new system if it is unreliable. Therefore the system must be as reliable as possible within the School's infrastructure - it would be unrealistic to expect it to be more reliable than the infrastructure within which it is running.

The system must offer a level of performance that will not put Users off using it over the current system. This means that it must be able to cope with the traffic that is expected – ~50 students and ~10 members of staff all using the system at the same time, and must not run at an unacceptably slow speed – i.e. >2 seconds to respond to an User's action.

The system must offer a level supportability that allows problems to be located and sorted with minimum disruption to the service. This includes the independence of actions, where possible, so that if one function isn't working then it shouldn't stop another unrelated function from working. It also means that any coding must be consistent, following an appropriate coding standard, to allow someone to be able to workout how it has been implemented, and locate and correct any problems easily.

Also the system must be scalable and expandable. This means that it should be simple to add an extra function, without needing to rewrite/redesign the whole system, and the system should be able to cope with a reasonable increase in the number of Users without a detrimental effect on performance.

2.3 Use Case Model

The Use Case Model is used in the RUP as a method of defining how the system will behave via simple threads of User interactions, and is easily understood by Users and developers alike (Kruchten 1998 p142). As with most of the artefacts produced in the RUP, the Use Case Model is used later to make sure that all the required behaviour is represented in the System Design, and in Test to make sure that this behaviour has been implemented (Kruchten 1998 p99).

Use Cases are described by detailing all the actions of the system: who initiates it, what data is required, what output is created, who has access to it, and any prerequisites and follow-up actions there may be. The Use Cases are derived from the Business Process (Chapter 2.1.1).

Use Case 1: Supervisor and Assessor Lists
At the start of the academic year, the list of people who will be the project supervisors and assessors is drawn up.
Actors: Senior members of the School of Computing.
Data requirements: The lists of lecturers in the School that are connected with the MSc course.
Outputs: The lists of the supervisors and assessors for the year.
Access: Students shouldn't have access to these lists.
Prerequisites: None
Follow up actions: Project Allocations (6).

The full Use Case Model is in Appendix C.

2.4 Software Requirements

Any solution developed will have to fit in with the architecture of the School of Computing, so that it is maintainable, scaleable and supportable by the School's technical staff. The architecture is a 3-tier client-server application (see Figure 1). This solution offers client-side platform independency. Users can access the system from any machine with a web-browser that is connected to the Internet, either in the School/University or elsewhere, no matter what

the operating system. There will be no setting up or installation required to the client machine(s).

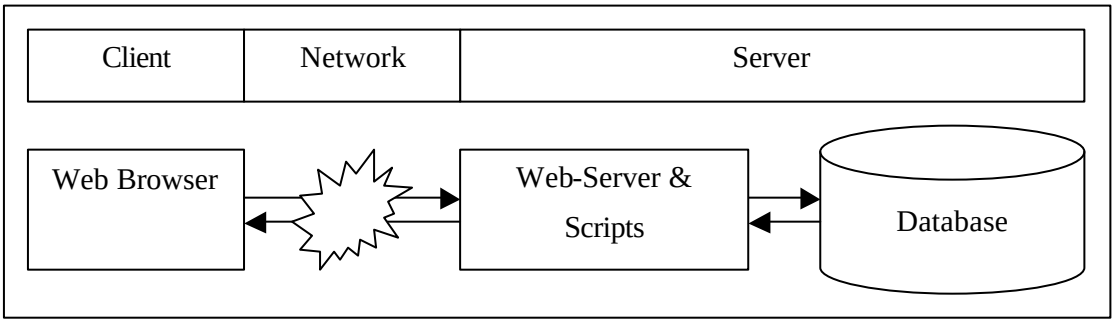


Figure 1 – 3 Tier Architecture

The server side software requirements are a web-server, with the facility of running server side programs, and a database.

The client-side software requirements are, very simply, a web-browser. Internet Explorer, Netscape Navigator, Konqueror, Mozilla etc, should all be supported. Also it would be good to support text-only browsers such as Lynx, but since all the Users are on a Windows or Linux machine, then this is not critical.

Design

The purpose of Design in the RUP is to translate the User Requirements into an implementable specification of the system. The design must be complete enough to be unambiguous, but the level of detail if goes into is open to interpretation by the Designer and Implementer (Kruchten 1998 p150). This Design will be documentation of the Database, Functionality, Interface and Security proposed for the system. The details of the programming methods used and the database relationships implemented will be documented in the Implementation section.

3.1 Database Design

A database will be used to meet the User's Requirement of providing centralised storage. The decentralised access will be achieved by the clients accessing the data in the database via the website interface.

The database will be implemented in a Relational Database Management System (R-DBMS), though the design is independent of the exact R-DBMS used. The design will cover the identification of the entities and relationships that will be present and proof that these relations will conform to either Boyce Codd Normal Form (BCNF) or 3rd Normal Form (3NF).

3.1.1 Entities

The entities are taken from the list of User Requirements on the data that must be stored by the system. The numbers in brackets identify the requirement(s) that are met by that relation. The entities are highlighted in **bold** and attributes are in *italic*. These will make up the relations in the database.

The **Schedule** has s*ubmission*s with *deadline* dates **[R02]**.

A **Student** has a *Username* (which uniquely identifies them and is their school email address), a *name*, a program of study, an out of term time contact *address*, an *URL* for their Log website, a *supervisor*, an *assessor*, a *project*, and the submissions; *Preferences Form*, *Aims and Requirements Form*, *Interim Report*, *Demonstration* and a *Final Report*, which each have the *date* that they were submitted and any *comments* by members of *staff*. For each of the submissions there could also be an extension to the deadline, therefore the new *deadline* needs recording, along with the issuing member of *staff*, the *date* issued and any *comments*. If

the student is on an external project they will have a *Legal Form*, and may have a preferred *e-mail* address **[R05, R09 and R17]**.

A member of **staff** has a *Username* and a real *name.*
An **assessor** is a member of staff and has many *students* to assess **[R04]**.
A **supervisor** is a member of staff and has many *projects* suggestions and many *students* to supervise **[R03]**.

An external **company** has a *name*, a *contact*, an *address*, and many *projects* suggestions **[R01]**.

A **project** has a *title*, a *description*, and a *suggestor*, who could be a supervisor, an external company, or a student **[R06]**.
An **Area of Interest** has a *title* and a *suggestor*, who is a member of staff.

Feedback is given by *student*s and their *comments* are store along with the *date* **[R13]**.

A **Preference Form** has the 1^{st}, 2^{nd}, and 3^{rd} choice projects and *reasons* why the student has chosen each project, and the *date* that it was submitted. *A* member of *staff* may make *comments* and the *date* of these would need recording **[R07]**.

The **Aims and Objectives Form** will contain the *project title*, the *aims,* the *objectives*, the *minimum requirements* and the submission *date*. *A* member of *staff* may make *comments* and the *date* of these would need recording **[R11]**.

For the **Legal Documents** the *date* that the student signs the Undertaking Document needs recording and the date that the Placement/Project Document has been signed by both the school and the external *company* **[R10]**.

The **Interim Report** will contain the *student*, the *format* (postscript or MSWord), the submission *date*, a reference/pointer to the *file*, and fields for comments by the *assessor* and *supervisor* **[R12]**.

Details of the **Demonstration** will need to record the *student*, the *Supervisor* and *Assessor* (or member of staff, if different) who attended the demonstration, as well as the *time* and *date* that it took place **[R14]**.

Final Report will have the *student*, the *format* (postscript or MSWord), the submission *date*, and a reference/pointer to the *file* **[R15]**.

An **Extension** may be given by a member of *staff* (usually the Project Coordinators), to a *student* for a particular *submission*. This will have the *date* the extension was given and the new submission *date*.

The **Assessment** of the final report is carried out by the assessor and supervisor jointly. This includes recording the *student*, the presented *deliverables*, and *comments*. Marks are given out of 100, and are spilt under the following headings; *Understanding* (20), *Solution* (40), *Evaluation* (20), *Write-Up* (15) and *Reflection* (5) **[R16]**.

These entities and attributes, and their relationships are represented in the Entity Relationship (ER) Diagram in Appendix E.

3.1.2 Normalisation

Normalisation takes a relation schema through a series of tests to certify whether it satisfies a certain Normal Form (Elmasri 2000 p483). There are 6 different classes of Normal Form (1NF to 3NF, BCNF, 4NF and 5NF), each being more restrictive than the previous – for example a relation in 4^{th} Normal Form is implicitly in 1^{st} 2^{nd} 3^{rd} and BCNF. It is desirable for relations to be in a high Normal Form, as they are then more likely to maintain the data integrity and consistency.

To prove that a relation is in a specific Normal Form we need to identify the functional dependencies in the relation and then we can test it against the definition of the required Normal Form. 4^{th} and 5^{th} Normal Form are for relations with Multivalue and Join Dependencies, which are not present in these relations, therefore we are looking for the relations in this database to be in either 3NF or BCNF.

For a relation, R, to be in 3NF then wherever there is a non-trivial dependency, $A_1 \ldots A_n \rightarrow B$, it is the case that either $\{A_1 \ldots A_n\}$ is a superkey or that B is a member of a key. Where a superkey is defined as any set of attributes the contains a key, and a set of attributes is a key if, and only if, it determines all other attributes in R and is minimal. A relation is in BCNF when wherever there is a non-trivial dependency, $A_1 \ldots A_n \rightarrow B$, it is the case that $\{A_1 \ldots A_n\}$ is a superkey of R. By definition any relation in BCNF is also in 3NF (Date 1995).

Using this definition a relation can be proved to be in BCNF as follows:

The Student relation has the attributes – username, name, program, email, URL, project, supervisor, and assessor – and the functional dependency – username determines name, program, email, URL, project, supervisor and assessor – username is the key of the relation and there are no non-trivial superkeys; therefore the relation is in BCNF.

All the other relations can be proved to be in BCNF in the same way.

3.1.3 Indexes

Indexes are used in DBMS to create lookup tables for certain attribute of the tables. These lookup tables are used when a query with a condition on that attribute is run against the relation.

Each relation in a database can have one primary (or clustering) index, which defines the attribute that will dictate the order in which the records in that table are physically stored on the computers Hard Disk Drive (HDD). In addition to the primary index, secondary indexes can also be set up, which provide a look-up table of the attribute's values with pointers to the records on the HDD (Mott 2001 p9).

The potential queries that will be run against the relations need considering when deciding on the indexes, because they can have a great effect on the performance of the database. The primary index needs special attention because there can only be one, so this should be the attribute that is most often used in the 'select' or 'join' condition in the queries. Secondary indexes aren't quite as critical, but adding unnecessary ones can impair the performance of inserts and updates to the relation.

The following indexes are required in this database (Key: *italic* = primary index, underline = secondary index):

Schedule {*submission*, deadline}
Company {*name*, contact, address}
Student {*Username*, name, email, URL, project, supervisor, assessor}
Project {*title*, description, suggestor}
Area {*title*, suggestor}
Staff {*Username*, name}
Extension {*student*, submission, deadline, issuer, date, reason}
Comments {*student*, date, comments, solution, staff, date2}
Preferences {*student*, project1, project2, project3, date, staff, comments, date2}

Objectives {*student*, title, aims, objectives, min_req, date, staff, comments, date2}
Legal Form {*student*, date, company, school}
Interim Report {*student*, format, date, file, a_comments, s_comments}
Demonstration {*student*, Supervisor, Assessor, time, date}
Final Report {*student*, format, date, file}
Assessment {*student*, date, deliverables, comments, marks}

3.2 Functionality

This section details all the functions that will be implemented and how they will all fit together. This is depicted in Appendix D by a 'site map' showing how the Users can navigate though all the functions of the system.

The following reports will be generated by the system from the information stored in the database. These reports will be available to different User groups, with different update and edit permissions as detailed in Table 1.

Reports are defined as any output from the system to the screen that contains variable data.

Table 1: Report Read-Write Permissions of the different User groups.

Report Title	Student	Staff	Coordinator
Student List	×	×	RW
Student Details	RW	R	RW
Staff List	×	×	RW
Staff Details	×	R	RW
Company List	×	×	RW
Company Details	×	×	RW
Missed Deadlines	×	×	R
Project Schedule	R	R	RW
Project Suggestions	R	RW	RW
Student Feedback	W	×	R

The reports are detailed as follows. The **bold** numbers in the brackets relate to the User Requirement(s) that is satisfied by that report.

- Student List ~ This report lists all the students that are undertaking the MSc Project and shows their progress through the process. The report will show, via ticks, crosses or N/A, which of the following steps have been completed: submission of preferences, objectives, legal form, interim report, demonstration, final report, and assessment, and the assignment of supervisor, assessor and project. **[R21, R25, R26, R27, R28, R29]**

- Student Details ~ This will details all the student's personal details (name, program, email etc), their project details (title, supervisor and assessor) and details of their submissions and any extensions. Students will be able to update their preferred email address and the URL of their log website, and the Project Coordinator will be able to update the Students' name and program, and be able to assign projects, supervisors and assessors. **[R21, R22, R23, R25, R26, R27, R28, R29, R31, R37]**
- Staff List ~ This report lists all the members of staff that are supervising or assessing MSc projects this year (or have done so in the past). It lists their names and the number of students supervised and assessed and the number of projects that they have suggested this year. None of the information contained in this report can be altered, but new members of staff can be added to the list.
- Staff Details ~ This report will display the details of the member of staff (name, email etc), the details of all the students that they are supervising and assessing (deadlines missed etc), and any projects/areas of interest that they have proposed. **[R30]**
- Company List ~ This report contains a list of all the companies that have supplied projects in the past and a count of all the projects that they have suggested for this academic year. The Project Coordinator will have the ability to add new companies to this list, but none of the other details contained can be updated.
- Company Details ~ This report contains all the details of the selected company (name, contact details and comments) and a list of all the projects that have been supplied this year. The company contact details can be altered and saved, and, if the company doesn't have any project suggestions associated with it, it can be deleted from the system.
- Missed Deadlines ~ This report lists all the deadlines that have past and all the students that have yet to submit, and details of any extensions that have been granted. None of the information contained in this report is editable **[R21, R25, R26, R27, R28, R29, R30]**.
- Project Schedule ~ This will display the order and deadlines for all the submissions due in the course of the project. The Coordinator will be able to alter the deadline dates. **[R20]**
- Project Suggestions ~ This report will contain a list of all the details of the projects that have been suggested by members of staff and external companies. It will also contain the areas of interest. The members of Staff can add their own suggestions and the Coordinators can add and remove suggestions from the lists. **[R19]**

- Student Feedback ~ This report lists all the comments that have been submitted by the students. These comments are anonymous, and cannot be edited or deleted. **[R32]**

3.3 Interface Design

Issues to be addressed by the interface design are the layout of the screens, the options on the menus and how the system will help guide the Users through their tasks. The design must consider Health and Safety (H&S) and Human Computer Interaction (HCI) guidelines on the ergonomic, cognitive and social aspects of the system (Matravers 1998 section 4).

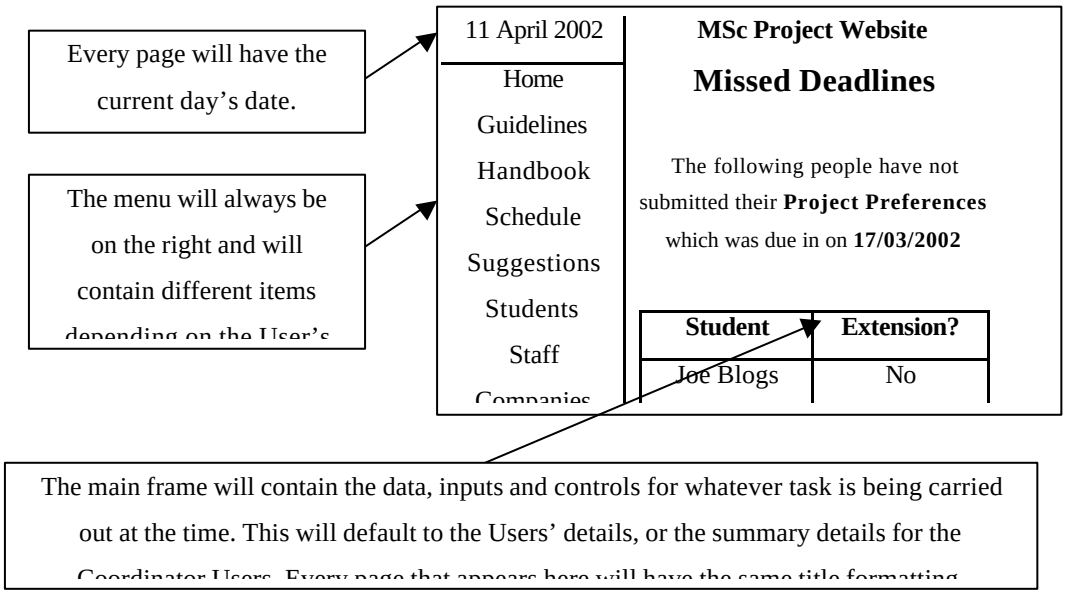
The main implications of these are:

- Colours – not using combinations of colours that cannot be distinguished by colour-blind people, or colours that do not contrast well, as this is likely to cause eyestrain (i.e. yellow-white, blue-red, etc).
- Screen Layout – this should consider the grouping of information to minimise eye movements, and therefore eyestrain.
- Workflow – tasks should be designed to flow though to completion in an expected manner. Keyboard to mouse hand movements should be minimised to reduce the chances of Repetitive Strain Injury (RSI).
- Dialogues – the 'conversation' between the computer and the User (on screen instructions etc) should be designed to be informative and at a level that the User can understand. They should be concise and guide the User through the tasks and not hinder or confuse them.
- Errors – these should be avoidable, recoverable and informative. They should not panic the User – i.e. "The 'Name' field cannot be left blank." not "Fatal Error: 12654."
- Consistency – The same action should have the same effect in all states.
- Learnable – The system should support novice Users in learning the system, while not hindering expert Users. Actions and their consequences should be implicit.

The Health and Safety issues relating to the workstation are beyond the scope of this project. The Users are already using computers in their jobs and day-to-day activities; therefore it is assumed that these H&S requirements have already been met. This not a good assumption to make, because it is the responsibility of all Information Systems developers to ensure that the H&S requirements have been met and not to assume that someone else has done it.

3.3.1 Screen Layout

There are three distinct groups of Users (Students, Staff and Coordinators) and each will be using the system to carryout different tasks. Therefore there will be three different interfaces into the system, each will have the same layout, but different options on the menu signifying the different tasks that they will be performing.

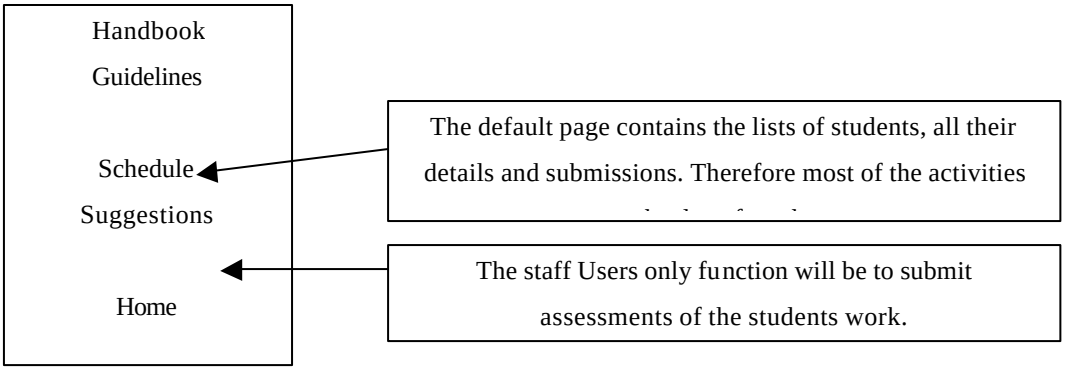


3.3.2 Menus

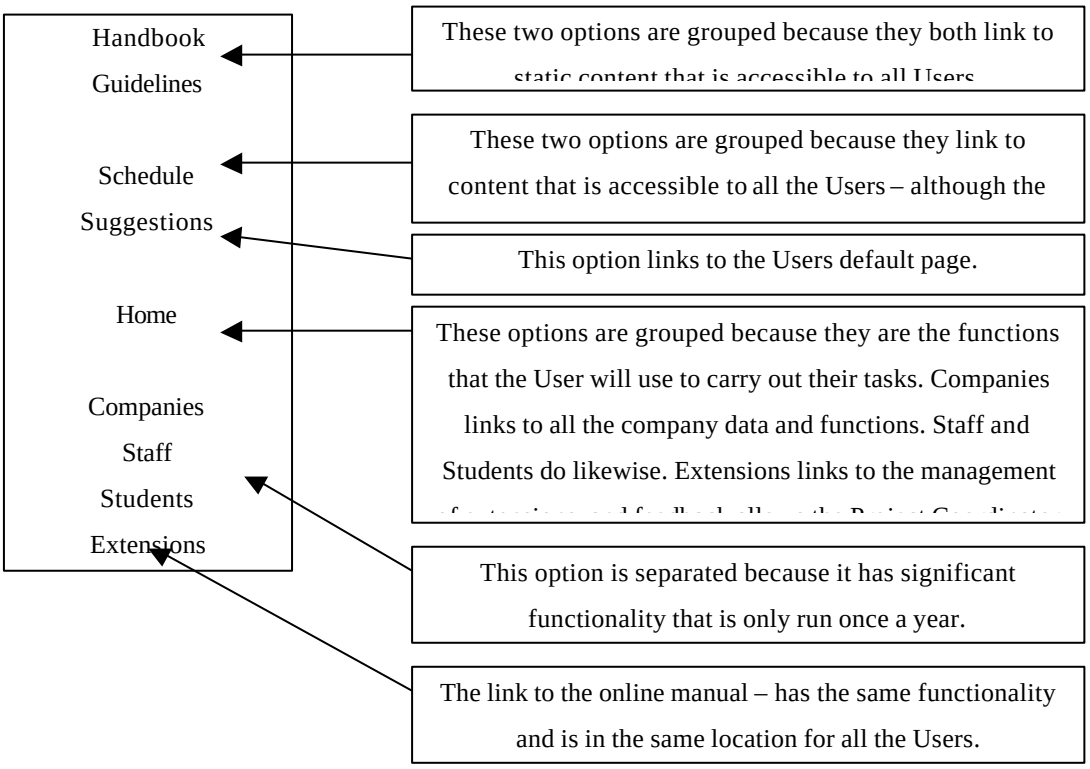
The menus will be different for each User group, signifying the different functions that each will be carrying out. There will, however, be some options that are common to all the User groups: the links to the static HTML pages that make up the current MSc Project Website containing the Project Guidelines and the Project Handbook, the links to the Project Schedule and the Project Suggestions, the link to the Online Help which will contain the User manual for the systems, and the 'Home' link which will return the User to their default page.

The menus will be laid out as follows:

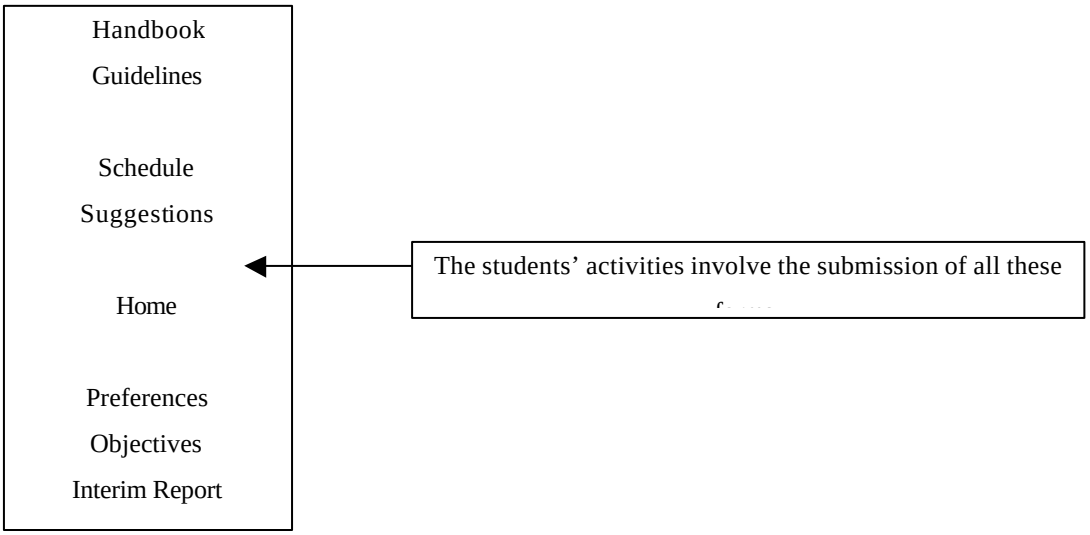
Supervisors and Assessors Menu



Project Coordinators Menu



Students



3.3.3 Data Input

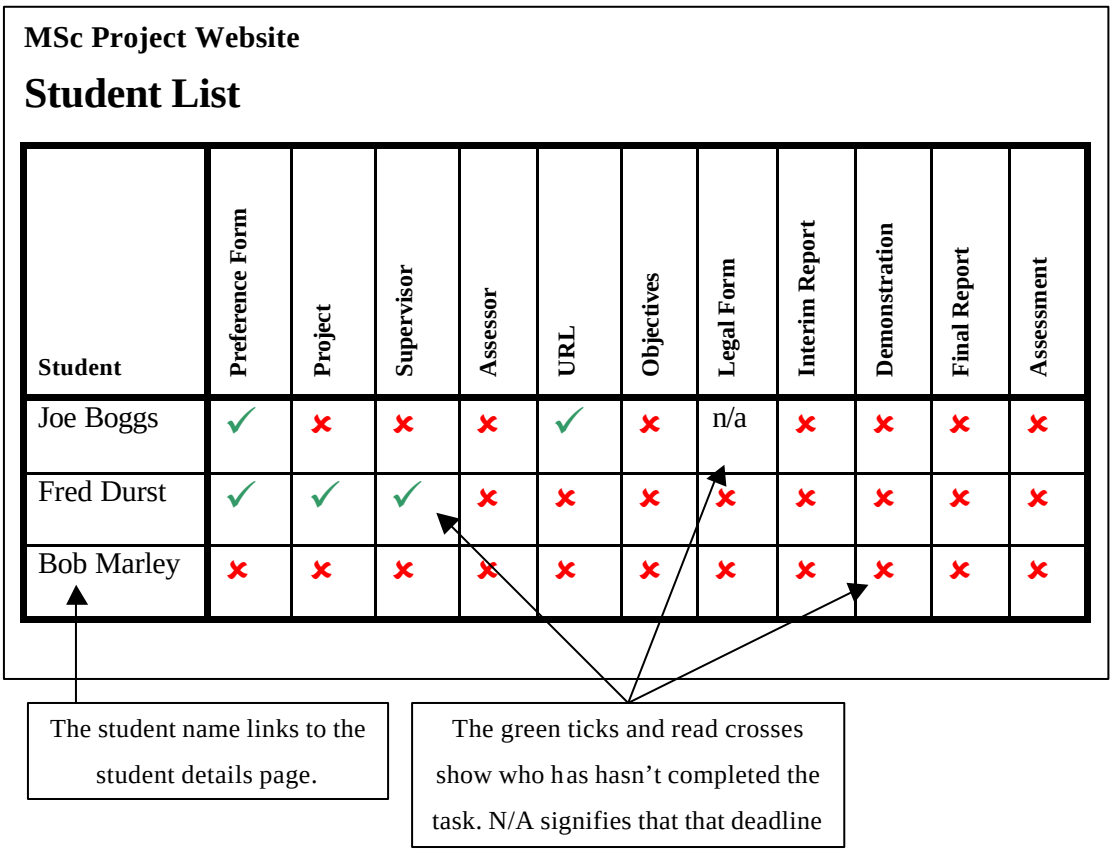
The data will be input into the system via the use of HTML forms. The pages will be as close a representation of the paper documents that are currently used as is possible. This negates the need to design the format of these forms since they have evolved over many years usage.

Any of the input fields where there are a finite number of options then a HTML 'select' will be used so the Students can choose their option from a list generated by the system. Also where the data is predetermined (i.e. the students name and program) then these fields can be populated automatically by the system.

Students can fill in the required fields on these documents, submit them using the 'Submit' button, and these will then be saved by the system. When a User wants to view a saved document all the fields will be populated with the saved data. See Appendix F for copies of the original paper documents.

3.3.4 Report Layout

The Student List report will be formatted as shown here. The designs for the output of all the other reports generated by the system are detailed in Appendix G.



3.4 Security

The site needs to have some security to ensure that it accurately records which student is submitting work, and also to control access to sensitive/personal data. The security will be

handled by the Users logging onto the website. To avoid unnecessary disruption to the Users, they will be made to logon only once during each session, and the Username and password will be the same as their current SCS Usernames and passwords.

Because the Hyper-Text Transfer Protocol (HTTP) is stateless – meaning each request received by the Web Server is treated independently of all other request, and details of one request cannot be known to another request – once the User has logged on to the system their user details will need transmitting to each of the functions that they access so that the script implementing the function will know who the user is.

Implementation

The purpose of the Implementation section is to document decisions that need to be made on the choice of the technologies to use and how the system is actually implemented.

4.1 Technology

4.1.1 Web Server

The School currently has Apache running on Linux and Microsoft's Internet Information Server (IIS) running on Windows 2000. Therefore a choice needs to be made between these two, firstly by looking at the strengths and weaknesses of the Apache web-server, then at IIS.

Apache is a free web-server originally developed for the Unix operating system, but has now been transported over to Windows, and many others. The fact that it is free, and an open source active development project, means that it is the most popular web-server on the Internet - 60% of public websites (Rajagopal 2000). This popularity means that there are many User community groups where development help can easily be had and there are always useful discussions going on about the deployments of Apache. Other advantages of Apache are that it incorporates a Perl Interpreter, used for CGI programs, and other modules are freely available to expand on the core web-server, such as the Tomcat Java plug-in.

Some of the drawbacks of Apache are that it requires quite a lot of initial configuration and tuning. There is no Graphical User Interface (GUI) for the administration functions of the server, and becoming competent at optimising the configuration takes quite some time. This can be overcome by referring to one of the many FAQ sites and newsgroups available on the Internet that are related to the Apache web server. These configuration problems are also not an issue in this project because Apache has already been installed and configured by the School.

There are many alternatives to the Apache web-server, each offering its own advantages, coming from various different vendors. The major competitor in the commercial market is Internet Information Server (IIS), by Microsoft. This offers a GUI interface for administration functions of the server, but its major drawback is that it is restricted to the Microsoft Windows 2000 operating system, and also requires other Microsoft products such as Active Directory.

The functional advantages that IIS has over Apache can be rectified by installing modules that are available from third parties, and these modules are to be included in the next major release (v2.0) of Apache.

The bottom line in this discussion is that ISS will provide the same functionality to this project as Apache. So, given the extra development help that is available for the Apache web server, this will be use for the implementation.

4.1.2 Security

The security will be implemented via Basic Authentication, which is included in the Apache web server. This authentication involves the User's browser being prompted for a User name and password every time there is a request for a document from a protected directory on the web server. The browser will prompt the User for their details on the first request, and will then store the details and send them together with all requests for documents from that realm. This authentication method also allows the Users to be set-up into groups. This will allow the students, supervisors and assessors, and the project coordinators to be set up in different user groups.

Upon deployment, the User password file used by Basic Authentication will be the password file used by Linux. This will allow the Users to have the same password for this system as they do for the network, and this file will have the high security associated with the Linux password file. Group files will be set up manually and will list the User ID's that belong to students, staff and project coordinator groups.

4.1.3 Programming Language

There are several different options of server side programming technologies to choose from for the implementation of this project, each with its common implementation language. The choices are between Common Gateway Interface (CGI) implemented in Perl, although C/C++ can and are often used, Java Servlets that are implemented in Java, and Java Server Pages (JSP) or Active Server Pages (ASP) that are implemented in Java and Visual Basic respectively.

The technology behind all of these solutions is very similar, HTTP requests are addressed to the program scripts from the client browser, and the web-server starts the programs, which run through to completion, sending HTML back to the browser. JSP and Servlets require a 'plug-in' for the web-server that has the Java Virtual Machine (JMV), the interpreter and execution environment for the programs. The Java 'plug-in' for Apache, called Tomcat, is

freely downloadable from the Apache website and is simply installed and configured. ASP requires that a Visual Basic 'plug-in' be installed in the same way as JSP. In fact JSP and ASP are essentially the same technology except for the programming language used – PHP is also very similar in this respect – therefore, for the sake of argument, this discussion will concentrate on these three technologies - Java Servlets, JSP and CGI.

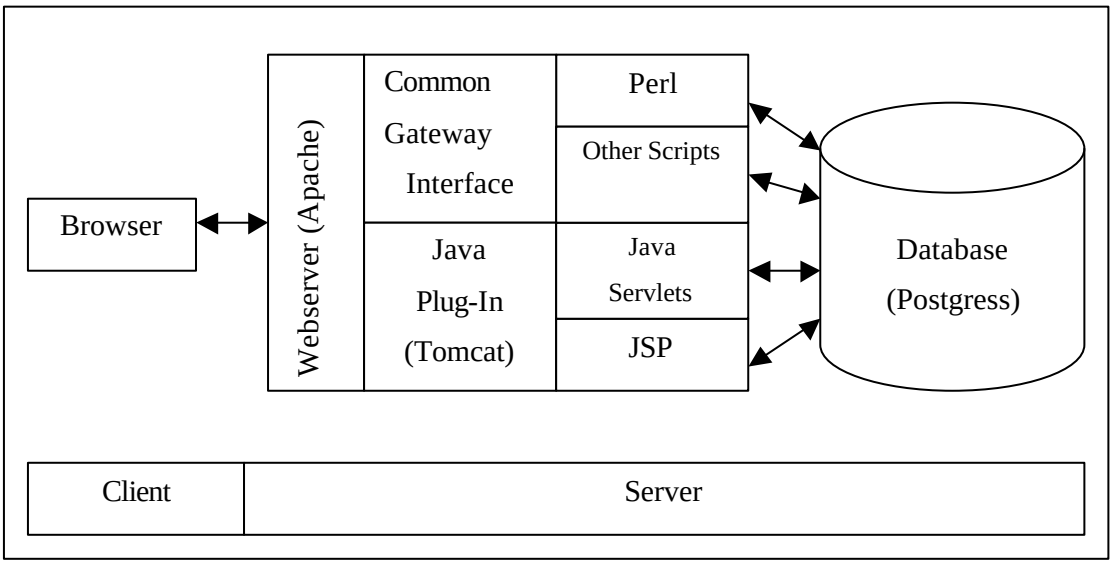


Figure 2 – Client Server Technology

CGI Perl requires the machine that the web-server is running on to have the Perl Interpreter installed, which is the case on the School's machine. When a Perl script is run then the web-server has to create an instance of the interpreter, which then compiles and runs the script. This overhead each time a script is run can be expensive, 5/6MB of memory each time a script is run which can have a detrimental effect on performance.

Using Mod-Perl overcomes this problem by only loading the interpreter once and compiling the script when it is first used, then using this compiled version each subsequent time. This makes Mod-Perl very similar to the Java solutions, which load one instance of the JVM and stores all the classes, so that there is only a hit loading the classes into memory the first time they are used (Apache Org 2001). In other performance issues Perl, which is implemented using C, is generally accepted to be faster than Java but since this isn't going to be a speed critical application then this will not make a lot of difference.

Java Servlets are Java programs that are based around and implement the Servlet Interface. The Servlets run and output HTML back to the web server for it to return to the client browser, in the same way as CGI.

JSP on the other hand, have the programming code embedded in the HTML pages and the web server processes the pages returning the HTML and executing the code.

Since there is nothing to choose between the performances of the options then we have to look at other factors. Java is a very strict Object Orientated (O-O) language, which means that for larger programs then the code becomes tidier and more structured. Perl has modules to address the concept of object orientation and code reuse, but it isn't as tidy as the Java option which means maintaining the scripts becomes more difficult. Its simplicity makes Perl ideal for less complicated applications. Java is also very commercial, unlike Perl where you get communities of developers who share modules of code freely between themselves (Pisoni 1999).

For this project there seems to be very little between the two technologies and languages, Perl is the more powerful of the two languages at text manipulation, which is going to be the major part of the programs' functions, and Java is more formally structured.

One of the requirements was that the system fits within the School's infrastructure and currently JSP and Java Servlets are not supported, therefore the implementation will be done in Perl.

4.1.4 Database Management System

Choosing which Database Management System (DBMS) to use follows on from the choice of the web server because this determined that the implementation would be carried out on the Linux platform instead of Windows 2000.

The School currently runs three major DBMS – Microsoft SQL Server, MySQL and PostgresSQL (Postgress). Since SQL Server is on the Windows 2000 platform, and it is wanted, for ease, to carry out all the implementation on the same platform, the choice will be restricted to MySQL and Postgress, which are on Linux.

The first point found was that there is a limit the maximum 8kb row size in Postgress. This is equivalent to 8,000 characters, so might not be of major concern but will have to be taken into account. According to research done by Tim Perdue (Perdue 2000) into how to choose between MySQL and Postgress, there is also an issue with importing data into relations that have a counter field, or auto number.

Other issues raised by the paper were in relation to the performance and the integrity of the databases, when subjected to high usage. A summary of the findings is that Postgress allows the most concurrent connections at approximately 120, while MySQL only copes with approximately 50, before starting to throw errors. This wouldn't be an issue for this project alone, but the instance of the database will also be used for other things in the School so that MySQL's capacity might become a little restrictive. MySQL is 30% faster that Postgress, but MySQL locks a whole table when doing an update or insert, whereas Postgress only locks the affected rows. This could cause adverse effects in MySQL, if there are concurrent inserts/updates etc.

Functionality differences between the databases could be more of an issue. MySQL doesn't support 'rollbacks', allowing changes to be undone, which is useful if some part of a transaction fails. Since this project is unlikely to have any complex dependencies in the transactions then this will not be an issue. Foreign Keys and Triggers are also available in Postgress and not in MySQL; these provide some very handy integrity checking and enforcement on inserts, updates and deletes, and will be used for this project.

The added functionality and features of Postgress make up for the reduced speed. Also the increased concurrent contention limits in Postgress make it more likely to cope with increased traffic in the future, although in reality the throughput of 15 queries a second (500,000 per day), which both could cope with, is unlikely to ever be required of the system. If this limit is ever reached then SQL Server could be used and, since the program implementation is independent from database, there would be little/no changes required in the coding.

Therefore Postgress has been chosen for the implementation of this project.

4.2 Script Implementation

This section will describe how the Implementation of the scripts was undertaken, the structure of the programs, any programming standards implemented and any Perl Modules used.

4.2.1 Structure

Perl scripts will be used to implement every function of the system, such as data input or report generation. These scripts will deal with the output of the HTML, any validation that needs to be carried out on the data input by the User, and querying the database.

Functions that can be reused by other scripts are identified and collected in a Perl Module that is used by all the Perl scripts. Constant values, such as the address of the database, will also

be grouped together into a module, so they only need to be defined once for all the scripts. This means that any changes that are necessary to these functions or values need only to be done once and all the scripts will automatically use the new version. These modules will be grouped in the 'common' package, while all the other scripts will be arranged in packages depending on which User group has access to them (i.e. Coordinators, Staff and Students).

The implementation will be arranged into the directory structure show in Figure 3. Static HTML such as the index pages, which will split the page into two frames (one for the menu and one for the functions), and the menus will be stored in the student, staff and admin directories as appropriate. Images for the options on the menus will be in the images directory. The scripts will be placed in the appropriate directory, with the Basic Authentication security set up, under the 'cgi-bin' directory. All the common functions and modules will be stored in the common directory.

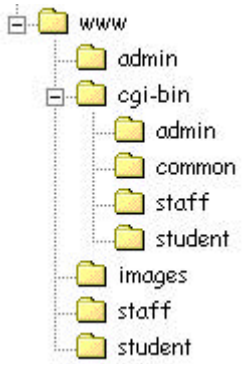


Figure 3 - Directory Structure

4.2.2 Modules

To remove the need to implement complex functions the following Modules that are included with the Perl distribution, will be used:

- **DBI** ~ The Database Interface module provides an interface to query the database without the code having to know about the specific database implementation. A DBI object is created and passed strings containing the database connection string, the database Username and password, and any options. The connection string contains the driver, the name, the host computer's name and the port that the DBMS is listening for connection from. The connection string is "dbi:Pg:dbname=isyis;host=csdb;port=5432" for this implementation, but if the database moves to a different computer or changes altogether then all that would need changing would be this string.
- **CGI** ~ This is a specific module designed for using Perl in CGI Scripts. It gives easy access to all the Environment variables associated with CGI programming and to the data that is sent by the Users. It can also be used to generate the HTML to be returned and set cookies etc on the User's computer. This module also has a useful feature to help in the testing and debugging of the scripts, called "CGI::Carp". This allows details of fatal errors occurring within the program to be returned to the browser, which helps during the development, but can be removed and replaced by more appropriate error messages for the final deployment of the system.

4.2.3 Code Standards

Since Perl's syntax is a mixture of several different languages, such as C and Visual Basic, and there are many ways of writing the same code, this set of Coding Standards was devised to maintain a consistency in the implementation. An example of a script following these standards is included in Appendix H.

- Start of Script – all scripts will start with the line identifying it as a Perl script and pointing the web server to where the Perl Interpreter is installed (i.e. `#!/usr/bin/perl –wT`). The 'w' and 'T' options turn on warnings about possible logical mistakes in the script (w) and turns on 'taint' checking (T), which is recommended for Perl Scripts that are used as CGI Programs.
- Comments – every script will start with a comment block with a brief description of what the script does, the author, and the date and details of any changes that has been made.
- Indenting – will follow the standard C-style of indenting all lines of code within a control, such as a loop, by one tab (3 spaces).
- Naming – variables will be named in lower case, with the fist letter of new words having a capital (i.e. `$studentCounter`), constants will be in all upper case with words separated by an underscore (i.e. `$STUDENT_NAME`), function names will start with a capital letter and then lower case, with new words starting with a capital (i.e. `&GetNameFromDB()`), packages will be single words in all lower case (i.e. `common`), modules will be named with a capital first letter and a .pm extension (i.e. `Constants.pm`), and the scripts will be named in lower case characters with a .cgi extension (i.e. `studentDetails.cgi`).
- IF statements – these can be implemented in various different ways so for consistency they will all be done in the C-Style 'full' format – for example:

```
if ($testVariable ne "Some string")
{
 &TheBitOfCodeToExecute();
}
```

- Declarations – all global variables that are used will be declared, assigned a default value at the start of each script and given a comment about what it is used for. Variables that are local to a function will be declared at the start of the function with the lexical scope specification '`my`'.

- Strings – will be enclosed by double quotes (“), unless single quotes (‘) are specifically required. This allows variables to be placed inline with the string and the value of variable be substituted for the variable name (i.e. `$string = “Number = $number”;`)
- Function calls – will be preceded by optional ampersand and followed by opening and closing brackets (i.e. `&Function();`). These will help identify the self-defined functions from the rest of the code and the brackets will enclose any of the arguments that are required.
- Print – when printing a string that spans more that two lines of code the following statement will be used to remove the needs for the text to ‘wrap’ around many lines of code or for the repeated use of the print command.

  ```
  print <<END;
  All the text can be written here and formatted as required, and is terminated by...
  END
  ```

- Fatal Errors – commands that could cause major problems if they fail (i.e. opening and writing to a file handle or connecting to the database) will be followed by the ‘or die’ command with a suitable error message so that the script will terminate if the command fails (i.e. `$db->connect() || die “Unable to connect to db.”;`).
- End of Script – all scripts will end with a `1;` to let the web server know that the script has finished executing and exited O.K. A number other than 1 (i.e. 0) would signify that there was a problem, and no number would mean that the script had ended prematurely.

4.2.4 Perl Doc

Perl Doc is a utility that is included in the Perl Distribution that, like Java Doc, scans through Perl programs locating Pod comments and formats them into documentation like the man pages available on Unix (Siever 1999).

Unlike Java Doc, the Pod comments, which are located in amongst the code in the program files, are not formatted like regular comments and need to include formatting instructions with them in the code. This means that the Pod formatting instructions and comments can interfere with the legibility of code; therefore it has been decided not to make use Perl Doc, but to just use normal comments for explaining the code.

4.3 Database Implementation

The database implementation includes all the DBMS specific details and the SQL required to create the relations and indexes.

4.3.1 Data Types

The only feature of Postgress that makes the SQL any different to that which would be needed for another DBMS (i.e. Oracle) is that Postgress doesn't have a COUNTER data type. Therefore a Postgress SEQUENCE needs to be set up and the next value of that sequence be assigned as the default value for each of the fields that require a counter.

All the date fields will be implemented using the DATE data type which is the same as the DATE/TIME or TIMESTAMP data types available in other DMBS.

The fields containing text will be implemented in one of two different ways. Where the text is going to be of a known maximum length (i.e. a name field) the VARCHAR data type will be used and assigned a suitable maximum length. This is chosen in preference to a plain CHAR data type because the latter 'pads' out any empty space in the field with space characters – meaning that inserting the value "`Michael`" into a CHAR of 15 characters in length, will result in the database storing "`Michael `".

Where the text is likely to be of a unknown and highly variable length (i.e. a comments field) then the TEXT data type will be used. This data type allows text to be stored up to 1GB in size without the need to set a maximum length.

4.3.2 Tables

The tables are created in the database by using the PSQL program, which acts as an interface to the Postgress DBMS. When PSQL is run it requires the details of the database that it is connecting to – the name, host machine etc – and the user details – name and password – then is allows SQL to be entered and run against the database.

The following is an example of the SQL used to create one of the tables in the database. The SQL for the generation of all the tables, relationships and indexes in included in Appendix I. This SQL creates a sequence for the ID field of the table and then creates the table to contain the company information. The table consists of the ID field, which is a counter field of a maximum 4 characters (9999 records) and the Primary Key of the table, the NAME, CONTACT, and ADDRESS of the company, which are VARCHAR fields of various lengths

(50 and 70 characters) and COMMENTS, which is a text field. The NAME and ID fields cannot be left blank (null).

```
CREATE SEQUENCE co_id;

CREATE TABLE company (
 id       int4         NOT NULL DEFAULT nextval('co_id') PRIMARY KEY,
 name     varchar(50)  NOT NULL,
 contact  varchar(50)  ,
 address  varchar(70)  ,
 comments text         );
```

4.3.3 Relationships

Relationships in the database are implemented by adding FOREIGN KEY constraints to the appropriate fields of the tables. The following SQL shows how a Foreign Key can be added to the STUDENT table, making sure the value in the SUPERVISOR field is also in the USERNAME field of the STAFF table.

```
ALTER TABLE student
 ADD CONSTRAINT supervisor_FK FOREIGN KEY (supervisor)
 REFERENCES staff (username) MATCH FULL;
```

4.3.4 Indexes

In Postgress the DBMS implicitly creates the primary index on the Primary Key attribute of the table, therefore only secondary indexes can be explicitly defined. The following SQL defines two secondary indexes called ‘student_supervisor’ and ‘student_assessor’ and on the STUDENT table’s SUPERVISOR and ASSESSOR attributes, respectively.

```
CREATE INDEX student_supervisor ON student (‘supervisor’);
CREATE INDEX student_assessor ON student (‘assessor’);
```

4.4 Functionality

Because of the time restrictions on the project the Implementation is limited to producing a Prototype of the proposed system, offering a subset of the functionality planned for the final solution. Attention has been focussed on the implementation of the output from the system and the generation of the reports documented in the Design (Chapter 3.2) and not on the input of the data into the system (Chapter 3.3.3).

The functionality that has not been implemented is documented in the Future Work section (Chapter 6.3) of this report, which details all the work that still needs doing to deliver a fully working solution to the problem.

Testing

The purpose of testing in the RUP is to assess the quality of the product. This involves verifying the interaction between, and integration of, the different components, and that all the User's Requirements have been met (Kruchten 1998 p169).

This section of the report will cover the functional testing of the system checking that all the components are integrated and that the output on the reports is accurate and expected. Checking that the solution produced meets the User's Requirements will be covered in the Evaluation section of this report.

5.1 Test Plan

Testing the integrity and functionality of the system will involve running it with Test Data that has been specifically chosen to cover every possible situation, and checking that the system behaves as expected. The Test Data will also include some tests that are intended to try and break the system, testing how the system reacts to invalid inputs.

The Testing will be structured to check the input, editing and output of data. Table 2 (Chapter 5.3) details the functionality that needs testing. These functions are then tested using every possible combination of valid, invalid, missing and duplicated data.

The valid data test will involve inputting data that is expected by the system, this will be repeated to make sure that all the possible combinations of options are tested. The invalid data test will involve including non-alphanumeric characters into the data input, inputting dates in the wrong format and choosing illegal combinations of options, and testing how the system responds. The missing data test will meaning adding data that is valid into the input fields but leaving each one blank in turn and testing that the system behaves in an expected way. Finally, the duplicated data test involves adding the same data to the system twice. This can be in the form of a whole record (i.e. student) or just a single field of a record (i.e. student's name).

Since there are 78 functions to test, there are over a thousand of tests that need executing and documenting. Therefore Table 2 just documents whether the system passed each of the tests (**V**alid, **I**nvalid, **M**issing and **D**uplicated).

5.2 Test Data

Test Data is required to carryout the tests in a realistic manner. The Test Data documented in Appendix J will be used for all the input tests and then will be made available in the system (manually if necessary) for the editing and output tests. The Test Data consists of several instances of valid data for each of the entities stored by the system (i.e. student, project, etc).

5.3 Test Results

The following table contains all the functions that needed testing and the results of the tests. Details of the tests that failed (marked with a ×) are documented in Appendix K. Some of these tests could not be carried out at this time because the prototype hasn't implemented them yet – these are marks by "Not Implemented". Example screen shots of the reports generated by these tests are included in Appendix L.

Table 3: Functionality Test Results

Test	Functionality	User Group	V	I	M	D
Data Input						
T01	Add Company	Project Coordinator	✓	✓	×	✓
T02	Add Staff	Project Coordinator	✓	✓	×	×
T03	Add Student	Project Coordinator	✓	×	×	×
T04	Add Project	Project Coordinator	✓	✓	×	✓
T05	Add Area of Interest	Project Coordinator	✓	✓	×	✓
T06	Make Allocations	Project Coordinator	✓	✓	✓	✓
T07	Add Extension	Project Coordinator	Not Implemented			
T08	Add Project	Staff	✓	✓	×	✓
T09	Add Area of Interest	Staff	✓	✓	×	✓
T10	Add Assessment of Project	Staff	Not Implemented			
T11	Add Preference Form	Student	Not Implemented			
T12	Add Objectives Form	Student	Not Implemented			
T13	Add Legal Form	Student	Not Implemented			
T14	Add Interim Report	Student	Not Implemented			
T15	Add Demonstration	Student	Not Implemented			
T16	Add Final Report	Student	Not Implemented			
T17	Add Feedback	Student	✓	✓	×	✓
T18	Add URL	Student	✓	✓	✓	✓
T19	Add Email Address	Student	✓	✓	✓	✓
Data Editing						
T20	Edit Company Details	Project Coordinator	✓	✓	×	×
T21	Delete Company	Project Coordinator	×			
T22	Delete Staff	Project Coordinator	Not Implemented			
T23	Edit Student Details	Project Coordinator	✓	×	✓	✓
T24	Delete Student	Project Coordinator	Not Implemented			
T25	Delete Project	Project Coordinator	×			
T26	Delete Area of Interest	Project Coordinator	✓			
T27	Edit Extension	Project Coordinator	Not Implemented			
T28	Delete Extension	Project Coordinator	Not Implemented			

T29	Edit Schedule Deadlines	Project Coordinator	✓	✓	✓	✓
T30	Edit Allocations	Project Coordinator	✓	✓	✓	✓
T31	Edit Assessment	Staff	Not Implemented			
T32	Edit Email	Student	✓	✓	✓	✓
T33	Edit URL	Student	✓	✓	✓	✓
T34	Edit Preferences	Student	Not Implemented			
T35	Edit Objectives	Student	Not Implemented			
T36	Edit Legal Form	Student	Not Implemented			
T37	Edit Interim Report	Student	Not Implemented			
T38	Edit Demonstration	Student	Not Implemented			
T39	Edit Final Report	Student	Not Implemented			
Data Output						
T40	Schedule	Project Coordinator	✓			
T41	Projects	Project Coordinator	✓			
T42	Missed Deadlines	Project Coordinator	✓			
T43	Student List	Project Coordinator	✓			
T44	Student Details	Project Coordinator	✓			
T45	Staff List	Project Coordinator	✓			
T46	Staff Details	Project Coordinator	✓			
T47	Company List	Project Coordinator	✗			
T48	Company Details	Project Coordinator	✓			
T49	Preferences Form	Project Coordinator	Not Implemented			
T50	Objectives Form	Project Coordinator	Not Implemented			
T51	Legal Form	Project Coordinator	Not Implemented			
T52	Interim Report	Project Coordinator	Not Implemented			
T53	Demonstration	Project Coordinator	Not Implemented			
T54	Final Report	Project Coordinator	Not Implemented			
T55	Assessment	Project Coordinator	Not Implemented			
T56	Extensions	Project Coordinator	Not Implemented			
T57	Feedback	Project Coordinator	✓			
T58	Schedule	Staff	✓			
T59	Projects	Staff	✓			
T60	Staff Details	Staff	✓			
T61	Student Details	Staff	✓			
T62	Preferences Form	Staff	Not Implemented			
T63	Objectives Form	Staff	Not Implemented			
T64	Legal Form	Staff	Not Implemented			
T65	Interim Report	Staff	Not Implemented			
T66	Demonstration	Staff	Not Implemented			
T67	Final Report	Staff	Not Implemented			
T68	Assessment	Staff	Not Implemented			
T69	Schedule	Student	✓			
T70	Projects	Student	✓			
T71	Student Details	Student	✓			
T72	Preferences Form	Student	Not Implemented			
T73	Objectives Form	Student	Not Implemented			
T74	Legal Form	Student	Not Implemented			
T75	Interim Report	Student	Not Implemented			
T76	Demonstration	Student	Not Implemented			
T77	Final Report	Student	Not Implemented			
T78	Assessment	Student	Not Implemented			

Evaluation

The Evaluation section of this report fulfils the RUP requirement for the solution to be checked against the User's Requirements defined in Chapter 2. This Evaluation also includes an Evaluation of the Project against the Objectives and Minimum Requirements set out in Chapter 1, and documents the work that wasn't completed during the development of the prototype and other work that needs doing in the future.

6.1 Evaluation of the Solution

The Evaluation of the Solution begins with checking that each of the Functional Requirement that were defined in Chapter 2.2.1 have been met, and then checks that the Non-Functional Requirements defined in Chapter 2.2.2, which are more subjective, have also been met.

6.1.1 Functional Requirements

The solution provides Centralised Storage for the following information:

- ✓ **R01** - *Contact details for companies who supply external projects.*
- ✓ **R02** - *The schedule for the current year's deadlines.*
- ✓ **R03** - *Lists of the people who will be project supervisors.*
- ✓ **R04** - *Lists of the people who will be project assessors.*
- ✓ **R05** - *Lists of the students who are on MSc courses, including their preferred e-mail, their course, and the URL of their log website.*
- ✓ **R06** - *Lists of all the project suggestions (external and internal).*
- ✓ **R07** - *The MSc Project guidelines and handbook.*
- ✓ **R08** - *Copies of the Students' Preference Forms.*
- ✓ **R09** - *Allocations of the students, projects, supervisors and assessors.*
- × **R10** - *Copies of the Legal Forms, filled in by the external project students and the companies they are working for, and the various stages of completion (student, School, and company to sign).* It was not possible to meet this functionality since these are legal documents and a physical copy needs to be kept – although the system has been designed to record the fact that the Legal Document has been submitted.
- ✓ **R11** - *Copies of the Project Aims and Objectives.*
- ✓ **R12** - *Copies of the Interim Report and the comments made by the supervisor and assessor.*
- × **R13** - *Any issues/actions, from student feedback*. This requirement was not implemented – although the system does allow the students to give feedback anonymously, the issues and actions resulting from this are not recorded.

- ✓ **R14** - *Details of the Project Presentations, date and who attended etc*. This requirement has been implemented, but there is still a decision to be made over who should be responsible its submission – the students or the staff.
- ✓ **R15** - *Copies of the Final Reports, in Microsoft Word format and/or Post Script.*
- ✓ **R16** - *Copies of the comments and provisional marks given by the supervisors and assessors*.
- × **R17** - *Contact details for the students when they are out of School.* This requirement was not met by the current system, but it would be a simple task of adding it at a later date, by adding the required extra fields to the database and the Student Details report.
- × **R18** - *Issues and Resolutions of queries regarding the provisional marks*. This requirement was not met since these issues are best solved in a personal face-to-face manner; the system could be expanded to record the details of these queries, but currently doesn't offer any way of doing this.

The solution provides Decentralised Access to the following data that it holds:

- ✓ **R19** - *Publication of the lists of Project suggestions*.
- ✓ **R20** - *Publication of the deadline schedule.*
- ✓ **R21** - *Monitoring of who has/hasn't submitted the Preferences Forms.*
- ✓ **R22** - *All the information required for making the student-project-supervisor-assessor allocations. For example, project suggestions, lists of supervisors, students' preferences and students' exam performance*. This requirement has been met in part, but the student's exam performance isn't available through the system.
- ✓ **R23** - *Publication of the Project Allocations.*
- ✓ **R24** - *Publication of the MSc Project guidelines and handbook.* These are still published via the MSc Project Website, which this new system is now a part.
- ✓ **R25** - *Monitoring students who have/haven't submitted the legal Forms, and who was required to*.
- ✓ **R26** - *Monitoring students who have/haven't submitted the Project Scope*.
- ✓ **R27** - *Monitoring students who have/haven't submitted the Interim Report*.
- × **R28** - *Monitoring students who have/haven't submitted the Feedback Form.* This requirement was found to be redundant because the students giving feedback became an optional – and now anonymous – activity.
- ✓ **R29** - *Monitoring students who have/haven't submitted the Final Report, and details of any extensions*.

- × **R30** - *Monitoring staff that have/haven't returned the marked Reports (supervisors and assessors).* This requirement was found to be redundant by the fact that the system allows the staff to download their own copies of the reports.
- × **R31** - *Out-of-School contact addresses for the students, to distribute the provisional marks.* This requirement was not met by the current system, but it would be a simple task of adding it at a later date, by adding the required extra fields to the database and the Student Details report.
- ✓ **R32** - *Issues raised via the Feedback Form and their solutions.* The Project Coordinators being able to access the feedback given by the students from anyway meets this requirement, but the system doesn't record solutions to the feedback.
- × **R33** - *Issues raised about the provisional marks and their solutions*. The system does not record any issues raised by the student regarding their provisional marks as they are expected to bring these up via e-mail or in person with the Project Coordinators.
- × **R34** - *Report for the Exam Board's final meeting*. This requirement was non-essential and has not been met, but could be included by future development of the system.
- × **R35** - *Automated letter writing to the companies for external projects*. This requirement was non-essential and has not been met, but could be included by future development of the system.

The solution provides the following Additional Reports:

- × **R36** - *Lists of students on external projects*. This requirement has not been met by the system, but the students on external projects can be identified via the reports that are provided.
- ✓ **R37** - *The student information – detailing their progress, what was submitted and when, their project, supervisor, assessor and external company if appropriate. Ordered/viewed by student, supervisor, assessor, external projects, or whole year.*

The solution provides the following security:

- ✓ Students can only see their own personal details and can only submit documents in their own name.
- ✓ Staff can only see the details of the Students that they are the Supervisor or Assessor of, and can only submit project suggestions in their own names.
- ✓ Project Coordinators have access to all the data stored in the system and can submit projects in the name of any Member of Staff or external Company.

4.4.1 Non-Functional Requirements

The solution has met the following Non-Functional Requirements:

- ✓ **Ease of Use** – the majority of the Inputs into the system are exact replicas of the paper documents that would have to be completed by the Users anyway. Therefore there is no additional work for any of the users, they just have to fill in the documents on line instead of on paper, and the system has removed the need for manual records of 'who has done what'.
- ✓ **Reliability** – the system is available 24 hours a day, and will only be unavailable if the whole of the School's web system goes off-line.
- × **Performance** – the current response time for a requested report is acceptable at approximately 2 seconds, though this deteriorates slightly, to approx. 3 seconds, when the system is experiencing heavy usage. The performance could be improved by the use of more complex SQL queries – instead of the multiple simple ones implemented. This would remove the execution overhead from the Web Server and Network to the Database, which in turn would experience less 'hits', because there would be only one query instead of several.
- ✓ **Supportability** – the system is implemented in a manner that should make it easier to support. All the scripts are independent of each other and written in a consistent manner, shared functions and constant values are located in Modules so changes to any of these need only be made once.
- ✓ **Scalability** – the system is expandable because each function is independent of the others, so a new function could be added without the need to alter all the existing ones. The architecture that the system is implemented in will expand to meet a substantial increase in use before it would become unsuitable – the DBMS will cope with up to 500,000 queries per day, the average function makes 2 database queries, which, even with 100 users of the system (currently there would be approximately 60- 50 students, 8 staff and 2 coordinators) it would mean each user making 2,500 accesses of the system in a day to reach that limit.

6.2 Evaluation of the Project

The purpose of the Evaluation of the Project is to check that the Minimum Requirements and Project Objectives that were set out at the start of the project have been met. It will also assess the suitability of using the Rational Unified Process for this development project.

The Minimum Requirements that were set out in Chapter 1.4 were all met and exceeded as follows:

- ✓ *To produce an outline of the managements and monitoring activities for the MSc Project process and summarise the requirements*. This was achieved through the use

of the RUP Business Analysis and User Requirements stages in Chapters 2.1 and 2.2 respectively.

- ✓ *To create a prototype system which will run within the School's IT infrastructure (web-client→ Linux, Apache, Postgress).* This was achieved through the implementation of the Design documented in Chapter 3. The prototype created meets the following minimum functionality
- ✓ *Ability to record the "progress" of a student project by logging the flow of forms and reports from/to supervisor, assessor, project coordinator, and external supervisors (if appropriate), throughout the project cycle*. These 'forms and reports' now flow through the on-line system and are stored and monitored throughout the project cycle.
- ✓ *Ability to generate a report from the information stored in the system.* All the relevant Users can access all the information from the system in a number of different formats.
- ✓ *Ability to report the "progress" of individual students*. The students' progress through the project cycle can be monitored by themselves, their Supervisor and Assessor, and the Project Coordinator.

The objectives of the project as defined in Chapter 1.3 were also met as follows:

- ✓ *To understand the requirements of the MSc project process, for both Users and systems*.
- ✓ *To produce the design which meets these requirements and will fit into the School's Information Systems infrastructure.*
- ✓ *To produce a prototype of the designed system, to be reviewed by the Users.*
- ✓ *To evaluate the success of the system in terms of the usefulness of the prototype at meeting the Users requirements.*

Using the Rational Unified Process to approach this project worked out well. The structure of the methodology allowed it to be tailored to the specific requirements of the project, with stages dropped that weren't required. It also helped to make sure that all the appropriate steps had been completed in gathering User Requirements. Because of the in depth work, gathering and documenting the User Requirements, at the start of the project the later stages became more straight forward and easier to verify, by checking what was being done against the earlier stages' documentation.

6.3 Future Work

The Future Work section of this report covers the details of the functionality that wasn't implemented in the prototype and any changes or improvements to the functionality that was implemented.

6.3.1 Changes

The following changes to the functionality of the prototype have been identified as being required. Some of these are because of changes to the User's Requirements and others are due to identified advancements in the Implementation technique.

- The method used in the implementation of the database queries was to keep the SQL as simple as possible. In some cases this resulted in several queries being used where a single, more complex query would have achieved the same results. Reducing the number of queries required to complete a task reduces the number of 'hits' the database experiences and also reduces the network traffic of the query and results going to and from the database. An example of where this would be useful is in the Student Details report where there is one query to get all the details from the student table of the database and then separate queries to get the corresponding project, supervisor and assessor details. This would be more efficient if there was a single query using INNER JOINS to link the details from these other tables. This particular example could be implemented using the following SQL – where <?> is substituted for the ID of the student whose details are being displayed:

```
SELECT student.*, project.title, staff.name AS supervisor,
staff2.name AS assessor FROM student, staff, staff AS staff2,
project WHERE staff.username=student.supervisor AND
staff2.username=student.assessor AND project.id=student.project
AND student.username=<?>;
```

- The Demonstration submission needs moving to be carried out by the Supervisor and not the Student. It also needs renaming to "Progress Meeting" to keep the MSc Project Process inline with the Undergraduate Project Process. This can be achieved by removing the "Demonstration" option from the Students menu, and adding a "Progress Meeting" option to the Staff Menu, also the script which implements the function needs moving into the directory containing all the scripts for the Staff's functions, finally cosmetic changes need doing to the scripts and the database to change references to Demonstration to be Progress Meeting.
- There are a number of 'bugs' that were identified by the Testing (Chapter 5). These are mainly to do with the validation of data inputs into the system, which haven't been fully implemented. There are also some issues to address with the reporting of errors – all the technical error messages need replacing by something that the users can understand.

6.3.2 Still to Implement

The functions that are still to be implemented are mainly the submitting of the forms from the students. These forms have been created so the background code needs writing to process the information returned from these forms, validate and save it to the database. The following submissions still need implementing:

- Project Preferences Form by the students.
- Aims and Objectives Form by the students.
- Legal Form ~ there are issues still to be decided over what information should be recorded and by whom.
- Interim Report by the students.
- Final Report by the student.
- Extensions to the deadlines by the Project Coordinator.
- Assessments by the Staff.

The functions to be able to view this information once it has been submitted also need implementing. These will be implemented in the 'common' directory of the system, because all Users will have access to them but the links to them will be protected by the Users logging onto the system. The pages will be accessed by links on the pages containing the students 'progress', but if the pages are access directly they will load blank because there will be no identifier for the student, whose form if to be displayed, passed into the script via CGI.

References

Apache Org – 2001 – Apache Organisation Website – accessed on 29 November 2001 - http://java.apache.org/faq/fom-serve/cache/237.html

Date, CJ – 1995 – An Introduction to Database Systems (6th Edition) – Addison Wesley.

Elmasri, Ramez & Navathe, Shamkant – 2000 – Fundamentals of Database Systems (3rd Edition) – Addison Wesley.

Gough, Tom – 2001 – IN31 Lecture Notes on People Centred Information Systems Development – School of Computing, University of Leeds.

Kruchten, Philippe – 1998 – The Rational Unified Process (An Introduction) – Addison Wesley.

Matravers, Julika (Dr) – 1998 – SI13 Human Computer Interaction ~ Lecture Notes – School of Computing, University of Leeds.

Mott, Peter – 2001 – DB31 ~ Advanced Databases ~ Optimisation Slides – School of Computing, University of Leeds.

Perdue, Tim – 2000 – MySQL and Postgress Compared – accessed 28 November 2001 – http://www.phpbuilder.com/columns/tim20000705.php3 - published 5 July 2000 http://www.phpbuilder.com/columns/tim20001112.php3 - published 12 November 2000

Pisoni, A – 1999 – Popular Perl Complaints and Myths– accessed on 29 November 2001 – http://perl.apache.org/perl_myth.html – published 5 August 1999

Rajagopal, R – 2000 – Comparing Apache and Internet Information Server – accessed on 29 November 2001 – http://www.networkedcomputing.com/unixworld/1124/1124uw.html – published 4 December 2000

Siever, Ellen, Spainhour, Stephen & Patwardhan, Nathan – 1999 – Perl in a Nutshell – O'Reilly.

Appendix A: Reflection on the Project Experience

This project has given me a great insight to and a better understanding of the stages that are involved in undertaking a larger development project.

All the Personal Learning Objectives that I set out at the beginning of the project have been met and I have benefited from the interaction with real users during the Requirements Analysis stage – something that I hadn't fully experienced in any module or coursework done while at university.

I feel I have learned a lot from the completion of the whole development cycle, and not just the Implementation stages. I feel that I am stronger in the technical aspects of Computer Studies, and found the coding activities fairly straightforward – after learning the new technologies – but the analysis and documentation were a lot more challenging. Finding unambiguous ways of recording the user's requirements accurately and drawing up the Designs for the database and scripts before going anywhere near the coding was very difficult.

Despite the supposed straightforwardness of the Implementation I was shock by how time consuming it actually was. I had scheduled 4 weeks for the Implementation, which I thought would be ample time, but in reality turned out to be a significant underestimation. In the end the Implementation took 6 weeks of intensive work and I still didn't complete all the functionality I had intended.

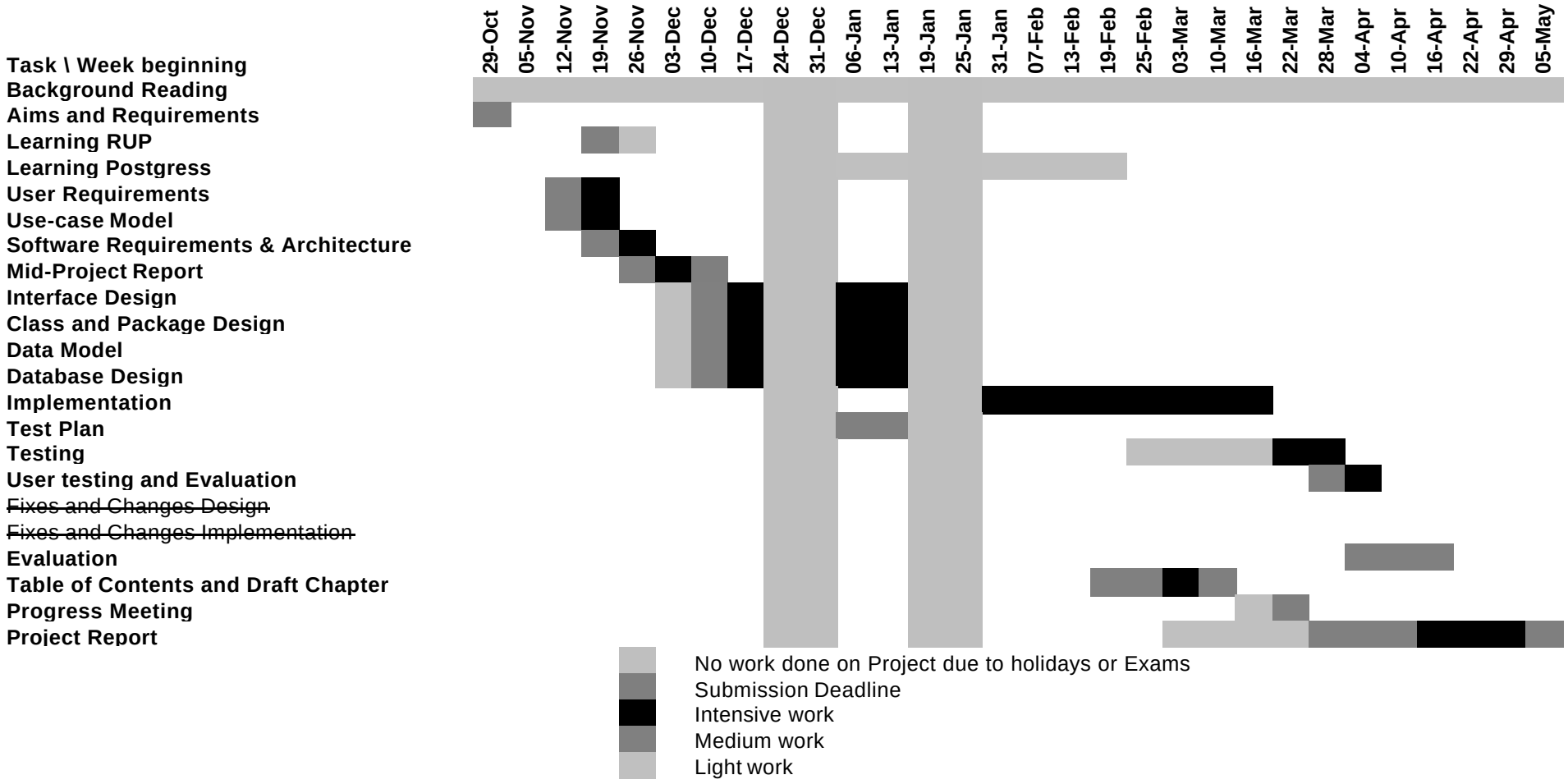
As I was doing the implementation I kept finding better, and tidier, ways of implementing similar functions. This became frustrating, as I had to go back and change code that I had already written to keep it all consistent. I think that this will always be a problem, but will become less significant as I develop a more in-depth knowledge of the programming languages used, which will come with experience.

I've also learned the importance of thorough and planned testing. Although the implementation wasn't of the whole system, and I had carried out unit testing as the different sections of the system were being implemented, the number of different tests that I actually came up with (nearly 1000 individual tests) and the quantity of errors that were identified by them was quite alarming. I'm glad that I had scheduled 3 weeks for a second cycle of design

and implementation to correct any errors identified by the testing – even though this stage wasn't actually possible due to earlier stages overrunning.

Overall I feel that the project was very successful and very enjoyable. I'm glad that I have been able to motivate myself to spend time on it throughout the academic year, even when the deadline seemed so long way. This time management and motivation is the most important thing that I think I have learned from the experience.

Appendix B: Schedule in Practice


Task \ Week beginning
Background Reading
Aims and Requirements
Learning RUP
Learning Postgress
User Requirements
Use-case Model
Software Requirements & Architecture
Mid-Project Report
Interface Design
Class and Package Design
Data Model
Database Design
Implementation
Test Plan
Testing
User testing and Evaluation
Fixes and Changes Design
Fixes and Changes Implementation
Evaluation
Table of Contents and Draft Chapter
Progress Meeting
Project Report
29-Oct
05-Nov
12-Nov
19-Nov
26-Nov
03-Dec
10-Dec
17-Dec
24-Dec
31-Dec
06-Jan
13-Jan
19-Jan
25-Jan
31-Jan
07-Feb
13-Feb
19-Feb
25-Feb
03-Mar
10-Mar
16-Mar
22-Mar
28-Mar
04-Apr
10-Apr
16-Apr
22-Apr
29-Apr
05-May
No work done on Project due to holidays or Exams
Submission Deadline
Intensive work
Medium work
Light work

Appendix C: Use Case Model

Use Case 1: Supervisor and Assessor Lists

At the start of the academic year, the list of people who will be the project supervisors and assessors is drawn up.

Actors: Senior members of the School of Computing.

Data requirements: The lists of lecturers in the School that are connected with the MSc course.

Outputs: The lists of the supervisors and assessors for the year.

Access: Students shouldn't have access to these lists.

Prerequisites: None

Follow up actions: Project Allocations (6).

Use Case 2: Deciding the Schedule

At the start of the academic year, the timetable of the deadlines for the year's projects is drawn up.

Actors: MSc Course Director and the Project Coordinators.

Data Requirements: a calendar of the academic year.

Outputs: the schedule.

Access: Students should have access to the published schedule.

Prerequisites: None.

Follow up actions: Everything.

Use Case 3: Acquiring External Projects

In December, companies are contacted to invite them to suggest projects to be done by the MSc students within their organisation.

Actors: Project Coordinators.

Data requirements: The contact details of the companies.

Outputs: Correspondence to the companies (automated letter).

Prerequisites: None

Follow up actions: Publishing the project suggestions (4).

Use Case 4: Publication of the Project Suggestions

In January, the list of all the suggested projects is published for the students to be able to study.

Actors: Project Coordinators.

Data Requirements: The lists of suggestions by the project supervisors and the external companies.
Outputs: The combined lists of suggestions.
Access: Students should be able to read these lists, but not edit them.
Prerequisites: The selection of supervisors (1), the acquisition of external project suggestions (3).
Following actions: The submission Student's Preference Form (5).

Use Case 5: Student Preference Forms
In February, preference forms are submitted by the students outlining the projects that they would like to undertake.
Actors: project administrator and the students.
Data Requirements: The lists of the project suggestions.
Outputs: List of those who haven't submitted the forms on time.
Access: Students should be able to see their own, but no one else's, form - these could be submitted on-line.
Prerequisites: The publication of project suggestions (4).
Follow up actions: The project allocations (6).

Use Case 6: Project Allocations
In February each student is allocated a project to work on, and a supervisor and assessor.
Actors: The Project Coordinators.
Data Requirements: The student's preference forms, their exam results, the lists of projects, supervisors and assessors.
Outputs: List of the project allocations.
Access: Students should be able to see their own allocations, but no one else's, and they shouldn't be able to edit them.
Prerequisites: The students returning the preference forms (5), and the lists of assessors and supervisors being complete (1).
Following actions: The legal forms (7) and the project scope (9).

Use Case 7: Legal Forms
In February the companies taking on students to do external projects are sent a legal form to sign and return.
Actors: The Project Coordinators and administrator.
Data Requirements: The lists of students that are undertaking external projects, the contact details of the external companies, and the legal document.

Outputs: None.
Access: Students should be able see their own legal forms.
Prerequisites: The project allocations (6).
Following actions: The submission of the project scope (9), or the reallocation of students to projects (8).

Use Case 8: Reallocation of Project

If a company does not agree to the legal form then the student will need to be allocated a new project.
Actors: The Project Coordinators or MSc Course Director.
Data Requirements: The student preference forms, exam results and the lists of supervisors, projects and assessors.
Outputs: List of project allocations.
Prerequisites: The failure to complete the legal form (7).
Following actions: Either the project scope (9) or the legal form (7).

Use Case 9: Project Scope

In March a project scope is submitted outlining the aims and objectives of the project.
Actors: The project coordinators, administrator and students.
Data Requirements: None.
Outputs: Lists of those who haven't submitted the scope on time.
Access: Students should only be able to submit and see their own work.
Prerequisites: The project allocations (6) and the legal form (7).
Following actions: The submission of the interim report.

Use Case 10: Interim Report

After the Easter break, April/May, the interim report is submitted.
Actors: The project coordinators and administrator.
Data Requirements: None.
Outputs: Lists of those who haven't submitted the report on time.
Access: Students should only be able to submit and see their own report.
Prerequisites: The submission of the project scope (9).
Following actions: Interim report feedback (11).

Use Case 11: Interim Report Feedback

In May the supervisors and assessor read the interim report and give the students feedback.
Actors: The project supervisors and assessors.

Data Requirements: The interim report.
Outputs: Comments by the supervisors and assessors.
Access: Students should only be able to see their own feedback.
Prerequisites: The submission of the interim report (10).
Following actions: The submission of the student feedback form (12).

Use Case 12: Student Feedback

During the summer students are given the chance to comment on their progress and raise any issues about their projects.
Actors: The project coordinators, administrator and the students.
Data Requirements: None
Outputs: Completed student feedback form.
Access: Students should only be able to submit and see their own comments.
Prerequisites: Project Allocations (6)
Following Actions: Feedback Follow Up (13)

Use Case 13: Feedback Follow Up

After the students return the feedback forms there may need to be action to sort out any issues raised. These could be anything, so the system will have to be flexible enough to be able to cope with this vague use case.
Actors: Anyone.
Data Requirements: Student Feedback form (12).
Outputs: Documentation of what the issues and resolutions were.
Access: Students should only be able to see their own form.
Prerequisites: Students submitting the feedback form - raising problems.
Following Actions: Anything.

Use Case 14: Project Demonstration

In early September, before the final reports are submitted, the students are required to demonstrate their projects to their supervisor and assessor. Stand-ins can be arranged for either of these in special circumstances.
Actors: Students, project supervisors and assessors.
Data Requirements: times to be arranged between the students, supervisors and assessors.
Outputs: List of those who haven't completed this action on time.
Access: Students have to record this information themselves, and should not have access to anyone else's.
Prerequisites: Interim Report (10).

Following Actions: Final Report (15).

Use Case 15: Final Report

In September, after the project demonstration has taken place, the students submit two copies of the final draft of their project report, which are sent to the supervisor and assessor to be marked.

Actors: Student, assessor, supervisor, administrator.

Data Requirements: Project Allocations.

Outputs: Lists of those who haven't submitted the reports on time.

Access: Students should only be able to submit and see their own report.

Prerequisites: Project Demonstration (14)

Following Actions: Provisional Marks (16)

Use Case 16: Provisional Marks

In September/October the mark and comment sheets are returned by the supervisors and assessors, with the reports, and the provisional marks are sent to the students at their Out-of-School address.

Actors: Supervisors, Assessors, Administrators and Coordinators.

Data Requirements: The returned marks and the students' Out-of-School address.

Outputs: Provisional marks.

Access: Students should only be able to see their own marks.

Prerequisites: The submission of the final report (15).

Following Actions: Resolution of any issues with the provisional marks (17), and the final exam board meeting (18).

Use Case 17: Issues with provisional marks

Student can raise queries concerning their provisional marks, which will be resolved by the MSc Course Director.

Actors: Student, MSc Course Director.

Data Requirements: Provisional marks and the query made by the student.

Outputs: Documentation of the query and the solution.

Access: Students should only be able to submit and see their own documentation and solution.

Prerequisites: Provisional marks being sent out (16).

Following Actions: Final exam board meeting (18).

Use Case 18: Final Exam Board Meeting

At the end of the academic year the University Exam Board meet to finalise the marks for all the assessments.
Actors: MSc Course Director and the University Exam Board.
Data Requirements: Provisional marks for the projects and a summary of any issues and resolutions regarding these marks.
Outputs: Final grades for the project.
Prerequisites: Provisional marks (16) and resolution of any queries (17).
Following Actions: Cycle starts again (1).

Use Case 19: Progress Report/Query

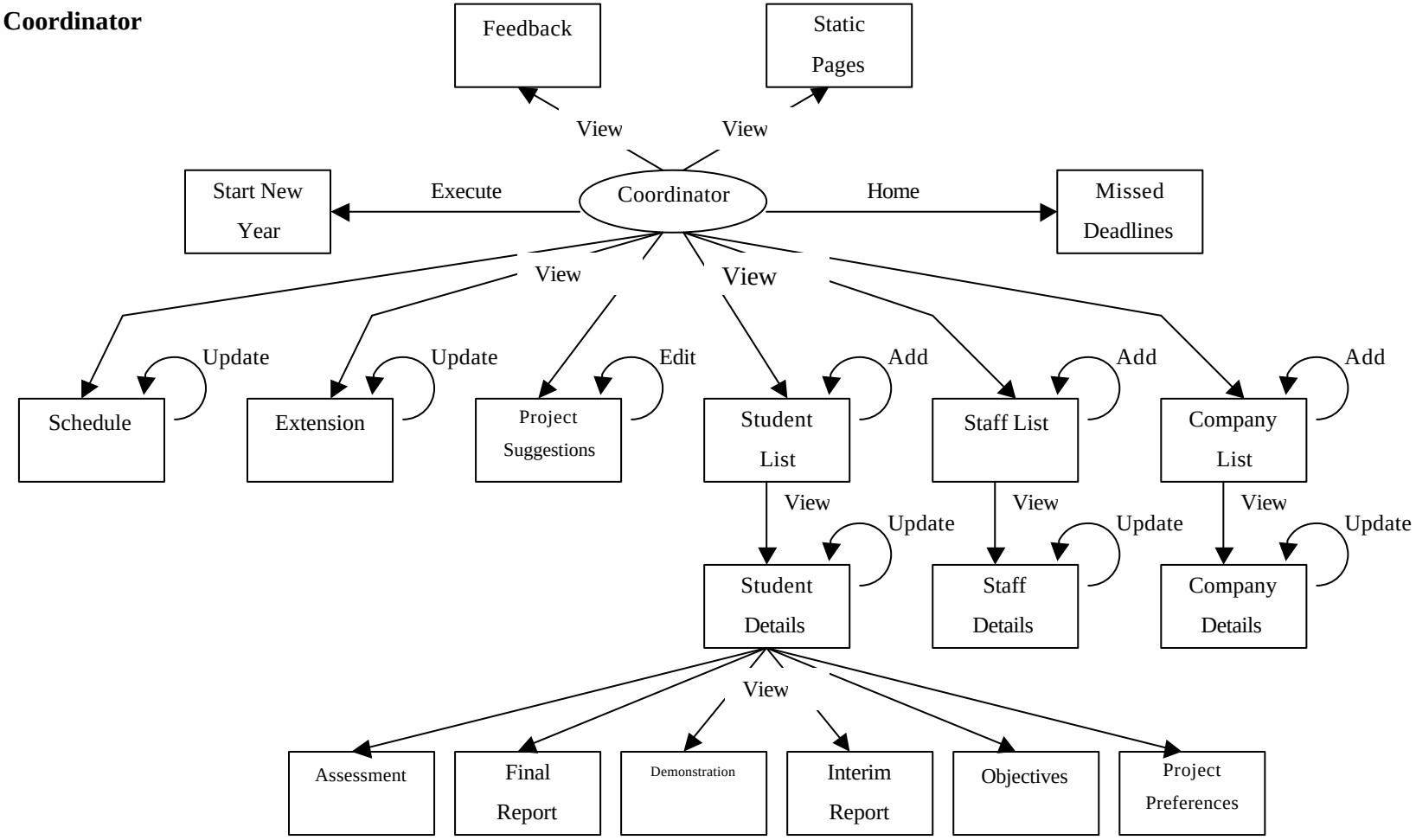
At any point during the process, the MSc Course Director, Project Coordinator, Project Supervisor, or Project Assessor may want to check on the progress of any one, or more, of the students. They may want this data to be presented by supervisor, assessor, student or project (internal/external) order/format. For example they may want to see the list of students who are doing external projects.
Actors: MSc Course Director, Project Coordinator, Project Supervisor or Project Assessor
Data Requirements: Everything in the system at that time.
Outputs: The results of the query.
Access: Students should only be able to see their own progress.
Prerequisites: None.
Following Actions: None.

Use Case 20: Extensions

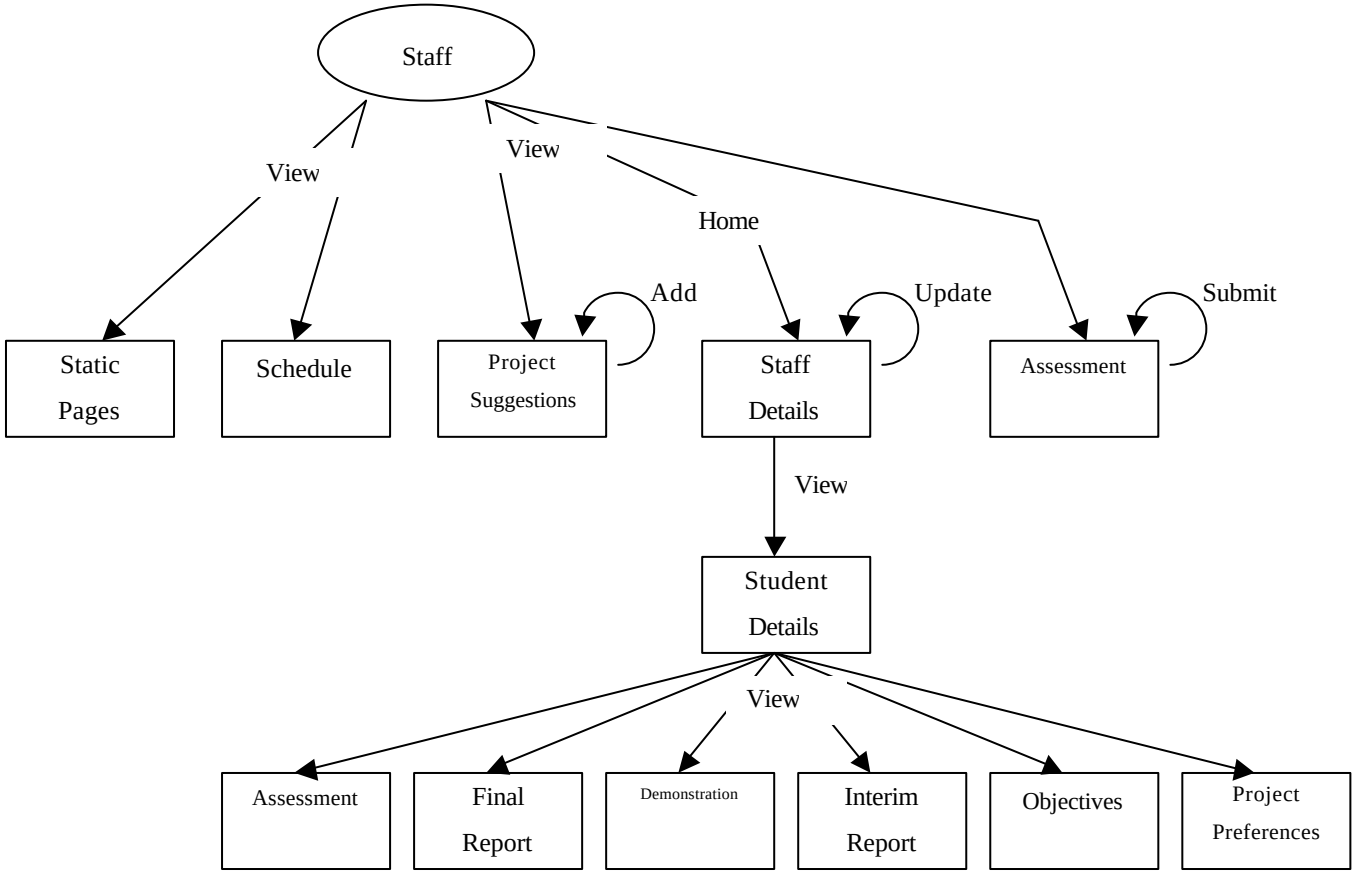
Sometimes it is necessary for students to be granted an extension on one of the deadlines. These would have to be arranged with the project coordinators or course director by the student.
Actors: MSc Course Director, Project Coordinator and Student.
Data Requirements: None.
Outputs: None.
Access: Student should be able to see their own extension, but not alter them.
Prerequisites: None.
Following Actions: Submission.

Appendix D: Functionality Diagrams

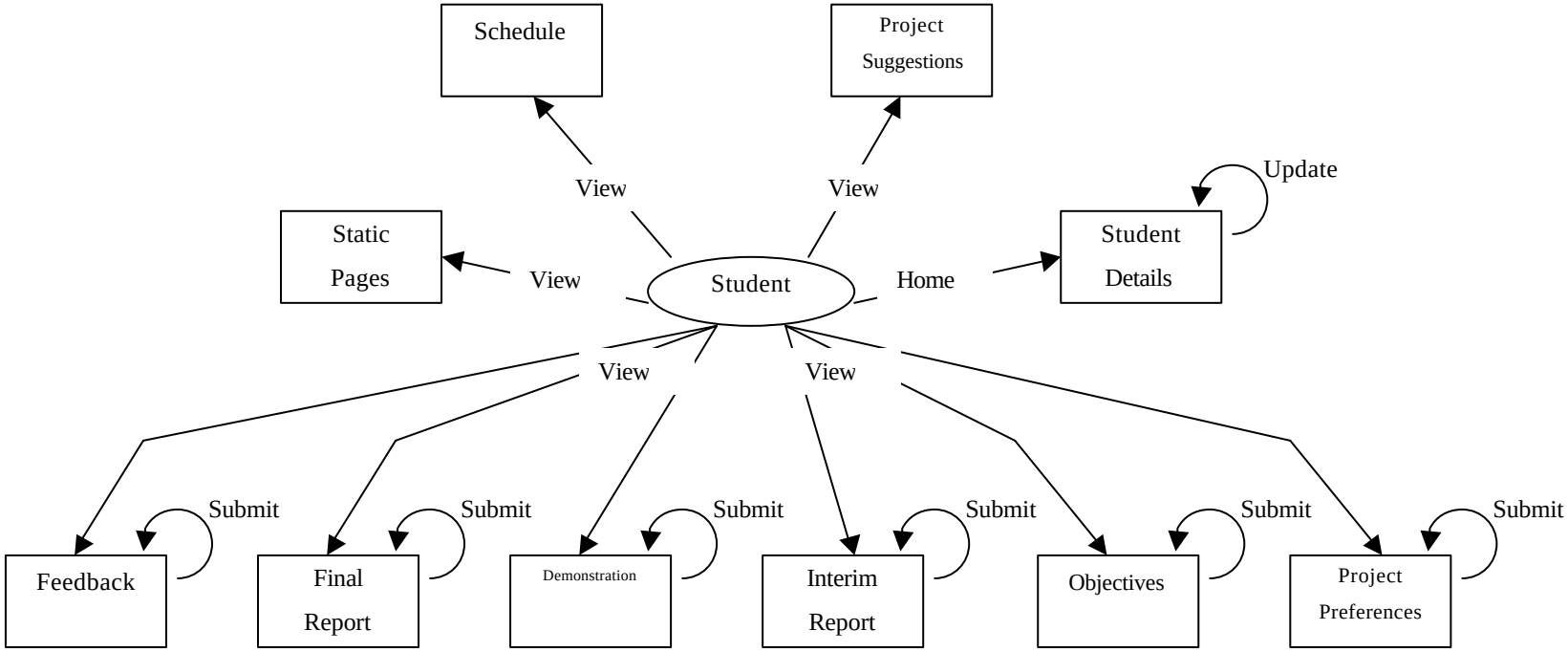
Project Coordinator



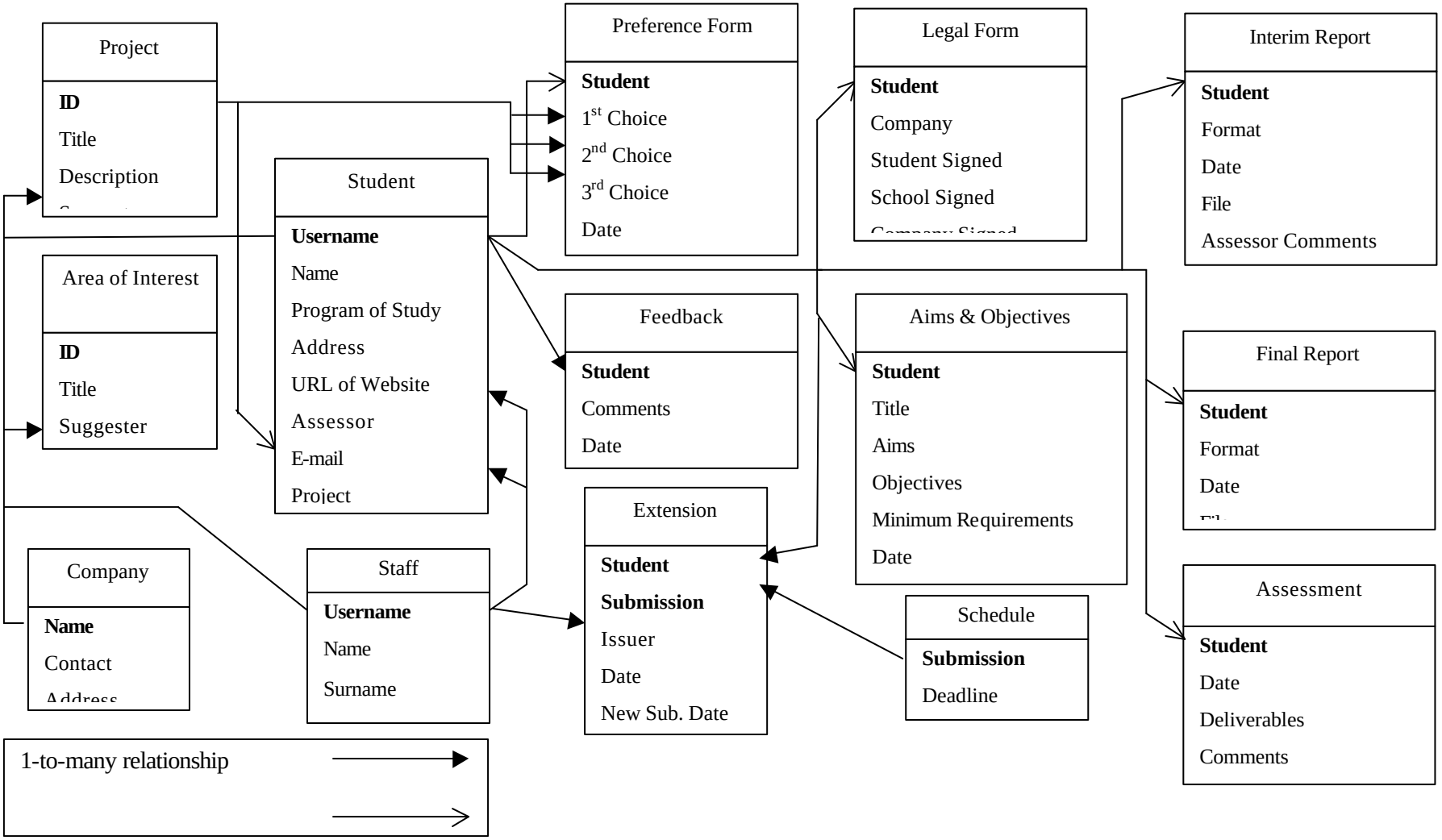
Supervisors and Assessors



Students


Schedule
Project Suggestions
View
View
Update
Static Pages
View
Student
Home
Student Details
View
View
Submit
Submit
Submit
Submit
Submit
Submit
Feedback
Final Report
Demonstration
Interim Report
Objectives
Project Preferences

Appendix E: Entity Relationship Model



Appendix F: Documents

Appendix G: Report Designs

Student Details

MSc Project Website

Student Details

Name	Joe Bloggs
Program	Information Systems
Email	Joeb@hotmail.com
URL	www.joebloggs.co.uk
Supervisor	Jane Hayfield
Assessor	Paul McKenna
Project	Something nice and easy

Submissions

Preference Form	Objectives	Legal Form	Interim Report	Demonstration	Final Report	Assessment
✓	✘	N/a	✘	✘	✘	✘

MSc Project Website

Staff List

Member of Staff	Supervises	Assesses	Projects
Paul McKenna	2	3	6
Jimmy Saville	6	2	2
Paul Dobson	0	2	0

Add New Staff

Username	Name

Staff List

MSc Project Website

Staff Details

Paul McKenna

Student	Preference Form	Objectives	Legal Form	Interim Report	Demonstration	Final Report	Assessment
Supervises							
Joe Bloggs	✓	✗	N/a	✗	✗	✗	✗
Assesses							
Fed Durst	✓	✓	✗	✗	✗	✗	✗

Projects

Staff Details

Company List

MSc Project Website

Company List

Company	Projects
British Telecom	6
Leeds United AFC	2
Some other company	0

Add New Company

Name

MSc Project Website

Company Details

Company	Leeds United AFC
Contact	David O'Leary
Address	Elland Road, Leeds 11
Comments	These boys are the greatest footy team in the whole wide world

SAVE

CANCEL

DELETE

Company Details

Missed Deadlines

MSc Project Website

Missed Deadlines

The **Project Preference** form was due in on **12/04/2002** and the following people have not yet submitted it.

Student	Extension
Joe Bloggs	No

MSc Project Website

Project Schedule

Submission	Deadline
Preferences Form	12/05/2002
Objectives	12/06/2002
Legal Form	15/06/2002

Project Schedule

MSc Project Website

Project Suggestions

Projects

1	**Some project title**
Some project description would go here and have all the waffle describing what work needs doing and all the interesting things like that blah blah blah.	
	Paul McKenna

2	**Some project title**
Some project description would go here and have all the waffle describing what work needs doing and all the interesting things like that blah blah blah.	
EXTERNAL	Paul McKenna

Project Suggestions

MSc Project Website

Student Feedback

31/06/2002
Some anonymous comments that have been submitted by the students about what they think of the project process and any improvement that they want to suggest.

29/04/2002
Some anonymous comments that have been submitted by the students about what they think of the project process and any improvement that they want to suggest.

Student Feedback

Appendix H: Example Script

```
#!/usr/bin/perl -wT

# Perl script to display all the projects
# Author: Ian Sugden
# History: 11 April 2002 ~ created

package admin;

# Import my modules
use lib qw(/home/csunix/isyis/WWW/cgi-bin/);
use common::Functions;

# Import other modules
use CGI::Carp ('fatalsToBrowser');
use DBI;

# Variables
$db        = DBI->connect($common::DBI_CONNECTION,
                          $common::DBI_USER,
                          $common::DBI_PSWD,
                          $common::DBI_OPTIONS) || die DBI-errstr;
$db_query  = "";  # the handle for the nested db queries
$db2       = DBI->connect($common::DBI_CONNECTION,
                          $common::DBI_USER,
                          $common::DBI_PSWD,
                          $common::DBI_OPTIONS) || die DBI-errstr;
$db_query2 = ""; # the handle for db queries
@data      = (); # for the results from the db
$title     = "Project Suggestions"; # the title of the page
$areaID    = ""; # the id of the area of interest
$areaTitle = ""; # the title of the area of interest
$areaSug   = ""; # the member of staff who suggested the area of interest
$projID    = ""; # the id of the project
$projTitle = ""; # the title of the project
$projDesc  = ""; # the description of the project
$projExt   = ""; # Boolean flag - true for external projects
$projSug   = ""; # the suggestor of the project (company or staff)

# start html
&common::StartHTML($title);
print "<h2>Projects</h2>";

# get all the projects from the db
$db_query = $db->prepare("SELECT * FROM $common::PROJECT_TABLE");
$db_query->execute() || die DBI->errstr;
while (@data = $db_query->fetchrow_array())
{
       $projID    = $data[0];
```

```
        $projTitle = $data[1];
        $projDesc  = $data[2];
        $projExt   = $data[3];
        $projSug   = $data[4];

        print <<END;
$common::TABLE<tr><th colspan=\"2\">$projID: $projTitle</th></tr>
<tr><td colspan=\"2\">$projDesc</td></tr>
<tr><td width=\"20%\">EXTERNAL</td>
END
        # get the company's name from the db
        $db2_query = $db2->prepare("SELECT $common::COMPANY_NAME FROM
$common::COMPANY_TABLE WHERE $common::COMPANY_ID='$projSug'");
        $db2_query->execute() || die DBI->errstr;
        @data = $db2_query->fetchrow_array();
        print "<td width=\"80%\" align=\"right\">$data[0]</td></tr>";
}

# end the html
print "</body></html>\n\n";

# end of script
1;
```

Appendix I: SQL

```
CREATE TABLE schedule (
 submission  varchar(20) NOT NULL PRIMARY KEY,
 deadline    date        NOT NULL);

CREATE SEQUENCE company_id_seq;

CREATE TABLE company (
 id       int4         NOT NULL DEFAULT nextval('company_id_seq') PRIMARY KEY,
 name     varchar(50)  NOT NULL,
 contact  varchar(50)  ,
 address  varchar(70)  ,
 comments text         );

CREATE TABLE student (
 username   varchar(10) NOT NULL PRIMARY KEY,
 name       varchar(50) NOT NULL,
 program    varchar(20) NOT NULL,
 email      varchar(50) ,
 url        varchar(70) ,
 project    int4        ,
 supervisor varchar(10) ,
 assessor   varchar(10) );

CREATE SEQUENCE project_id_seq;

CREATE TABLE project (
 id          int4        NOT NULL DEFAULT nextval('project_id_seq') PRIMARY KEY,
 title       varchar(50) NOT NULL,
 description text        NOT NULL,
 external    boolean     NOT NULL,
 suggestor   varchar(10) NOT NULL );

CREATE TABLE area (
 id        int4        NOT NULL DEFAULT nextval('project_id_seq') PRIMARY KEY,
 title     varchar(99) NOT NULL,
 suggestor varchar(10) NOT NULL );

CREATE TABLE staff (
 username  varchar(10) NOT NULL PRIMARY KEY,
 name      varchar(50) NOT NULL );

CREATE SEQUENCE extension_id_seq;

CREATE TABLE extension (
 id         int4 NOT NULL DEFAULT nextval('extension_id_seq') PRIMARY KEY,
 student    varchar(10) NOT NULL,
 submission varchar(20) NOT NULL,
 deadline   date        NOT NULL,
```

```
 issuer     varchar(10) NOT NULL,
 date       date        NOT NULL,
 reason     text        NOT NULL );

CREATE SEQUENCE feedback_id_seq;

CREATE TABLE feedback (
 id       int4        NOT NULL DEFAULT nextval('feedback_id_Seq') PRIMARY KEY,
 date     date        NOT NULL,
 comments text        NOT NULL);

CREATE TABLE preferences (
 student        varchar(10) NOT NULL PRIMARY KEY,
 first_project  int4        NOT NULL,
 first_reason   text        NOT NULL,
 second_project int4        NOT NULL,
 second_reason  text        NOT NULL,
 area           int4        NOT NULL,
 reason         text        NOT NULL,
 date           date        NOT NULL,
 comments       text        );

CREATE TABLE objectives (
 student    varchar(10) NOT NULL PRIMARY KEY,
 title      text        NOT NULL,
 aims       text        NOT NULL,
 objectives text        NOT NULL,
 min_req    text        NOT NULL,
 date       date        NOT NULL,
 comments   text        );

CREATE TABLE legal_form (
 student varchar(10) NOT NULL PRIMARY KEY,
 date    date        NOT NULL,
 company date        );

CREATE TABLE interim_report (
 student  varchar(10) NOT NULL PRIMARY KEY,
 format   varchar(10) NOT NULL,
 title    varchar(70) NOT NULL,
 date     date        NOT NULL,
 file     varchar(50) NOT NULL,
 comments text        );

CREATE TABLE demonstration (
 student    varchar(10) NOT NULL PRIMARY KEY,
 supervisor varchar(10) NOT NULL,
 assessor   varchar(10) NOT NULL,
 time_date  date        NOT NULL);

CREATE TABLE final_report (
```

```
 student varchar(10) NOT NULL PRIMARY KEY,
 title   varchar(70) NOT NULL,
 format  varchar(10) NOT NULL,
 date    date        NOT NULL,
 file    varchar(50) NOT NULL);

CREATE TABLE assessment (
 student       varchar(10) NOT NULL PRIMARY KEY,
 date          date        NOT NULL,
 deliverables  text        NOT NULL,
 comments      text        NOT NULL,
 understanding varchar(20) NOT NULL,
 solution      varchar(20) NOT NULL,
 evaluation    varchar(20) NOT NULL,
 writeup       varchar(20) NOT NULL,
 reflection    varchar(20) NOT NULL);

ALTER TABLE student
 ADD CONSTRAINT supervisor_FK FOREIGN KEY (supervisor)
 REFERENCES staff(username) MATCH FULL;

ALTER TABLE student
 ADD CONSTRAINT assessor_FK FOREIGN KEY (assessor)
 REFERENCES staff(username) MATCH FULL;

ALTER TABLE extension
 ADD CONSTRAINT extension_student_fk FOREIGN KEY (student)
 REFERENCES student(username);

ALTER TABLE extension
 ADD CONSTRAINT issuer_fk FOREIGN KEY (issuer)
 REFERENCES staff(username);

ALTER TABLE extension
 ADD CONSTRAINT extension_submission_fk FOREIGN KEY (submission)
 REFERENCES schedule(submission);

ALTER TABLE preferences
 ADD CONSTRAINT pref_student_fk FOREIGN KEY (student)
 REFERENCES student(username);

ALTER TABLE preferences
 ADD CONSTRAINT first_project_fk FOREIGN KEY (first_project)
 REFERENCES project(id);

ALTER TABLE preferences
 ADD CONSTRAINT second_project_fk FOREIGN KEY (second_project)
 REFERENCES project(id);

ALTER TABLE preferences
 ADD CONSTRAINT area_fk FOREIGN KEY (area)
```

```
 REFERENCES area(id);

ALTER TABLE objectives
 ADD CONSTRAINT objectives_student_fk FOREIGN KEY (student)
 REFERENCES student(username);

ALTER TABLE legal_form
 ADD CONSTRAINT legal_student_fk FOREIGN KEY (student)
 REFERENCES student(username);

ALTER TABLE interim_report
 ADD CONSTRAINT interim_student_fk FOREIGN KEY (student)
 REFERENCES student(username);

ALTER TABLE demonstration
 ADD CONSTRAINT demo_student_fk FOREIGN KEY (student)
 REFERENCES student(username);

ALTER TABLE final_report
 ADD CONSTRAINT final_student_fk FOREIGN KEY (student)
 REFERENCES student(username);

ALTER TABLE assessment
 ADD CONSTRAINT assessment_student_fk FOREIGN KEY (student)
 REFERENCES student(username);

CREATE INDEX student_supervisor ON student (‘superviosr’);
CREATE INDEX student_assessor ON student (‘assessor);
CREATE INDEX project_suggestor ON project (‘suggestor’);
CREATE INDEX area_suggestor ON area (‘suggestor’);
```

Appendix J: Test Data

Project Schedule	
Submission	**Deadline**
Preference	21/02/2002
Legal Form	15/03/2002
Objectives	22/03/2002
Interim Report	01/05/2002
Demonstration	06/09/2002
Final Report	06/09/2002

Companies			
Company Name	**Contact**	**Address**	**Comments**
McDonalds	Ronald MacDonald	St Johns Centre, Leeds, West Yorks.	Projects based around EPOS, and MIS systems.
Halifax	Irene Fisher	Meadow Lane, Holbeck, Leeds	Internet Banking developments.
Equifax	Andy Copley	St Georges Street, Bradford, W Yorks.	Consumer information and internet applications.
Leeds United AFC	David O'Leary	Elland Road, Leeds	AI research into player tracking during a game.
AIWA	Paul Bradley	Livingham Street, Halifax, W. Yorks.	Development of small application in hi-fi's.

Students							
Username	**Name**	**Program**	**Email**	**URL**	**Project**	**Supervisor**	**Assessor**
bill	Bill Clinton	IS	bill@bill.com	None	3	Dave	None
ian	Ian Sugden	DMS	ian@ian.com	None	None	Jane	Gillian
darren	Darren Mease	IS	None	None	1	Wendy	Jane
paul	Paul Smith	DMS	None	www.paul.com	2	None	Wendy
john	John Smith	DMS	None	www.john.net	None	None	None

Projects				
ID	**Title**	**Description**	**Suggestor**	**External**
1	Project Title number one	Description of Project number one telling what it will involve and what skills it requires.	Wendy	No
2	External Project Title	Description of the company that the project will be done for and what the project is about.	Leeds United AFC	Yes

3	Project Title number two	Description of Project number two telling what it will involve and what skills it requires.	Dave	No
4	Project Title number three	Description of Project number three telling what it will involve and what skills it requires.	Jane	No
5	External Project Title number two	Description of the company that the project will be done for and what the project is about.	Halifax	Yes

Area of Interest		
ID	**Title**	**Suggestor**
6	Artificial Intelligence	Jane
7	Gigabit Wireless Networking	Wendy
8	The Grid	Wendy
9	Information System Strategy Trends	Jane
10	Distributed Information System Strategies	Dave

Members of Staff	
Username	**Name**
Jane	Jane Slack
Wendy	Wendy Bailey
Dave	Dave Entwhistle
Gillian	Gillian Booth
Harry	Harry Henry

Extensions					
Student	**Submission**	**Deadline**	**Issuer**	**Date**	**Reason**
bill	Preferences	28/02/2002	Gillian	13/02/2002	Family Death

Comments		
Student	**Date**	**Comments**
bill	15/06/2002	I think the site is very good
darren	20/05/2002	I think the site isn't very good for the submission of Interim Reports

Preferences							
Student	**Proj 1**	**Reas 1**	**Proj 2**	**Reas 2**	**Area**	**Reas**	**Date**
paul	3	Interested in the area	2	Got good grades in the area	9	Good job prospects	20/02/2002

Objectives				
Student	**Title**	**Objectives**	**Min Req.**	**Date**
Paul	Project Title	Some working prototye and a huge report	Report	15/03/2002

Legal Form			
Student	**Date**	**Company**	**School**

Paul	15/03/2002	Leeds United AFC	Yes

Interim Report			
Student	**Format**	**Date**	**File**
Paul	PS	01/05/2002	Auto

Demonstration			
Student	**Supervisor**	**Assessor**	**Date**
Paul	Jane	Wendy	05/09/2002

Final Report			
Student	**Format**	**Date**	**File**
paul	MS Word	06/09/2002	Auto

Assessment								
Student	**Date**	**Deliverables**	**Comments**	**U**	**S**	**E**	**W**	**R**
Paul	11/10/ 2002	Prototype and report	Very well documented	15	30	10	5	7

Appendix K: Test Results

[T01] – failed the missing data test because it allowed a company to be added without a name.
[T02] – failed the missing data test because it allowed Staff to be entered without names or usernames; it also failed the duplicated data test because it didn't deal with the error properly.
[T03] – failed the invalid data test because it allowed the student to be added but wouldn't allow access to the student's details, missing data test failed because it allowed a student to be added without a name, it also failed the duplicated data test because it didn't deal with the error properly.
[T04] – failed the missing data test because it allowed project to be entered without a title, or description.
[T05] – failed the missing data test because it allowed area to be entered without a title, or description.
[T07] – failed the missing data test because it allowed project to be entered without a title, or description.
[T08] – failed the missing data test because it allowed area to be entered without a title, or description.
[T17] – failed the missing data test because it allowed the feedback to be submitted without any feedback.
[T20] – failed the missing data test because it allowed a company to have no name, and also allowed the company's details to be changed to be the same as another company's.
[T21] – failed because it allow the deletion of a company with a project, and wouldn't allow the deletion of a company without.
[T23] – failed because it crashed when a single quote was included in the students name.
[T25] – failed because it allowed a project to be deleted that has been allocated to a student.
[T45] – didn't fail but the project count should include the areas of interest.
[T47] – failed because it hasn't counted the projects accurately.

Appendix L: Screen Shots