Master of Science Thesis for
Software Engineering

A web service for organizing
and evaluating students’ group work

By Alharith Almusa

Supervisor: Monica Farrow
19 /08 / 2010
A statement of non-plagiarism

I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed:

Date:
Abstract

The main objective of this project is to develop a web service which can assist students in the organisation and evaluation of their group work. By using this web service, students can select their group members, arrange meetings, and evaluate students’ contributions to the group. In addition, tutors can read summary reports to monitor how well the group works together. Therefore, this project aims to increase the degree of transparency in students’ group work, especially in terms of students’ contribution to their groups.

As result, a RESTful web service was developed which meets the requirements captured and analysed during the project. The style and technologies of the web service were chosen after performing research among various web service styles and technologies. The development methodology of the web service was based on Incremental development processes. This methodology was selected after investigating a number of methodologies.

Moreover, the web service was evaluated by conducting a number of interviews with its target users. The documentation of the web service was also evaluated by a number of web developers. The evaluation shows that the web service meets the aims and objectives of the project.

This report contains a review of literature on web service technologies. It also includes the detailed development processes of the project. In addition, the achievements and the limitations of the project are described. Recommendations for future work and research are also discussed in this report.
Acknowledgements

I would like to thank my supervisor Monica Farrow for her guidelines and suggestions.

I also would like thank my family and friends for their encouragement and support.
# Table of contents

## Chapter 1 Introduction .................................................................................................................. 9
  1.1 Motivation ................................................................................................................................ 9
  1.2 Aims and Objectives .................................................................................................................. 10
  1.3 Project scope and the intended users ....................................................................................... 11
  1.4 Outline .................................................................................................................................... 11

## Chapter 2 Background and Literature Review ............................................................................. 13
  2.1 The need of web services ......................................................................................................... 13
  2.2 What are web services? ........................................................................................................... 14
  2.3 Web services architecture ....................................................................................................... 15
      2.3.1 Web services messaging layer ......................................................................................... 17
      2.3.2 Web services description layer ....................................................................................... 19
      2.3.3 Web services discovery layer .......................................................................................... 20
  2.4 Web services styles .................................................................................................................. 21
      2.4.1 SOAP web services ......................................................................................................... 21
      2.4.2 RESTful web services .................................................................................................... 22
  2.5 Comparison between web services styles ............................................................................... 23
      2.5.1 Web service principles comparison ............................................................................... 23
      2.5.2 Web services technologies comparison ......................................................................... 25
      2.5.3 Summary and dissection ................................................................................................ 26
  2.6 Related work ............................................................................................................................ 27
      2.6.1 Amazon web services .................................................................................................... 27
      2.6.2 Google web services ...................................................................................................... 29

## Chapter 3 Requirements Capture and Specification .................................................................... 33
  3.1 The Waterfall methodology .................................................................................................... 30
  3.2 Iterative Incremental Development ......................................................................................... 31
  3.2 Choosing the methodology ..................................................................................................... 31

## Chapter 3 Requirements Capture and Specification .................................................................... 33
  4.1 Methods of Requirements Capture used in this project ......................................................... 33
      4.1.1 Questionnaire .................................................................................................................. 33
      4.1.2 Interviews ....................................................................................................................... 34
  4.2 Initial functional requirements .................................................................................................. 35
  4.3 Non-functional requirements .................................................................................................... 37
      4.3.1 Application interfaces (APIs) .......................................................................................... 37
      4.3.2 Technical requirements .................................................................................................. 38
  4.4 Evaluation requirements .......................................................................................................... 38
Chapter 5 – choice of tools and protocols ................................................................. 39
5.1 Authentication protocol .................................................................................. 39
5.2 Tools and technologies .................................................................................. 42
Chapter 6 - Database design and implementation .................................................. 45
6.1 The initial Entity Relationship Diagram ......................................................... 45
6.2 ER Diagram modification .............................................................................. 49
6.3 Adding table attributes and database implementation .................................. 49
6.4 Database testing ............................................................................................ 51
Chapter 7- The Development Increments ............................................................... 52
7.1 Further requirements analysis ....................................................................... 52
7.1.1 Increment one ............................................................................................ 52
7.1.2 Increment two ........................................................................................... 54
7.1.3 Increment three ......................................................................................... 56
7.1.4 Increment four .......................................................................................... 57
7.1.5 Increment five .......................................................................................... 58
7.2 Design ............................................................................................................ 60
7.2.1 Web service interfaces ............................................................................. 60
7.2.2 XML messages .......................................................................................... 62
7.2.3 HTTP responses and error messages ....................................................... 63
7.2.4 Code structure ......................................................................................... 65
7.2.5 Activity diagrams ..................................................................................... 67
7.3 Implementation .............................................................................................. 73
7.3.1 Database connection ............................................................................... 73
7.3.2 Converting Java objects to XML documents ......................................... 74
7.3.3 Implementing the web service interfaces .............................................. 75
7.3.4 Mapping Java Exceptions to HTTP responses ...................................... 77
7.4 Testing ............................................................................................................ 79
Chapter 8 – Evaluation ......................................................................................... 81
8.1 Evaluation Plan .............................................................................................. 81
8.1.1 The content of the web service ............................................................... 81
8.1.2 Interviews with students ......................................................................... 82
8.1.3 Interviews with teachers ......................................................................... 83
8.2 The documentation of the web service .......................................................... 84
8.3 Evaluation discussion .................................................................................... 85
Chapter 9 – Summary .......................................................................................... 86
9.1 Achievements ................................................................................................. 86
9.2 Limitations............................................................................................................. 87
9.3 Recommendations for future work........................................................................... 88
9.4 Conclusion............................................................................................................... 90
References ..................................................................................................................... 91

Appendix A - Students group work questionnaire.
Appendix B - SQL Create Statements for the database.
Appendix C - Textual description for Use case diagrams.
Appendix D - The designed web services Resources with their descriptions.
Appendix E - XML Schemas for the XML messages.
Appendix F - The class diagram of the web service.
Appendix G - The documentation of the web service Resources.
Appendix H - Interviews questions.
# Table of Figures

- Figure 1: Web service roles (Cerami, 2002) .......................................................... 16
- Figure 2: Web service protocol stack (Cerami, 2002) ............................................. 17
- Figure 3: SOAP request for a weather service (Cerami, 2002) .............................. 18
- Figure 4: SOAP response for a weather service (Cerami, 2002) ........................... 18
- Figure 5: XML-RPC request format for a weather web service (Cerami, 2002) ....... 19
- Figure 6: XML-RPC response format for a weather web service (Cerami, 2002) .... 19
- Figure 7: WSDL document for weather web service (Cerami 2002) ....................... 20
- Figure 8: SOAP web service operations (Snell et al, 2001) ................................... 22
- Figure 9: URI examples of Buckets and Objects (Murty, 2008) ............................. 27
- Figure 10: HTTP verbs and their operations (Murty, 2008) .................................... 28
- Figure 11: URI examples of Buckets and Objects (Murty, 2008) .......................... 28
- Figure 12: The Waterfall Methodology .................................................................. 30
- Figure 13: Incremental Development Processes .................................................... 31
- Figure 14: The methodology used in this project ................................................... 32
- Figure 15: the Authentication mechanism used in the web service ........................ 41
- Figure 16: the architecture of the developed system and the testing enjoinder ........... 44
- Figure 17: the initial ER diagram ........................................................................... 45
- Figure 18: the ER diagram after the modifications .................................................. 49
- Figure 19: The final ER diagram ............................................................................ 50
- Figure 20: Use case diagram for increment one ..................................................... 54
- Figure 21: Use case diagram for increment two ....................................................... 55
- Figure 22: Use case diagram for increment three .................................................... 57
- Figure 23: the use case diagram for increment four ................................................. 58
- Figure 24: Use case diagram for increment five ....................................................... 59
- Figure 25: modules resource and its sub-resources ............................................... 61
- Figure 26: a list of URIs for modules resource ....................................................... 62
- Figure 27: a tree diagram for the XML document "login-info" ................................. 63
- Figure 28: a tree diagram for the XML document "assignment" ............................... 63
- Figure 29: a class diagram for the resource "modules" ........................................... 65
- Figure 30: the designed java classes and their packages ........................................ 66
- Figure 31: the format of the key which used for users' Authentication ..................... 67
- Figure 32: Activity diagram for the login operation ............................................... 68
- Figure 33: Activity diagram for Create group operation ......................................... 69
- Figure 34: Activity diagram for “Create a join request” operation ......................... 70
- Figure 35: Activity diagram for “Response to a join request” operation ................. 72
- Figure 36: a sample code for connecting the web service with the database using connection pooling mechanism .............................................................. 74
- Figure 37: a java class uses JAXB annotations ..................................................... 75
- Figure 38: a java interface uses JAX-RS annotations .............................................. 76
- Figure 39: an example of mapping Java Exceptions to HTTP responses ............... 78
- Figure 40: a screen shot from a web page used to test the login resource ............... 79
Chapter 1  Introduction

This chapter describes the motivation, the aims, and the objectives of the project. In addition, it contains an outline which describes the structure of the report.

1.1 Motivation.

As a student, I participated with groups of students on a number of assignments during my undergraduate and postgraduate programs. Working with groups of students improved my skills and knowledge, and usually group members vary in these aspects. Perhaps these differences between group members do not limit the performance of the group as a whole, because students can assign different roles to each member depending on their skills and knowledge. In this case, some students may be assigned more work and it may appear that they do a greater percentage of the work than other students in the same group. However, all of them may spend the same effort and would gain the same knowledge from the assignment. Therefore, differences in skills and knowledge among group members would not be a significant issue.

On the other hand, it could be a serious issue in the group if there are differences in their concerns and objectives with regard to their assignment. Some students’ concern is simply to pass the course work, while others want to achieve the highest marks possible. This could lead to variation between students in terms of the effort spent on the course work, which could affect the group’s overall output. Thus, some students working in such groups would not be satisfied with the results.

Moreover, some group assignments require students to write individual reports that reflect their work on the assignments. The teacher can then judge whether they deserve equal marks or not. This can address most of the issues, especially for students who have excellent writing skills. However, students who are less skilled in writing may not be able to reflect their actual contributions on their assignments. Therefore, addressing this issue has suggested the idea of developing a system which allows students to evaluate each other in terms of their contributions to the group. In addition, teachers can use this system to gain detailed reports about students’ group work.
Furthermore, I learned the principles and concepts of web services which have been developed in the recent years. In my opinion, these technologies could be improved and would be the future of web applications. At the time of learning these technologies, I did not implement them practically, and this project was a good chance to try to practice them. In addition, the potential system could be used by different universities which would have different systems in terms of User Interfaces, programming languages and technologies. This was another reason to apply web service concepts in the potential system, as they are deemed to address this problem.

1.2 Aims and Objectives

The project aims to develop a system that can aid students to organise and evaluate their group work on an assignment. In addition, the system can provide tutors with reports that reflect students’ performance within a group. Therefore, this project aims to increase the degree of transparency in students’ group work, especially in terms of students’ contribution to their groups.

Moreover, the project also aims to apply web services concepts to the potential system. A main concept of web services is that they intend to allow interaction between applications located in a heterogeneous environment where they differ in terms of technologies and programming languages (Cerami, 2002). Thus, the project intends to apply this concept in the potential system.

In order to achieve these aims, the project has a number of objectives which are listed as follows:

- To perform a literature review about web services technologies in order to gain more understanding about their concepts.
- To compare and investigate web service styles and to choose a suitable style for this project.
- To compare a number of software development methodologies to choose a suitable methodology for the project.
- To investigate suitable tools and technologies to be used in the project.
- To perform a survey of target users in order to clarify the functionalities that the system should provide.
To perform software development processes based on the selected methodology in order to produce the potential system.

To evaluate the product of the project in order to determine whether or not the project achieved its aims and objectives.

1.3 Project scope and the intended users

This project focused on the development of a web service. This web service was designed to be invoked by various types of client applications. Therefore, the main consideration of this project was to use the appropriate technologies and protocols in order to allow different types of client applications to invoke the web service. In addition, the functionalities of the web service were designed to be used by students working in groups to accomplish assignments required for their modules. Moreover, other functionalities were designed and implemented for tutors who teach modules that have a number of assignments requiring students to work in groups. As students and tutors are the main users of the web service, they need to use client applications in order to use the functionalities which the web service provides. The development of client applications of the web service is outside the scope of this project, which focused on the development of the web service.

1.4 Outline

This report contains nine chapters which are listed and described as follows.

Chapter two- Background and Literature review

This chapter contains a literature review about web services technologies. It mainly focuses on these technologies in order to gain more understanding about them and to find a suitable style among the various styles of web service available.

Chapter three - Development methodologies

This chapter compares a number of methodologies in order to choose a suitable methodology for this project. In addition, the selected methodology is described in this chapter.
Chapter 4 - Requirements Capture and Specification
This chapter describes the methods used to capture the requirements. It then describes and analyses the surveys and interviews performed as part of this project. In addition, the initial functional requirements and non-functional requirements are listed and described in this chapter.

Chapter 5 – choice of tools and protocols
In this chapter, the Authentication protocols which could be used by this project are compared and discussed in order to choose a suitable protocol. Then, the tools and the technologies which were used in this project are listed and discussed.

Chapter 6 - Database design and implementation
In this chapter, the stages of analysing, designing, implementing, and testing the database are listed and discussed in detail.

Chapter 7- The Development Increments
This chapter describes the development increments of this project. First, it describes the process of assigning the requirements to each increment. Then, it describes all of the development processes of each increment such as further requirements analysis, design, implementation, and testing.

Chapter 8 - Evaluation
This chapter discusses the evaluation plan of the project. The surveys conducted for the evaluation are then described and discussed. In addition, the evaluation and the result are discussed in this chapter.

Chapter 9 – Summary
In this chapter, the achievements and the limitations of this project are described and discussed. In addition, recommendations for further work are also discussed in detail. Finally, this chapter contains the conclusion of the project.
Chapter 2  
Background and Literature Review

This literature review mainly focuses on web services technologies. First, the literature review will go through the main objectives for developing web services and why they are important. Web service concepts will then be defined in detail. The architecture of web services will also be described in detail. After that, the literature review will look at various web service styles and will identify the differences between each style. In addition, there will be a comparison between these styles in terms of concepts and technologies. Finally, the research will discuss some of the related work which has been done using web services concepts.

2.1 The need of web services

During the literature review, a number of objectives have been found that clarify why web services are an important approach. These objectives are listed and described below.

First, web services aim to enable applications in distributed environments to interoperate with each other seamlessly. According to (Cerami, 2002), “web services are ultimately designed to improve interoperability between applications”. Therefore, data can more easily be exchanged between different applications and different platforms.

In order to achieve interoperability, specialized software components need to be implemented to allow incompatible systems to work and communicate together, which requires expertise with both systems. Web services promise to provide interoperability at a much lower cost. Therefore, providing interoperability is the main purpose of using web services (Chappell and Jewell, 2002).

According to (Snell at al, 2001) “a client application that consumes a web service can run on any device with Internet connectivity and the ability to serialize and parse XML”. In other words, PC devices and other devices such as mobile phones and PDAs, can utilize web services. Therefore, web services improve the interaction between applications even when they are in different devices.
In addition, web services can offer reusable application components. Applications or components such as currency conversion, language translation or weather forecasting may be used by frequently. Instead of implementing them over and over again, they can be reused easily by using a web service-based mechanism. This will save a great deal of time and effort for developers and will take the advantage of being able to use the excellent applications which exist on the Internet (Chappell and Jewell, 2002).

2.2 What are web services?

During this research, a number of definitions of web services have been found. Some of these provide only a general idea of web services while others provide more details. In the following section, definitions from different resources are described.

First, (Cerami, 2002) defines a web service as “a piece of business logic, located somewhere on the Internet, that is accessible through standard-based Internet protocols such as HTTP or SMTP”. According to another definition by (Chappell and Jewell, 2002), “A web service is any service that is available over the Internet, uses a standardized XML messaging system, and is not tied to any one operating system or programming language”.

Therefore, web services are web-based applications that have special behavioral characteristics that make them successful in improving the interoperability of applications distributed in heterogeneous environments (Chappell and Jewell, 2002). One of these characteristics is that of using XML as the data representation layer. This allows web services to interact with others at their core level. For data transport, protocols for platform binding, networking or operating system are eliminated by XML in order to improve the interoperability between these applications (Cerami, 2002).

Another characteristic of web services is that they have to be loosely coupled. Developing a client and a server in closely tied logic implies a tightly coupled system. This means that if one of their interfaces changes, the other must also to be changed. Web services tend to address this issue, and their interfaces must have the ability to be changed over time without affecting the ability of clients to interact with them. This
characteristic improves the manageability of software systems and reduces the effort involved in integration between various systems (Snell et al., 2001).

Moreover, web services have the ability to be synchronous or asynchronous. Synchronous invocations mean that clients are blocked and must wait until the operation of a service is completed in order to continue processing other operations. On the other hand, asynchronous invocations mean that clients are not blocked and they can perform other operations before the operation of a service is completed. Thus, asynchronous ability is an important aspect in the development of loosely coupled systems (Chappell and Jewell, 2002).

In addition, web services support Remote Procedure Calls (RPCs). Clients can invoke remote objects’ functions, procedures and methods by using an XML-based protocol. Remote procedures describe what inputs and outputs parameters which should be supported by web services. Similarly, component-based development tools such as Enterprise JavaBeans (EJBs) and .NET are distributed and can be accessible via different RPC mechanisms. Web services provide services of their own and equivalent to EJBs and .NET mechanisms in order to support RPC (Cerami, 2002).

Another characteristic of web services support document exchange. Documents can be very simple, such as an address, or they can be very complex, such as an entire book. Web services have the advantage of using XML to represent these kinds of documents. This characteristic of web services makes business integration much easier (Snell et al., 2001).

2.3 Web services architecture.

In this section, the research will describe the architecture of web services. In addition, more details about the architecture will be provided in the sub sections below.

According to (Cerami, 2002), “There are two ways to view the web service architecture. The first is to examine the individual roles of each web service actor; the second is to examine the emerging web service protocol stack”.

15
Firstly, roles-based web service architecture consists of three major roles: service provider, service requestor, and service registry. The service provider is responsible for implementing the service and publishing it in order for it to be available on the internet. The service requestor is any client who wants to use a web service. In order to use the web service, the requestor is required to be able to connect to the internet, and also to be able to encode and decode XML documents. The service registry is a central place between service providers and service requesters so that developers can register new web services in the service registry and service requesters can find them there (Cerami, 2002). Figure 1 illustrates web services’ roles and how they interact with each other.

![Figure 1: Web service roles (Cerami, 2002).](image)

The second way of viewing web services architecture is as a protocol stack. The protocol stack-based web service architecture consists of four layers: transport layer, messaging layer, description layer, and discovery layer (Snell et al, 2001).

The transport layer is responsible for transporting messages between applications (Cerami, 2002). Currently, a number of protocols are included in this layer such as Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and Blocks Extensible Exchange Protocol (BEEP).

The messaging layer is responsible for standardising messages in XML format to be understandable by both service requesters and service providers (Cerami, 2002). SOAP and XML-RPC are examples of protocols included in this layer.
The description layer is responsible for describing web services’ interfaces (Chappell and Jewell, 2002). Therefore, service providers utilize protocols in this layer in order to describe their web services to service requesters. Currently, Web Services Description Language (WSDL) is the only protocol included in this layer.

The discovery layer is responsible for locating web services into a common directory. It is also responsible for providing a mechanism of publishing and finding web services in this directory (Cerami, 2002). Thus, by using this layer, service providers can implement and publish their web services in this directory so that services requesters can search for and use these web services. Figure 2 shows a web service protocol stack, identifying which protocols could be used in each layer.

![Figure 2: Web service protocol stack (Cerami, 2002).](image)

In the following sub sections, each layer of the protocol stack is described in more detail.

### 2.3.1 Web services messaging layer

According to (Snell et al, 2001), there are two protocols that are used commonly for exchanging messages in web services. The first protocol is Simple Object Access Protocol (SOAP) which is used to transport messages between service requesters and services providers over the internet. By using a message format which is defined in Web Service Document Language (WSDL), SOAP messages can be issued for both the requests and the responses of the web services. One advantages of SOAP protocol for messaging is its extensibility. Another advantage is that SOAP is platform and language independent. SOAP has these advantages because it is written in XML.
Since SOAP uses Hypertext Transfer Protocol (HTTP), SOAP messages are able to get around firewalls because HTTP is usually supported by firewalls (Cerami, 2002). Figures 3 and 4 show a SOAP message request and response format for a Weather Web Service.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2001/09 SOAP-ENV/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <nsi:getWeather
            xmlns:nsi="urn:examples:weatherservice"
            SOAP-ENV:encodingStyle="http://www.w3.org/2001/09 SOAP-ENV/"
            xsi:type="xsd:string">10016</zipcode>
    </nsi:getWeatherResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 3: SOAP request for a weather service (Cerami, 2002).

```xml
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2001/09 SOAP-ENV/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <nsi:getWeatherResponse
            xmlns:nsi="urn:examples:weatherservice"
            SOAP-ENV:encodingStyle="http://www.w3.org/2001/09 SOAP-ENV/"
            xsi:type="xsd:int">65</return>
    </nsi:getWeatherResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 4: SOAP response for a weather service (Cerami, 2002).

The second protocol in common use is XML-RPC which is similar to SOAP. As it is also based on XML, it has the same advantages such as extensibility and being platform independent. However, the format of XML-RPC is simpler than the format of SOAP. Therefore, SOAP has more features, especially with regard to security features (Cerami, 2002). Figures 5 and 6 show the format of XML-RPC request and response messages for a weather web service.
2.3.2 Web services description layer

Currently, there is only one protocol in the description layer. This protocol is Web Services Description Language (WSDL). WSDL is written in XML and used to specify the interface of web services. This includes information about data types used in an XML message; information about the transport protocol which is used such as HTTP and SMTP; and information about the location of the web service which is invoked. In addition, it has an extension for message protocol: this service currently applies for the SOAP service, but it is not tied to SOAP and can be used to describe other protocols such as XML-RPC. Moreover, WSDL documents are usually complex and difficult for humans to read, especially when there are a lot of operations involved. Therefore, there are a number of tools used for parsing WSDL documents which can generate client applications to invoke web services (Chappell and Jewell, 2002). Figure 7 shows the WSDL document of Weather Web Service.
2.3.3 Web services discovery layer

UDDI stands for Universal Description, Discovery and Integration. It can be used as a directory of web services, and also as a directory for businesses. According to (Cerami, 2002), “UDDI was originally created by Microsoft, IBM, and Ariba, and
represents a technical specification for publishing and finding businesses and web services”. Like SOAP and other web service protocols, UDDI is platform-independent. The data stored in UDDI is divided into three categories: white pages, yellow pages, and green pages. Businesses’ information such as the business’s name and address is located in the first category. The classification of the data of both business of web service is located in the second category. The third category has technical information about web services (Cerami, 2002).

However, according to (Pautasso at al, 2008), “While UDDI has been mature and stable for several years now, it has failed to reach widespread acceptance in the industry”. In other words, most companies have stopped using an UDDI approach globally. However, UDDI is still being used inside companies’ domains to centralize their web services within the companies.

2.4 Web services styles

During this research, two major styles of web service have been found. The first style is based on the WS-I profile which recommends using a number of protocols in order to improve the interoperability of web services. The second style is based on REST concepts which utilize HTTP operations and URIs for acting as web services. These styles are described in the sub-sections below.

2.4.1 SOAP web services

In this research, SOAP web services refer to developing web services using protocols that are recommended in the WS-I profile. WS-I stand for Web Services Interoperability Organization, which publishes profiles containing a number of web services protocols that are recommended to be used in order to improve the interoperability of web services (Gailey, 2004). This profile recommends using a number of protocols such as SOAP protocol for messages, WSDL for description and UDDI for discovery (Gailey, 2004). These protocols are described in the previous sections, and Figure 8 illustrates their interactions.
2.4.2 RESTful web services

Another approach to web services is RESTful web services (Richardson and Ruby, 2007). REST stands for Representational State Transfer. “REST by itself is not an architecture; REST is a set of constraints that, when applied to the design of a system, creates a software architectural style”. It utilizes existing web standards and protocols such as HTTP and XML. According to (Richardson and Ruby, 2007), RESTful web services are based on four principles.

The first principle is resources identification through URI. A set of resources can be identified by Unified Resource Identifications (URIs). These resources are identified by a RESTful web service, and the clients can use URIs to interact with these resources.

The second principle is uniform interface. Clients can manipulate resources by using a fixed set of HTTP’s operations. The most common operations are creating, reading, updating and deleting resources. These operations are performed by using HTTP’s PUT, GET, POST, and DELETE operations.

The third principle is self-descriptive messages. Resources’ content can be retrieved in different formats such as HTML, XML, PDF, etc. because these resources are decoupled from their original representations.
The fourth principle is stateful interactions via hyperlinks. Interactions with resources must be stateless. Explicit state transfer is the concept on which these interactions are based. In order to perform this, many approaches are used for changes of state such as URI rewriting, hidden fields and cookies.

2.5 Comparison between web services styles.

This section aims to compare SOAP with RESTful web service styles. The comparison is based on two methods. The first method is based on web service principles and the second method is based on web service technologies.

2.5.1 Web service principles comparison.

In this section, three principles of web services are examined. The principles are protocol stack, dealing with heterogeneous systems, and defining loosely coupled concepts.

Firstly, in terms of protocol stack, RESTful web services mainly use the web as a common medium for publishing global information, meaning that applications use URIs in identifying resources, information and services and utilize all of HTTP’s verbs (PUT, DELETE, POST and GET) (Muehlen at al, 2005).

On the other hand, SOAP web services use the web as a transport medium for exchanging messages between web service endpoints. HTTP protocol is mainly used to go around firewalls and only one verb is used (POST) to allow XML documents to transfer between web services. Therefore, instead of using HTTP to identify operations, they are identified via SOAP messages (Muehlen at al, 2005).

The second principle is dealing with heterogeneous systems. According to (Pautasso at al, 2008), both service types deal with heterogeneous systems in an efficient manner. The most common issue is the difference between browsers in dealing with incompatible JavaScript libraries and HTML pages. However, all of the browsers use the same transport protocol (HTTP). In addition, SOAP web services can deal with
more complex heterogeneous systems such as Enterprise computing where the
domain is a collection of autonomous distributed systems.

The third principle is defining loosely coupled concepts. According to (Gailey 2004),
there are four important aspects of a distributed system which define it as loosely
coupled. The first aspect is availability: whether or not the client would be affected
when the server is down. In this aspect, SOAP web services address this problem by
using document-based messages as opposed to RPC-based messages (Snell, Tidwell
and Kulchenko 2001). SOAP web services can use both types of messages while
RESTful web services use only RPC-based messages and so do not address this
problem (Muehlen at al, 2005).

The second aspect is the latency between client and the server. As SOAP web services
apply asynchronous interactions, the latency between the requester and the responder
is not an issue. However, RESTful web services do not support asynchronous
interactions, so this is an issue in terms of loose coupling (Richardson and Ruby,
2007).

Another aspect is the transparency of web services’ locations. SOAP web services are
more efficient in this aspect thanks to the UDDI protocol which addresses this
problem. On the other hand, RESTful web services deal with this problem by using
DNS address translation which increases the effort required to configure network
infrastructure (Pautasso at al, 2008).

The third aspect is service evolution. In this case, RESTful web services are more
efficient because they use the common HTTP verbs as the interface between
applications. Therefore, the interfaces of RESTful web services are unlikely to change
over time. On the other hand, interfaces of SOAP web services may change over time,
which would cause some issues (Pautasso at al, 2008). However, by using XML as
the messaging format, this issue is almost addressed (Richardson and Ruby, 2007).


2.5.2 Web services technologies comparison.

This section aims to examine what protocols different web services offer in each layer of the protocol stack. Protocol stack layers such as Transport, Payload, Identification, Description and Security are discussed as follows.

1- Transport
RESTful web services offer only one option in terms of transport protocols. They use HTTP for exchanging messages between web services and do not offer other options. On the other hand, SOAP web services have a variety of options, such as HTTP, STMP and FTP. By using WSDL documents, SOAP web services can identify which transport protocol is used (Pautasso at al, 2008).

2- Payload
SOAP web services use only XML-based messages as their payload format. In contrast, RESTful web services offer a variety of formats such as XML, JSON and MIMI. This aspect can be seen as a disadvantage for RESTful web services because it may reduce the interoperability between web services. This can be seen if a client is expecting to receive a message in a XML format but the server responds with a message in MIMI format. Therefore, unifying the format of the payload, can improve the interoperability between web services (Pautasso at al, 2008).

3- Identification
RESTful web services utilize URIs for identifying their services. There are positive aspects of using URIs. One advantage is that all of the information which is needed to identify a web service can be encapsulated in one URI. Another advantage is that URIs can easily be bookmarked or shared between consumers. On the other hand, SOAP web services have a standard addressing approach and only use URIs to identify the endpoints of web services in terms of exchanging messages (Muehlen at al, 2005).

4- Description
SOAP web services use WSDL documents to describe their services. WSDL documents contain all of the information which developers need in order to create
clients for web services. In addition, there are a number of tools which can be used to generate clients automatically. In contrast, RESTful web services use human readable documents to describe their services. If these documents are well written, they could make it easier for developers to create clients than using WSDL documents. However, developers would be limited in creating stubs and would not be able to take advantage of automatic generation from WSDL documents (Pautasso et al, 2008).

5- Security
RESTful web services utilise HTTP protocol security. They use HTTPS (point to point SSL security) in order to provide their messages with more security. While SOAP web services use the same approach for security, they can provide extra features such as a set of options to configure the QoS of exchanged messages. Therefore, SOAP web services out-compete RESTful in terms of security (Muehlen et al, 2005).

2.5.3 Summary and dissection.

From the comparison above, it can be seen that the two styles have a number of common features. For example, both of them can use HTTP as transport protocol and use XML as format for messaging documents. On the other hand, they have differences on a number in supporting a number of features. SOAP web services use WSDL for describing the web services while RESTful web services use human readable documents. SOAP web services support a variety of transport protocol such as HTTP, STMP and FTP however RESTful web services support just one transport protocol which is HTTP. In addition, they have slight differences in web service concepts and the protocols stack.

To summarize, SOAP web services provide more features and support more protocols. This can increase the popularity of using SOAP style rather than RESTful style especially when a web service requires more interactions. However, RESTful style seems much simpler than SOAP style and also provides have similar futures. Perhaps they take less time in terms of implementing the server side of the web
services. Therefore, RESTful style was selected as the style of the developed web service in this project. It was selected because the simplicity in terms of the implementation and also because it has the required futures for the proposed project.

2.6 Related work

During the research, a number of related works have been found. Some of the related works have been developed by Amazon, and others by Google. In the following sections, these web services are described.

2.6.1 Amazon web services

Amazon offers a number of web services that can be used by developers in facilitating applications developments (Murty, 2008). Amazon’s web services S3 and FPS are described below:

1- **Amazon Simple Storage Service (S3)**

Amazon offers an alternative method for storing data. It manipulates web service concepts to enable developers to utilize their services. This can be seen in its web service, called Amazon Simple Storage Service (S3). This web service aims to provide an online storage directory in which developers can back up their storage systems. In addition, they can use a number of methods to access the stored data and can have control over how their data is represented in the web services. Moreover, Amazon offers two types of web services in order to utilize this service. The first type is a RESTful web service and the second is a SOAP web service (Murty, 2008).

In the RESTful-type service, the resources are identified by using URIs and are divided into two kinds of resources: objects and buckets. Data and metadata are stored as objects. These objects are stored in containers which are called buckets. There are no limits for the number of objects that can be stored in each bucket (Murty, 2008).

<table>
<thead>
<tr>
<th>Resource</th>
<th>Example URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3 Service</td>
<td><a href="http://s3.amazonaws.com/">http://s3.amazonaws.com/</a></td>
</tr>
<tr>
<td>Bucket</td>
<td><a href="http://s3.amazonaws.com/bucket_name">http://s3.amazonaws.com/bucket_name</a></td>
</tr>
<tr>
<td>Object</td>
<td><a href="http://s3.amazonaws.com/bucket_name/objectkey">http://s3.amazonaws.com/bucket_name/objectkey</a></td>
</tr>
</tbody>
</table>

Figure 9: URI examples of Buckets and Objects (Murty, 2008).
Moreover, S3 RESTful web services utilize five verbs of the HTTP protocol. The verbs used are GET, HEAD, PUT, DELETE and POST.

<table>
<thead>
<tr>
<th>Resource</th>
<th>GET</th>
<th>HEAD</th>
<th>PUT</th>
<th>DELETE</th>
<th>POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3 Service</td>
<td>List your buckets</td>
<td>-</td>
<td>Create the bucket</td>
<td>Delete the bucket</td>
<td>-</td>
</tr>
<tr>
<td>Bucket</td>
<td>List the bucket’s objects</td>
<td>Retrieve the object’s data and metadata</td>
<td>Create or replace the object</td>
<td>Create or replace the object</td>
<td>-</td>
</tr>
<tr>
<td>Object</td>
<td>Retrieve the object’s metadata</td>
<td>-</td>
<td>Apply new ACL settings</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ACL (for a Bucket or Object resource)</td>
<td>Retrieve ACL settings</td>
<td>-</td>
<td>Apply new ACL settings</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Figure 10: HTTP verbs and their operations (Murty, 2008).

2- Flexible Payments Service (FPS)

Another Amazon web service is Flexible Payments Service (FPS). It allows developers to provide their applications with a payments service with minimum risks. Transactions between individuals and businesses can be made by using Amazon Payments accounts. Therefore, personal banking information need not be used, reducing the immediate security aspects of the transaction. This web service uses a RESTful web service approach, based on URIs for resources identification and HTTPS for transport and operations. It uses HTTPS to provide more security for the messages exchanged between clients and the web services. In addition, the web service utilizes only two verbs of the HTTP protocol, GET, and POST. GET is used for requesting information and POST is used for performing transactions. The verbs PUT and DELETE are not needed because the web service does not create or delete Amazon Payment accounts (Murty, 2008).

<table>
<thead>
<tr>
<th>Resource</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Management website</td>
<td><a href="https://payments.amazon.com">https://payments.amazon.com</a></td>
</tr>
<tr>
<td>API</td>
<td><a href="https://fps.amazonaws.com/">https://fps.amazonaws.com/</a></td>
</tr>
<tr>
<td>UI Pipeline</td>
<td><a href="https://authorize.payments.amazon.com/cobranded-ui/actions/start">https://authorize.payments.amazon.com/cobranded-ui/actions/start</a></td>
</tr>
</tbody>
</table>

Figure 11: URI examples of Buckets and Objects (Murty, 2008).
2.6.2 Google web services

Google provides developers with a web service for searching the internet. By using this web service, developers provide their application with a searching tool in a simple method (Calishain and Dornfest, 2003). This web service uses a SOAP web service approach which is based on exchanging XML messages between clients and servers. In other words, developers can use the WSDL document which contains information about the format of SOAP messages, location of the services, transport protocol, etc. Therefore, they can create their clients and request search queries (Calishain and Dornfest, 2003).
Chapter 3  Development methodologies

Before analysing the requirements of this project, it was necessary to choose an appropriate methodology for this project. In the following paragraphs, two popular and well known methodologies are described briefly, then aspects of choosing the appropriate methodology are discussed.

3.1 The Waterfall methodology.

One of the popular versions of the Systems Development Life Cycle is the Waterfall methodology. Normally, it consists of six phases such as project planning, requirements analysis, design, implementation, testing, and evaluation. Some versions of the waterfall model demand that each phase must be completed before moving to the next phase, and that there be no backward movements until the last phase.

However, other versions allow backward movements to or from each phase for the purpose of modification. Usually, mistakes in one phase can be seen more clearly in the next phase and could give rise to consequences for the whole project. Therefore, these versions are more often seen in reality than the other versions which do not allow backward movements. Figure 12 shows a diagram of the waterfall methodology.
3.1 Iterative Incremental Development

Another approach to the Systems Development Life Cycle is Iterative Incremental development. Unlike the waterfall model, it breaks down the requirements into increments, each of which provides a part of the functionality. In addition, each increment is considered as a mini-project and includes further requirements analysis, design, implementation and testing. At the end of each increment, the system is integrated and tested. Figure 13 shows a diagram of the Iterative Incremental Development.

![Diagram of Iterative Incremental Development Processes](image)

3.2 Choosing the methodology

For this project, there are a number of aspects which were considered when choosing an appropriate development methodology. One aspect is the scale of the project, which is quite large. Many system functionalities were found in the initial planning phase. Another aspect is that some of the technologies used were quite new to me and the time and effort involved in learning and using these technologies is hard to predict. Therefore, the waterfall methodology was not deemed to be appropriate, especially in the implementation phase of this project.

Another aspect which led to the exclusion of some methodologies such as Agile Process Development is that the project was to be developed by only one developer,
and some of these methodologies require multiple developers in order to apply principles such as pair programming.

Another aspect is that the main focus of this project is to produce a web service which can be invoked by other applications. In other words, the project will produce application interfaces (APIs) instead of user interfaces, so involving users during the development is not important as they will not interact with the system directly. Therefore, the appropriate users who could evaluate this web service are the developers who would implement client applications which invoke the web service. However, this approach was not possible.

After considering these aspects and investigating a number of methodologies, IID were found to be more relevant to this type of project. However, system evaluation will be the last phase and will not be performed in every increment of the development cycle. Figure 14 shows a diagram of the methodology used in the project.

![Figure 14: The methodology used in this project.](image)
Chapter 3  Requirements Capture and Specification

In this chapter, the methods used for capturing the requirements are described. The questionnaire, interviews and their results are analysed and discussed. In addition, the initial functional requirements and non-functional requirements are listed and described.

4.1 Methods of Requirements Capture used in this project

The functional requirements of this project were captured by three methods. As I had worked with a number of student groups previously, the first method was brainstorming and trying to find out what important functionalities the system must provide. The second method was conducting interviews with the target users of the system. The third method was conducting a survey by asking students from different universities to fill in questionnaires. After that, the data gathered from these interviews and questionnaires were analysed.

Secondly, a research of existing systems was done to capture system requirements. It was difficult to find web services providing a similar number of functionalities. However, some of them were similar in the type of functionalities which were provided.

4.1.1 Questionnaire

The survey was conducted to gather the functional requirements of the system. The sample was taken from among postgraduate and undergraduate students who had worked with groups previously and were studying at several different universities. The questionnaires consisted of seventeen questions and focused on a number of points such as group allocation, group contacts, work management, and group evaluation. They were written in web format using Google documents and were e-mailed to the subjects (see Appendix A).

The questionnaire was also e-mailed to my classmates and friends at Heriot-Watt University and other universities. They were then asked to forward the questionnaire to their friends as well.
After analysing the results, it was found that the total response to this questionnaire included forty-three students, the majority of whom were male. In more detail, sixty-five percent of the students were male and fifty-seven percent of the total responses were from postgraduate students.

In addition, the range of group sizes that the students worked with ranged from three to seven students. However, the majority of the groups contained four students.

Moreover, when the students were asked whether they had difficulties in joining their groups, most of the undergraduate students answered that they did not. On the other hand, unlike undergraduate students, most of the postgraduate students had difficulties when they attempted to organise themselves into groups. The length of the programmes may be a major factor in these results: the length of undergraduate programmes is far longer than that of postgraduate programmes, especially when it is considered that most of the postgraduates who participated in this survey were Masters Degree students. This means that undergraduate students can usually make more friends during their courses and that can make it easier for them to organise their groups.

Furthermore, the majority of the subjects had chosen one of their group members to be a group leader, which in most cases was a random choice. Most of the group leader’s tasks involved scheduling meetings and assigning tasks to group members. In addition, most of the subjects had no other roles in their groups.

Moreover, when the subjects were asked to evaluate each member’s contributions to the group, there was a variety of answers but the majority answered that most of the work was done by less than seventy percent of the group’s members. This shows that students need a system by which they can evaluate the contribution of each other.

### 4.1.2 Interviews

The interviews were conducted with a number of students and teachers at Heriot Watt University. The objective of these interviews was to capture further requirements for the system, especially from the teacher’s point of view. It was difficult to arrange a
large number of interviews because most of the students and university staff were on holiday at the time. Eight interviews were conducted, two of them with teaching staff. The information gathered from these interviews was important and provided a number of points that helped to specify the problem.

One point to bear in mind is that some undergraduate courses do not provide students the option of organising their groups. In these cases, group organisation was done by the tutors of these courses.

Moreover, some courses deal with group coursework as individual coursework so that each student works alone within the group. In other words, a course work assignment has a number of tasks, each of which is done by only one member of the group, then the course tutor marks each individual task and not the whole group.

In addition, the method used to mark coursework often varies between one course and another. Some courses provide a group with the same mark whereas other courses provide the students with marks dependent on their contribution to the group.

Another variable is the method of calculating each student’s contribution to the group. Some courses require students to write individual reports which describe their contribution and other students’ contribution. The tutor measuring the contribution depends on the quality of the reports and the points on which the majority of the group members agreed. Another method is that each student states each member’s contribution as a percentage, then the tutor takes the average of these percentages and decides the actual contribution.

4.2 Initial functional requirements

After conducting the survey and interviews described above, a number of initial functional requirements were captured. These requirements will undergo further analysis in later stages of the development life cycle. Brief descriptions of these requirements are provided as follows.
**RQ1 – Users authentication and account management.**
The system must provide an authentication mechanism in order to verify users’ identities and their authorities. In addition, users must be able to manage their accounts, so the system must provide the appropriate functionalities for this purpose.

**RQ2 – School registration.**
In school registration, university registrars are supposed to register Modules, Tutors and Students. In addition, they are expected to enrol Students and Tutors in Modules. Therefore, the system must provide functionalities which can perform these operations.

**RQ3 – Assignments management**
In assignment management, tutors must have functionalities for managing assignments in each module they teach. Therefore, the appropriate functionalities for assignment management must be provided.

**RQ4 – Group organization.**
During an assignment period, students are expected to organize themselves into groups. In addition, Tutors may support them in group organization. Therefore, the system must provide functionalities for group organization.

**RQ5 – Work management**
When students have organized themselves into groups, they are expected to manage the work by assigning tasks and scheduling meetings. Therefore, the appropriate functionalities for work management must be provided.

**RQ6 – Work evaluation**
When the deadline for an assignment has passed, students are supposed to evaluate each other in terms of each student’s contribution to the group and also the quality of their work on tasks. Thus, the system must provide the appropriate functionalities for work evaluation.
**RQ7- Summary reports**

During and after an assignment period, tutors are expected to view reports that show statistics about each group and also each individual student. Therefore, the appropriate functionalities for summary reports must be provided.

### 4.3 Non-functional requirements

From the literature review and similar systems study, a number of requirements were captured. These requirements are listed and described below:

#### 4.3.1 Application interfaces (APIs)

The Application Interfaces of the web service have a number of requirements which listed below.

- **URIs**
  
  In order to facilitate the use of the application interfaces, Uniform Resource Identifiers must be designed meaningfully. Their addresses must represent the resources they deal with.

- **HTTP Methods**
  
  Appropriate HTTP methods must be used with each URI. Four methods must be able to be used with each URI: GET, POST, PUT, and DELETE. Each method must represent its equivalent functionality which is shown in Table 1.

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>Equivalent operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Retrieve</td>
</tr>
<tr>
<td>POST</td>
<td>Update</td>
</tr>
<tr>
<td>PUT</td>
<td>Create</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete</td>
</tr>
</tbody>
</table>

*Table 1: HTTP methods and their equivalent operations.*

- **XML Schemas**
  
  With each xml document, an XML schema must be provided in order to allow developers to validate it.
APIs Documentation
In order to allow developers to know how to use Application interfaces, full documentation must be provided. This documentation must describe the URI address, HTTP methods, Media types and parameters used in each API.

4.3.2 Technical requirements
The system has a number of technical requirements which listed below.

Platform
The system and all of its components must be capable of installation on any platform.

Transport protocol
Any application must be able to invoke the web service by using HTTP protocol.

Messaging
Messages that are sent to or from the web service must be in either XML or URI Encoding format.

Security
The web service must have a security mechanism to protect users’ identities and their data.

Error handlers
The web service must use error handlers to avoid unexpected errors as far as possible.

4.4 Evaluation requirements
Evaluating this type of system is a challenge. This system does not have user interfaces and users do not interact with it directly. Therefore, the usability of this system is dependent on how the client application is designed and implemented. As designing and implementing clients for this web services is outside the scope of this project, evaluating usability for end users of the system is not required.
Moreover, the target users are developers who would design and implement clients for this web service. Therefore, they should be consulted to evaluate the documentation of this web service.

To sum up, the evaluation requirements are listed below:

1- The content of the system such as the functionalities provided and data stored must be evaluated by the end users, who are university staff and students.

2- The documentation of the Application Interfaces of system must be evaluated by developers who would design and implement clients for the system.

Chapter 5 – choice of tools and protocols

In this chapter, the Authentication protocols which could be used by this project are compared and discussed in order to choose a suitable protocol. Then, the tools and the technologies which were used in this project are listed and discussed.

5.1 Authentication protocol

In order to enforce users’ authentication in the system, there a number of authentication protocols which can be used. According to (Burke, 2009) there are three popular protocols which are supported by java servlet containers. These protocols are Basic authentication, Digest authentication, and authentication using X.509 certificates, and they vary in terms of complexity and reliability. These protocols are described briefly and discussed as follows.

1- Basic Authentication

This protocol requires clients to inject authentication information inside a request header and send it to the server. The authentication information is encoded as Base64 and the title of the header is Authentication. The server reads the authentication information in the Authentication header and verifies the user’s identity.
2- **Digest Authentication**

In this protocol, a set of secure MD5 hashes of authentication information is exchanged between clients and the server. Like Basic Authentication, clients inject their authentication information in the Authorization header but the header is encrypted by using the MD5 algorithm. Again, the server decrypted the header and verifies the users’ identity.

3- **Client Certificate Authentication**

This protocol uses SSL protocol not just for securing the connection between clients and the server, but also for users’ authentication. In other words, a client who wants to communicate with a server that uses SSL protocol needs to have a digitally signed certificate that is generated by the server, and the server demands the same certificate from all of its clients. This approach is only used to secure the communication between clients and the server. However, in order to verify users’ authentications, each user must have a unique certificate.

These protocols vary in terms of complexity and reliability. In terms of complexity, the simplest protocol among them is the Basic Authentication. Clients just need to send authentication information in the Authorization header. The Digest Authentication protocol is more complex than the Basic Authentication. Clients need to encrypt their authentication information using the MD5 algorithm and send them within the Authorization header. Client Certificate Authentication is the most complex protocol among them. Servers need to generate unique certificates for each user even if they use the same client or different clients.

In terms of reliability, the most reliable protocol is Client Certificate Authentication, followed by Digest Authentication and Basic Authentication respectively. It seems that the Basic Authentication protocol is less reliable than the others. However, using this protocol with the SSL communication protocol can make the authentication mechanism more reliable.
Therefore, Basic Authentication was chosen as the authentication protocol for the system because it does not increase the complexity of the clients.

Moreover, RESTful web services usually communicate in a stateless method. The state of each request is lost after the request is performed. This means clients need to provide authentication information in every request they perform. Therefore, it is necessary that the system knows what authentication information to expect with what request.

In order to address this issue, authentication information such as user name and password can be sent within every request. However, the server will require the clients to use a secure communication protocol such as SSL on every request and this may have a negative effect on the speed of communication. Another solution would be for clients to send the same authentication information (username and password), to which the server responds with a unique key. Once this is done, clients can use these keys on the rest of their requests. This solution requires that clients use a secure communication protocol only on their first request and not on every request. Therefore, this approach was chosen to provide a secure method of communication. Figure 15 shows a diagram of the Authentication mechanism used in the web service.

![Figure 15: the Authentication mechanism used in the web service](image-url)
5.2 Tools and technologies

There were a number of aspects that support the decision in choosing the appropriate tools and technologies in this project. One aspect which was driven from the system requirements is that the technologies used in this project must be a platform independent. In other words, the database management system, the web container and the application itself must be able to be installed on any platform. In addition, there are a number of popular tools and technologies available in the market that are open source and have free licenses. Therefore, they can be used without costing the project or the future clients extra money. The tools and technologies used in this project are described below.

Database Management System (DBMS)

For the DBMS, one of the popular DBMSs is the MySQL. This DBMS can be installed on most of the platforms such as Windows, Mac OS and Linex. In addition, it has a free license so MySQL is chosen as Database Management System for this project.

Web service programming language.

Web services can be implemented by using many programming languages. In addition, JAVA is platform independent and most of the IDEs used to support its implementation have free licences. Moreover, Java provides a number of Application Interfaces that can support the implementation of web services. Therefore, Java was chosen as the programming language of the web service.

Web container.

As java is the programming language of the web service, there are a number of web containers which have free licence and can hold applications implemented in java. The available web contains were Apache and Glass-fish. Both of them can be used as containers for the web service but it seems that apache needs extra java libraries and more configuration effort than glass-fish. Therefore, glass-fish was chosen as the web container of the web service.
Java APIs for web services

There are a number of Java Application Interfaces that support the implementation of web services. One of these APIs is the Java API for XML Web Services (JAX-WS) which can support SOAP Web services and RESTful Web services. As this JAX-WS was developed especially, the implementation of RESTful Web services needs more effort and low level coding. Another API is Java API for RESTful Web Services (JAX-RS) which recently developed to support the implementation of RESTful Web services. JAX-RS provides a wide range of annotations to facilitate the implementation of RESTful Web services. Therefore, as the developed system is a RESTful Web service, JAX-RS was selected to support the implementation.

Java API for XML mapping

In order to map XML document to and from a Java object, a significant effort is needed. For this purpose, Java Architecture for XML Binding (JAXB) was developed. JAXB is a Java API for mapping java objects to XML document and the inverse. This API was used to facilitate the implementation of the web service.

Java IDEs

In order to facilitate the implementation of java applications, a number of Integrated Development Environments (IDEs) can be used. The available IDEs are Eclipse and Netbeans which have free license. Netbeans was more familiar to me than Eclipse and also provides several plug ins that can facilitate the implementation of the web service. Therefore, Netbeans was chosen as the IDE by which the web service was implemented.

Testing tools

In order to perform unit test for a java application, JUnit classes can be used. They can perform automotive tests to each method and class in the application. However, it cannot be used to test java web applications because these applications are placed in web containers and can only run by them. Therefore, JUnit classes were not used to test the web service.
In addition, the testing environment was developed by using different tools and technologies in order to test the interoperability of the web services. These tools and technologies were used in previous projects so they already installed and configured.

Figure 16 shows the architecture of the developed system and the testing enjoinment.
Chapter 6 - Database design and implementation

The design process of the database was performed in four stages. The first stage was to produce a high level entity relationship (ER) diagram derived from the initial requirements. The second stage was to correct the ER diagram by adding additional entities in order to eliminate many-to-many relationships. The third stage was to identify the needed attributes with their data types in each entity and to implement the database. Finally, the fourth stage consisted of testing the database. These stages are described as follows:

6.1 The initial Entity Relationship Diagram.

The first ER diagram was needed to show the structure of database tables in a high level view. In order to draw the first ER diagram, the initial functional requirements need to be analysed to identify the main database entities. Figure 2 shows the first ER Diagram. Figure 17 shows the initial ER diagram.

![Diagram](image)

Figure 17: the initial ER diagram.
As the first ER diagram was derived from the initial functional requirements, the analyses of these requirements are described below:

**RQ1 – School registration**

It is required that the system must provide registration functionalities for registering Modules, Tutors and Students in the system. In addition, functionalities for enrolling Tutors and Students in Modules are also required. Therefore, three entities are needed that can store this information: these are Module, Tutor and Student.

As a tutor can teach one or more modules and a module can be taught by one or more teachers, the relationship between the entities Tutor and Module is Many-To-Many. In addition, a student can be enrolled in one or more modules and a module can have one or more students. Therefore, the relationship between the entities Module and Student is also Many-To-Many.

**RQ2- Log in and Account Management**

This requirement states that the system must provide a mechanism for controlling user access and managing users’ accounts. User accounts consist of three types: Administrator, Tutor, and Student. As Administrators of the system act only as the registrars and the monitors for the system, there is no need to add another entity that stores detailed information about them. However, there is a need to store their account information in order to provide them with access to the system. Therefore another entity which can store user accounts information was needed: this entity is Account.

Each tutor must have only one user account and a user account can only be owned by one tutor. Therefore the relationship between the entities Tutor and Account is one-to-one. In addition, each student must have one user account and a user account can be owned by one student. Thus the relationship between Student and Account is also one-to-one.
RQ3- Assignment management

It is also required that the system must provide functionalities for management of assignments. For example, functionalities such as adding an assignment to a module and removing an assignment from a module must be provided, so an Assignment entity is needed.

Each module can have one or more assignments, but an assignment must belong to only one module. Therefore, the relationship between the entities Module and Assignment is one-to-many.

RQ4- Group organization

This requirement states that the system must provide functionalities for organising students into groups. Students who are not in a group must have an option to allow them to join existing groups. In addition, students who have joined a group must have the option to invite other students to join the group. Therefore two entities were added: these are Group and Join_Request.

Each student group must belong to only one assignment, but an assignment can have one or more student groups. Therefore, the relationship between the entities Assignment and Group is one-to-many. In addition, a student can join only one group in each assignment that he/she has, but a group can have one or more students. This means that the relationship between the entities Student and Group is many-to-many. Moreover, a student who is not in a group can have one or more join requests, but a join request must belong to only one student. Also, a group can have one or more join requests, but a join request must belong to only one group. Therefore, the relationship between the entities Student and Join_Request are one-to-many and the relationship between the entities Group and Join_Request is also one-to-many.

RQ5- Work management

As work management requirements, student groups are expected to perform a number of tasks during an assignment period, and they can also schedule a number of meetings. Therefore, two entities were added: these are Task and Meeting.
Each task can belong to one or more students, but must belong to only one group. In addition, a group can have one or more tasks and a student can also have one or more tasks. Thus the relationship between the entities Group and Task is one-to-many and the relationship between the entities Student and Task is many-to-many.

RQ6- Work evaluation
It also is required that system must provide functionalities for work evaluation. In other words, students can evaluate other students’ contribution to their groups and also their contribution on each task they work on. Thus, the entity Rate needs to be added to the other entities.

A rate must be created by only one student, but a student can have one or more rates. In addition, a student contribution can be evaluated by one or more rates, but each rate can belong to only one student. Therefore, there are two relationships between the entities Student and Rate, both of which are one-to-many. Moreover, a rate must belong to only one group, but a group can have one or more rates. Also, a rate can evaluate only one task but a task can be evaluated by one or more rates. Thus, the relationship between the entities Group and Rate is one-to-many and the relationship between the entities Task and Rate is also one-to-many.

RQ7- summary reports
To comply with the requirement for work summary reports, the system is required to produce a number of reports which summarise groups’ work during an assignment’s period. These reports are expected to be generated while the system is running, and there is no need to store them in the database. Therefore, there is no need to add entities that store information about these reports.
6.2 ER Diagram modification

By looking at the initial ER diagram, it can be seen that there are five many-to-many relationships. These relationships need to be broken down by adding additional entities. Therefore, five more entities were added to the ER diagram. The new entities and the modified ER diagram can be seen in Figure 18.

![ER Diagram after modifications](image)

**Figure 18**: the ER diagram after the modifications.

6.3 Adding table attributes and database implementation

The objective of this stage is to add the appropriate table attributes with their data types and to produce the final ER Diagram. Figure 19 shows the final ER diagram.
Figure 19: The final ER diagram

By looking at Figure 19, it can be seen that various data types are used for table attributes. First, most of the primary keys of the tables are INT(11) and have Auto Increment functionality. This means that when a row is inserted in a table the database will generate a unique integer as the primary key of this row. This functionality reduces the effort required to implement functions in the application that produce unique keys. However, there are other tables such as Student and Tutors in which the primary keys are VARCHAR(11) and do not have Auto Increment functionalities. Therefore, the primary keys of these tables will be provided by the application and not by the database. The reason for this is that the data input to these tables is expected to exist already in university databases and the user may want to use the same primary
keys as this data. Therefore, the primary keys of the tables Module, Student, Tutor and Assignment are \texttt{VARCHAR(11)} and will not be generated by the database.

Moreover, SQL Create statements for the database are provided in Appendix B.

\textbf{6.4 Database testing.}

The purpose of testing the database before implementing the application is to reduce the number of errors that may arise in the application tests. Therefore, all of the tables were tested by executing a number of SQL statements.

During the test, a number of errors were raised, most of which were relationship errors. For example, some of the relationships between tables were modified by adding constraints to their foreign keys such as “ON UPDATE CASCADE” and “ON DELETE CASCADE”. This means that when a reference to a foreign key is updated or deleted, the foreign key will be updated or the whole row will be deleted. This was necessary for some tables such as Module. For example, when a row of Module is deleted, all of the rows in other tables that are referenced by this row will be deleted as well.

Thus, testing the database before implementing the application was very useful and avoided a large number of errors that may have arisen later in the development of the project.
Chapter 7- The Development Increments

In this chapter, the development increments and all of their phases are described. First, the initial requirements are assigned to a number of development Increments each of which has a number of phases. The phases are further requirements analysis, design, implementation and testing. The assigned requirements of each development increment can be seen in Table 2.

<table>
<thead>
<tr>
<th>Increment Id</th>
<th>Requirements Ids</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>RQ1</td>
</tr>
<tr>
<td>Two</td>
<td>RQ2, RQ3</td>
</tr>
<tr>
<td>three</td>
<td>RQ4</td>
</tr>
<tr>
<td>four</td>
<td>RQ5</td>
</tr>
<tr>
<td>five</td>
<td>RQ6, RQ7</td>
</tr>
</tbody>
</table>

Table 2: the development increments and their assigned requirements.

In the following, the phases of the development increments are described.

7.1 Further requirements analysis

In this phase, a further requirements analysis was performed for each development increment. In the following, a detail description for the requirement analysis is provided.

7.1.1 Increment one

During the first development increment, the initial functional requirement RQ1 had a further analysis. The analysis of this requirement outputted a number of sub-requirements. These requirements are described as follows.

RQ1 - Users Authentication and Accounts Management.

This requirement has three sub- requirements which are Users Authentication, Users Authority and Account Managements. These sub requirements are described as follow:
**RQ1.1 - Users Authentication** *(high priority)*

The system must provide a mechanism for verifying users’ identity. This means users are required to provide authentication information such as username and password in order to claim a key. Then, they can use this key to perform the rest of the operations.

**RQ1.2 - Users Authority** *(high priority)*

Users must be categorized in three groups of roles each of which provide them a level of permissions. First, Admin role have the highest level of permissions and can be assigned to university registrars. Then, Tutor role provides users less level of permissions and can be assigned to university teachers. The third role is Student which has the lowest level of permissions and can be assigned to university students.

**RQ1.3 - Account Managements** *(high priority)*

Functionalities for accounts management must be provided. These functionalities must allow users to retrieve or update their authentication information such as username and password.

The use case diagram for the required functionalities of increment one can be seen in Figure 20. In addition, the textual descriptions of the use case diagram are provided in Appendix C.
7.1.2 Increment two

During the second development increment, the initial functional requirement RQ2 and RQ3 had a further analysis. The analysis of these requirements produced a number of sub-requirements. These requirements are described as follows.

**RQ2 – Schools Registration**

After analysing this requirement, it was divided into four sub-requirements which are Modules registration, Tutors registration, Student registration and List registration. These sub requirements are described as follow:

**RQ2.1 - Modules registration (high priority)**

The system must provide functionalities for registering Modules. This means functionalities such as Create update and Delete a Module must be provided.

**RQ2.2 - Tutors registration (high priority)**

Functionalities for registering Tutors must be provided. This includes operations such as Create, Update and Delete a Tutor. In addition, functionalities for enrolling tutors into modules must be provided as well.

**RQ2.3 - Student registration (high priority)**
Functionalities for registering Students must be provided. This involves operations such as Create, update and Delete a Student. Moreover, functionalities for enrolling Students into modules must be provided as well.

**RQ2.4 – List registration (low priority)**

The system can provide additional options for the registration methods. In other words, additional functionality such as create a list of modules at a time can be provided. This functionality can facilitate the registration process by allowing users to register list of modules instead of registering each module at a time. Therefore, Module, Student and tutor registration processes can have these options as well as student and tutors enrolments in the modules.

**RQ3 - Assignments management (high priority)**

The system must provide functionality for managing assignments such as Create, Update and Delete an Assignment.

The use case diagram for the required functionalities of increment two can be seen in Figure 21. In addition, the textual descriptions of the use case diagram are provided in Appendix C.

\[\text{Figure 21: Use case diagram for increment two.}\]
7.1.3 Increment three

In the third development increment, the initial functional requirement RQ4 had a further analysis. The analysis of this requirement produced a number of sub-requirements. These requirements are described as follows.

**RQ4 - Group organization**

Functionalities for organizing groups within an assignment must be provided. In other words, students must have a mechanism for organizing themselves in groups.

**RQ4.1 – Manual group organization (high priority)**
The system must provide functionalities for organizing student groups manually. In other words, a user (Tutor or Admin) can simply create a group and add students to it. In addition, the user can remove students from a group and also delete groups.

**RQ4.2 – Interactive group organization (medium priority)**
This requirement specifies that tutors should not be involved in group organization. The processes of organizing groups can be lift to students. This means a user (Student) can create a group and invited other student to join the group. Then, the student who receives the Invite can accept or deny it. In addition, a student who is not joined a group can request to join a group. After that, the group which receives the join request can decide whether they accept it or not.

**RQ4.3 – Automotive group organization. (low priority)**
In this requirement, the processes of organizing student groups can be performed completely by the system. In other words, a user (Tutor or Admin) needs only to perform one action to start the organization process. Then, the system can organize students into groups depending on a number of properties that are must be provided by the user.
The use case diagram for the required functionalities of increment three can be seen in Figure 22. In addition, the textual descriptions of the use case diagram are also provided in Appendix C.

![Use case diagram for increment three.](image)

**Figure 22**: Use case diagram for increment three.

### 7.1.4 Increment four

During the fourth development increment, the initial functional requirement RQ5 had a further analysis. The analysis of this requirement produced a number of sub-requirements. These requirements are described as follows.

**RQ5 - Work management**

**RQ5.1 – Tasks Management. (high priority)**

The system must provide functionalities for managing group Tasks. In other words, Functionalities such as Create, Update Delete a Task must be provided. in addition, functionalities for adding students to a task must be provided.

**RQ5.1 – Meetings Management. (high priority)**

Functionalities for managing group Meetings must be provided. For example, operations such as Create, Update Delete a Meeting must be provided. Moreover, functionalities for adding students to a Meeting must be provided.
The use case diagram for the required functionalities of increment four can be seen in Figure 23. In addition, the textual descriptions of the use case diagram are also provided in Appendix C.

![Use Case Diagram](image)

**Figure 23**: the use case diagram for increment four

### 7.1.5 Increment five

In the fifth development increment, the initial functional requirement RQ6 and RQ7 had a further analysis. The analysis of these requirements produced a number of sub-requirements. These requirements are described as follows.

**RQ6 - Work evaluation**

**RQ6.1** – Tasks evaluation. *(high priority)*

When a student is assigned a task during an assignment period, his or her work on the task must be evaluated by other group members. In addition, when a number of students participate in the same task, each of them must be evaluated on their contribution to the task.

**RQ6.2** – Students evaluation. *(High priority)*

When an assignment deadline passes, students who work together in the same group must have functionality for evaluating the contribution of each group
member. In other words, this is an evaluation of students’ general contribution during an assignment period.

**RQ6.3– Meetings evaluation. (low priority)**

When a student attends a meeting during an assignment period, he or she should be evaluated in terms of the attendance and the contribution to the meeting.

**RQ7– Summary reports**

The system must provide summary reports describes how well a group work together on an assignment. These reports must show a number of statistics that related to a student such as total tasks accomplished, total meetings attended general contribution to the group, average contraption on tasks.

The use case diagram for the required functionalities of increment five can be seen in Figure 24. In addition, the textual descriptions of the use case diagram are also provided in Appendix C.

Figure 24: Use case diagram for increment five
7.2 Design

In this section, the approaches used in the design phase of the development increments are described. First, the approach of designing the web service interfaces, then designing the XML messages which will be exchanged between Clients and the web service and also the design of error messages. After that, the approach of designing the code and how it is structured. Then some of the complex operations were designed using a number of Activity diagrams. All of these points are described below.

7.2.1 Web service interfaces

Approach

The design approach for RESTful web services is similar to the approach used in object-oriented programming. In object-oriented programming, a system is broken down into classes, each of which has its own operations and interactions with other classes. Similarly, RESTful web services can also be broken down into resources, which are equivalent to classes and also have their own methods. However, the methods of each resource are limited to a small number of uniform methods, namely GET, PUT, POST, and DELETE. In order to have more methods in a resource, sub-resources need to be identified, each of which can also have the uniform methods.

In this project, the identification of resources was based on the requirements of each development Increment. In other words, as the development of the project moved from one Increment to the next Increment, a number of new resources were designed. For example, the resources “login” and “accounts” were designed in the first increment, the “modules” resource was designed in the second increment, and so on.

Design

As a large number of resources are required from the system, all of which have a number of similarities. One of the resources and its sub-resources is described below as an example.
modules resource:

This resource contains all of the operations that relate to Modules such as the operations Retrieve, Create, Update, or Delete a Module. In addition, it contains other operations to Retrieve Assignments, Students, or Tutors that belong to a module. Figure 25 shows the structure of this resource and its sub-resources.

By looking at figure 25, it can be seen that the root resource “modules” has two methods, GET and PUT. GET is responsible for retrieving all of the modules and PUT is used to create a new module. In order to retrieve one module by providing its ID, the GET method would be needed again. However, this would mean that two GET methods would be assigned in one resource but two methods with the same name cannot be assigned to one resource. Therefore, the sub resource “{module-id}”
is needed. The name of the resource “\{module-id\}” will be replaced by the actual ID of a module. In addition, by using the resource “students” with the method GET, all of the students who are enrolled in this module will be retrieved; the same can be applied to the resources “assignments” and “tutors”. The URI addresses for the resource modules and its sub-resources can be seen in Figure 26.

```
/modules
/modules/{module-id}
/modules/{module-id}/assignments
/modules/{module-id}/students
/modules/{module-id}/tutors
```

Figure 26: a list of URIs for modules resource

In addition, the full list of the designed resource and their descriptions can be seen in Appendix D.

### 7.2.2 XML messages

Most of the message payload between the web service and its clients will be generated in XML format. In addition, these XML documents must have XML Schemas in order to identify the necessary elements and their data types in each XML document. Therefore, developers who intend to program clients for this web service can understand the structure and the data types of these messages.

During the development increments, fifteen xml documents and their xml schemas were designed. Some of these XML documents and their schemas are described below:

`login-info.xml`

This XML document will be sent to users who have successfully performed the login operation. It has a root element which is “login-info” and three sub-elements which are “user-id”, “account-type” and “key”. The “user-id” element of it contains the ID of the user regardless of whether the user is a student, tutor or administrator. The account type of the user can be found in the “account-type” element. The third
element is “key” which contains a key that must be attached with each request a user performs. Figure 12 shows a tree of this XML document.

![Figure 27: a tree diagram for the XML document "login-info".](image)

**assignment.xml**

This XML document contains the data associated with an assignment. Its root element is “assignment” and it has nine sub-elements. Figure 13 illustrates the tree of this XML document.

![Figure 28: a tree diagram for the XML document "assignment".](image)

In addition, the xml schemas of these xml documents and the others which were developed during the development increments can be seen in Appendix E.

### 7.2.3 HTTP responses and error messages.

HTTP protocol provides for a number of Uniform responses that can be performed by the server. The type of response can be identified from the status code of each response. Table 3 shows the common status codes used in HTTP responses.
<table>
<thead>
<tr>
<th>status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
</tr>
<tr>
<td>204</td>
<td>No content</td>
</tr>
<tr>
<td>400</td>
<td>Bad request</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
</tr>
<tr>
<td>500</td>
<td>Server error</td>
</tr>
</tbody>
</table>

Table 3: common HTTP response codes and their description

By using these status codes, clients can be notified when an operation is performed or an error occurs in the server. However, these responses do not provide details about errors, which increases the difficulties of implementing clients for this web service.

In order to address this problem, custom error messages were designed to be attached with HTTP responses when an error occurred in the server. The format of these error messages is XML, including a number of elements which identify the ID, the name and the description of the error. The description of the cause of an error will be generated when the error occurs. Table 4 shows the IDs of the errors with their names

<table>
<thead>
<tr>
<th>Error ID</th>
<th>Error Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER001</td>
<td>Login Error</td>
</tr>
<tr>
<td>ER002</td>
<td>Authorization Error</td>
</tr>
<tr>
<td>ER101</td>
<td>Not Created Error</td>
</tr>
<tr>
<td>ER102</td>
<td>Module is not created</td>
</tr>
<tr>
<td>ER103</td>
<td>Not Updated Error</td>
</tr>
<tr>
<td>ER104</td>
<td>No Data is found</td>
</tr>
<tr>
<td>ER105</td>
<td>Delete operation is not performed</td>
</tr>
<tr>
<td>ER106</td>
<td>Student enrolment is unsuccessful</td>
</tr>
<tr>
<td>ER107</td>
<td>Operation is not allowed</td>
</tr>
<tr>
<td>ER108</td>
<td>Tutor is not Registered</td>
</tr>
<tr>
<td>ER109</td>
<td>Student is not Registered</td>
</tr>
<tr>
<td>ER110</td>
<td>Tutor is not Enrolled</td>
</tr>
<tr>
<td>ER201</td>
<td>Data Format is invalid</td>
</tr>
<tr>
<td>ER202</td>
<td>Bad Entry Error</td>
</tr>
</tbody>
</table>

Table 4: Custom error messages and their description.
7.2.4 Code structure

As the selected programming language is Java, it is possible to map the resources of the application interface to an object oriented programming structure. Therefore, four categories of classes were needed. These are listed below:

1- Classes which are able to map HTTP methods to Java methods.
2- Classes which are able to generate HTTP responses, to convert Java objects to xml documents and to convert xml documents to Java objects.
3- Classes which can connect to the database.
4- Classes which are able to be converted to xml documents.

Figure 29 shows an example of a class diagram which has the classes described above.

![Class Diagram](image.png)

Figure 29: a class diagram for the resource “modules”
During the development increments, a number of classes were added to the design to give a total of fifty classes. Figure 30 shows some of these classes and their packages.

In addition, the relationships between these classes can be seen in Appendix F.
7.2.5 Activity diagrams

In order to facilitate the implementation of some of the required operations, a number of activity diagrams were drawn. These activity diagrams were drawn during the design of each development increment. In the following sections, a number of the required operations and their activity diagrams are described.

Login operation

In order to permit users to perform the web service functionalities, users must claim a key which must be appended with all their requests. This key is a combination of their user account number and a random long integer. Figure 31 shows the format of the key.

![Figure 31: the format of the key which used for users' Authentication](image)

When a username and a password are sent to the server, the server will verify them and if they are valid, the server will generate a key and store the current date and time. The current date and time are stored to keep track of users’ sessions which will be update whenever they perform an operation. The activity diagram of this operation can be seen in Figure 32 (on the next page).
Create a group
This operation allows users to create a group to which students can be added. In order to perform this operation, a number of points must be validated based on user roles. These points are listed below

1- User session.
   Each user’s session must be validated by verifying the key provided with the request.
2- User entries.
   User entries such as “Assignment ID” must be validated. In other words, the system must validate whether the assignment which has the provided ID exists or not.
3- User roles.
   The role of the user must be identified by extracting the provided key. In addition, the system must perform other validation based on the role of the user. User roles and their validations are listed below:
Admin and Tutor roles

Users who have Admin or Tutor roles are expected to provide a student ID which represents the manager of the group. Therefore, further points must be verified for the provided ID. These points are listed below:

A. Whether or not the student exists.
B. Whether or not the student is enrolled in this module.
C. Whether or not the student has already joined a group.

Student role

If the role of the user is Student, he or she will be the manager of the group. Therefore, a number of aspects must be validated which are listed below:

A. Whether or not the user is enrolled in this module.
B. Whether or not the user has already joined a group.
C. Whether or not the group allocation deadline has passed.

The activity diagram of this operation can be seen in figure 33.
Create a join request

This operation is based on the requirement RQ3.2 which has been described earlier in this report. In order to perform this operation, a number of validation points must be performed. These points are listed below.

1- User’s session.
2- User’s entries.
3- User’s role.

   This operation is only permitted to users who have Student role.
4- Whether or not the user is enrolled in this module.
5- Whether or not the user has already joined a group.
6- Whether or not the user has already made a join request to the group.

The activity diagram of this operation can be seen in figure 34.
**Response to join request**

When a student has requested to join a group, the manager of the group must decide whether or not he or she accepts the request. In addition, when the group manager accepts a request, the student who made this request must join the group. In order to perform this operation, a number of points must be validated which are listed below:

1. User’s session.
2. User’s entries.
3. User’s role.
   
   This operation is only permitted to users who have Student role.
4. Whether or not the user is the manager of the group.
5. Whether or not the student (who made the join request) has already joined a group.

The activity diagram of this operation can be seen in Figure 35 (on the next page).
Figure 35: Activity diagram for "Response to a join request" operation
7.3 Implementation

In this section, the key methods used in the implementation phase of each development increment are described. First, the approach used to connect the application with the database. Then, the method used to convert Java objects into XML documents. After that, implementing web service interfaces and how they are mapped to HTTP URIs and methods. Then, error handlers and how are they mapped to HTTP responses. All of these methods are described in the following.

7.3.1 Database connection

For connecting the application to the database, an API called Java Database Connectivity (JDBC) can be used. This API provides methods for performing SQL statements in the database. In addition, when a client needs to connect to the database, a new connection will be created. This connection can only be used by one client and it will be closed after the client finishes the transaction. Therefore, it would not be an issue when there is only a single client, or a few clients, performing transactions at the same time. However, when the number of simultaneous clients increases, the number of database connections would also increase, which would create a considerable overhead for the server. Therefore, the number of database connections is unpredictable and hard to manage using this mechanism.

In order to address this issue, connection pooling can be used. By using this approach, a client can request a connection from the connection pool instead of creating a new connection. The connection pool contains a number of pre-created connections which can be managed. Figure 36 shows a sample code for connecting the web service with the database using connection pooling approach.
7.3.2 Converting Java objects to XML documents

In order to provide the ability for Java objects to be mapped into XML documents, the Application Interface JAXB (which is described earlier in this report) can be used. Figure 22 shows a Java object which can be mapped to an XML document.
By looking at figure 37, it can be seen that the root element “module” is defined by using the annotation @XmlRootElement. The sub elements “id”, “name” and “description” are defined by using the annotation @XmlElement. In addition, the annotation @XmlType can be used to define the order of the elements. Therefore, the Application Interface JAXB provides various annotations in order to map Java objects to XML documents.

### 7.3.3 Implementing the web service interfaces

In order to implement the interfaces of the web service, the Application Interface JAX-RS is used. This API provides several annotations that allow clients to interact with the web service. Figure 38 shows an example of how the interfaces of the web service are implemented.
By looking at figure 38, it can be seen that various JAX-RS annotations are used. These annotations are described below:

- **@GET, @POST, @PUT and @DELETE** specifies the HTTP method used in each function.
- **@PATH** identifies the URI address of each method.
- **@Consumes** specifies the Media type of the income message such as XML or Text. This annotation matches the “Content-Type” header of an HTTP request.
- **@Produces** specifies the outcome message from the functions. This annotation matches the “Accept” header of an HTTP request.
- **@HeaderParam** and **@PathParam** are used to identify the location and name of the parameters.
7.3.4 Mapping Java Exceptions to HTTP responses

In order to avoid unexpected behaviour of the system, error handlers can be used. For example, clients can identify the type of errors occurring in the server by implementing error handlers. These error handlers work best when the system is implemented using only one programming language. However, the system developed in this project is expected to interact with clients that may be implemented in different programming languages. In this case, clients cannot implement error handlers and they may not be notified properly when an error occurs in the server.

In this project, the HTTP protocol is used as a mechanism for transporting messages between clients and the server. This protocol provides a status code for each response the server performs. The status code identifies whether the response is a success or an error message. Therefore, custom HTTP responses are used in this project to notify clients when an error occurs in the server.

In order to produce custom HTTP responses, the Application Interface JAX-RS has a class for this purpose. The class is WebApplicationException which provides a method to generate custom HTTP responses. Figure 24 shows how a Java exception can be mapped to a HTTP response.
By looking at figure 24, it can be seen that in the class AccountManager the Java exception “UnAuthorizedException” is thrown when the provided username or password is not valid. This exception is then handled by the class LoginResourceImp and it throws the exception “webApplicationException”. After that, webApplicationException produces a custom HTTP response with a status code (401) and a custom error message.
7.4 Testing

During the development increments, all of the implemented functionalities were tested. As the web service developed in this project does not have user interfaces, users cannot test the web service directly. They need external applications that interact with the web service and provide user interfaces. Therefore, there was a need to develop a client application that enables the web service to be tested.

Client application

The client application contains a number of web pages which were developed by using ASP.Net scripts and VB.Net programming language. Each web page interacts with one resource of the web service. Figure 40 shows a web page which interacts with the “login” resource. By using this resource with HTTP method POST, users can log in to the web service.

![Figure 40: a screen shot from a web page used to test the login resource.](image-url)
Figure 41 shows another web page which interacts with the “\groups\{group-id\}\task” resource. By using this resource with HTTP method PUT, users can add tasks to the group ID provided.

By looking at figures 40 and 41, it can be seen that the testing web pages shows detailed information about requests sent to the web service and also about the web service’s responses. These web pages were helpful in validating the operations of the web service.

Moreover, the testing web pages were implemented using different programming language and technology to the web service. This approach verified the platform independence of the web service, and verified that the web service can be invoked by any client that is able to using HTTP protocol and XML messages.
Chapter 8 – Evaluation

This chapter discusses the evaluation plan of the project. The surveys conducted for the evaluation are then described and discussed. In addition, the evaluation and the result are discussed in this chapter.

8.1 Evaluation Plan

The evaluation requirements state that the system must be evaluated with regard to two aspects. First, the content of the system such as the functionalities provided and data stored must be evaluated by the end users, who are university staff and students. Second, the documentation of the web service interfaces must be evaluated by developers who would be expected to design and implement clients for such a web service.

Therefore, the evaluation plan is to conduct interviews with university staff and students. In these interviews, the functionalities of the web service should be described. The information that is stored by the web service should be discussed by describing the tables and their columns of the database. Moreover, further interviews with developers who are experts in web services development should be conducted. As it may be difficult to find expert developers in a university context, Computer Science students who have a fair knowledge about web services or web development could be interviewed as well.

8.1.1 The content of the web service

During the evaluation phase of this project, nine interviews were conducted with two groups of target users, university staff and students. At the beginning of the interviews, all of the groups were given a general overview about the web service in terms of its objectives and its target users. However, they only had a description of the functionalities relevant to them which they were supposed to evaluate. These interviews are described as follows.
8.1.2 Interviews with students

Firstly, five of the interviews were conducted with students who were studying post-graduate programmes at Heriot-Watt University. After providing them with a general overview about the web service, they were given an oral description of the functionalities that were relevant to them. For example, groups’ organisation functionalities were described, such as how a student can join or leave a group. Work management functionalities were also discussed, such as organizing tasks and scheduling meetings. In addition, the types of data stored in the database were described, such as the data about groups, students, tasks, and meetings.

During the interviews, they were asked questions to determine their opinion about the web service. Their responses to the questions are summarized in the following. (Some of the questions asked during the interviews are provided in Appendix H)

Positive points
- Most of the interviewees liked the functionalities provided for group organisation. They liked the variety of options provided for students to join a group, as a student can either be invited by a group or can request to join a group.

- The majority of the interviewees liked the functionalities provided for work management. They also liked the data stored for interviews and tasks.

- Most of the interviewees liked the mechanism of evaluating students

Negative points
- Some of the interviewees stated that the data stored for students in the database is not enough. In order to invite a student to join a group, more information about the student would be needed.

- Some of them said that additional functionalities were needed for work management. The mechanism of assigning tasks and meetings should be similar to that provided for group organisation: in other words, functionalities for students to
request to join a meeting, or to invite a student to attend a meeting, should be provided and the same mechanism should be applicable for tasks.

8.1.3 Interviews with teachers

Secondly, four of the interviews were conducted with teachers from different universities who were available at Heriot-Watt University. Some of them are currently PhD students who had taught a number of courses that require students to work as groups. Like the interviews with students described above, they were given a general overview of the web service. They were also given an oral description of most of the functionalities provided by the web service.

During the interviews, they were asked questions to determine their opinion about the web service. The interesting points resulting from the interviews are described as follows. (Some of the questions asked during the interviews are provided in Appendix H)

Positive points
- The majority of the interviewees liked most of the provided functionalities by the web service. They liked the options for school registration, such as registering a single Module or a list of Modules.

- Similarly to the students’ response described above, they liked the functionalities provided for group organisation and work management.

- Most of them liked the mechanism of student evaluation.

Negative points
- One of the interviewees did not like the functionalities provided for school registration in general. The teacher said that these functionalities are not new and are available in most of the universities as part of their existing systems.

- Some of the interviewees stated that the information provided in summary reports is not enough.
8.2 The documentation of the web service

As the documentation of the web service is required to be evaluated, more interviews were conducted. As it was difficult to conduct interviews with expert web service developers, the plan was altered to conduct them with Computer Science students. Therefore, there were three interviews with students who study Computer Science at Heriot-Watt University.

In the interviews, they were given an overview of the web service, an overview of its objectives and its target users, and a description of its functionalities. In addition, they tried the client application which was used for testing the web services. After that, they were provided with the documentation of the web service in order to express their opinion about it (the documentation is provided in Appendix G).

As a result, the interviews provided a number of positive and negative points about the documentation. These points are listed as follows.

Positive points
- They agreed that the documentation have enough details about the URIs and the HTTP methods of each resource.
- They liked the description of user permissions for each resource.
- Most of them liked the description of the error messages for each resource.
- In general, they agreed that the information provided is enough for developing a client for the web service.

Negative points
There was only one negative point raised about the documentation of the web service. All of the interviewees agreed that the documentation needed a brief description of HTTP URIs, Methods and Media Types; they said it would provide them with a better understanding of the web service.
8.3 Evaluation discussion

With regard to the results of the interviews, it can be seen that the web service in general meets its objectives in terms of the provided functionalities and the stored information. In addition, the documentation of the web service was well written and provides enough information for the development of client applications for the web service.

However, some of the interviewees could not gain a clear idea about the functionalities the web service provides as the web service does not have User Interfaces. This was observed when they attempted to evaluate the content of the web service. Therefore, in my opinion, they were unable to evaluate the content of the web service fully.

Moreover, it was not possible to find expert web services developers. The developers who attempted to evaluate the documentation of the web service did not have a great deal of practical experience. They had a good level of knowledge about web development but less experience with web services. Therefore, it may be the case that the documentation of the web service was not well evaluated.

As my own evaluation of the web service, this web service has a number of negative and positive aspects. As a negative aspect, the web service has a large number of functionalities. Having this degree of functionality increases the complexity of developing clients that invoke the web service. Therefore, it could be argued that the project should develop a web site that has the same number of functionalities instead of developing a web service. However, as the project used an incremental development process, the total number of the functionalities was not clear at the beginning of the project and it increased when the project moved from one increment to the next.

On the other hand, this project developed a web service which is completely decoupled from the User Interface. The user interfaces for this web service can be implemented using different technologies and programming languages. They can be implemented on mobile applications, web sites, and desktop applications. Therefore,
the developed web service meets the aims and objectives of this project as it can be used by different Universities and they can implement User Interfaces that meet their individual needs.

Another positive aspect is that all of the functionalities provided by the web service are stateless. This means that each functionality works alone and does not depend on other functionalities. Therefore, the web service is scalable in terms of functionalities and its client applications can use its functionalities and ones provided by other applications together.

Chapter 9 – Summary

This chapter discusses the achievements and limitations of this project. In addition, recommendations for further research and work are also discussed.

9.1 Achievements

As a result of this project, all of the core requirements which are described in the initial chapters of this report have been achieved. In other words, this project produced a web service which meets the system requirements and has the following characteristics:

- The technologies used in developing the web service allow it to be installed on most platforms, including Microsoft Windows, Mac OS and Linux.
- As the web service uses HTTP protocol for transporting and XML format for representing complex data types, it can be invoked by any client that is able to use HTTP protocol and parse XML messages.
- The XML messages which are sent and received by the web service are self-descriptive because of the use of XML Schemas.
- Its interfaces were developed by using meaningful URI addresses and Uniform HTTP methods. Each URI address represents a resource and each HTTP method represents an operation.
- It uses Uniform HTTP responses which represent the status of HTTP requisites. In addition, it has a number of custom error messages that provide further description about errors occurring in the server side of the system.
• It has detailed documentation which can be used as guidelines for developing clients that are able to invoke the web service.

In addition, the web service meets all of the functional requirements which have high and medium priorities and also some of the functional requirements which have low priorities. Table 5 shows the functional requirements which were achieved.

<table>
<thead>
<tr>
<th>Main Requirements</th>
<th>Sub Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1 - User Authentication and Account Management</td>
<td>RQ1.1 - User Authentication</td>
</tr>
<tr>
<td></td>
<td>RQ1.2 - Users’ Authority</td>
</tr>
<tr>
<td></td>
<td>RQ1.3 - Account Management</td>
</tr>
<tr>
<td>RQ2 – School Registration</td>
<td>RQ2.1 - Modules registration</td>
</tr>
<tr>
<td></td>
<td>RQ2.2 - Tutors registration</td>
</tr>
<tr>
<td></td>
<td>RQ2.3 - Student registration</td>
</tr>
<tr>
<td></td>
<td>RQ2.4 - List registration</td>
</tr>
<tr>
<td>RQ3 - Assignment management</td>
<td>RQ3.1 - Manual group organization</td>
</tr>
<tr>
<td></td>
<td>RQ3.2 - Interactive group organization</td>
</tr>
<tr>
<td>RQ4 - Group organization</td>
<td>RQ4.1 – Task Management.</td>
</tr>
<tr>
<td></td>
<td>RQ4.2 – Meeting Management.</td>
</tr>
<tr>
<td>RQ5 - Work management</td>
<td>RQ5.1 – Task evaluation.</td>
</tr>
<tr>
<td></td>
<td>RQ5.2 – Student evaluation.</td>
</tr>
<tr>
<td>RQ6 - Work evaluation</td>
<td>RQ6.1 – Task evaluation.</td>
</tr>
<tr>
<td>RQ7 - Summary reports</td>
<td>RQ6.2 – Student evaluation.</td>
</tr>
</tbody>
</table>

Moreover, this project produced a client application for testing the web service. This client application provides a number of web pages to test the operations of the web service.

### 9.2 Limitations

This project has a number of aspects which need to be considered. These aspects led to a number of limitations in the project. First, a considerable amount of time was spent on learning some of the technologies used in the project. For example, learning
how to use the Application Interfaces JAX-RS and JAXB cost a significant amount of time.

Second, the scale of the project was large as the web service has forty resources, each of which has a number of operations, giving a total of seventy-eight operations. In order to implement these operations, fifty Java classes were programmed. In addition, seventy-three web pages were implemented in order to test all of these operations.

As result, the time spent on learning these technologies and performing the large amount of coding affected the project negatively. Therefore, some of the functional requirements which had a low priority have not been achieved. The unachieved functional requirements are listed below.

- Automated group organization (*RQ4.3*)
- Meeting evaluation. (*RQ6.3*)

Another limitation is that the summary reports the web service produces could have included more details. It could also have produced additional reports about students and groups. However, the time was limited due to the factors mentioned above, and the reports the web service currently produces have a fair degree of detail. Therefore, functionalities for producing additional reports have not been implemented.

### 9.3 Recommendations for future work

During the project, a number of aspects were found that need to be improved. Some of these aspects are related to the web service produced by this project. Other aspects are related to RESTful web services in general.

**Toward improving the web service**

The first aspect is that the web service produced by this project has a large number of functionalities. Client applications which intend to invoke the web service should be able to call on all of these functionalities. In other words, programming a client for this web service is relatively complex. In addition, providing additional functionalities for the web service will increase the complexity of its clients, which would negatively affect the efficiency of the web service.
Therefore, instead of providing additional functionalities, further work could be done in terms of improving the web service with the objective of reducing the complexity required of its client applications. As client applications need to parse XML messages received from the web service, further effort is needed in order to present these messages in human readable format. Therefore, the presentation of the XML messages could be performed by the web service by providing Cascade Style Sheets (CSS) with each XML message. As a result, clients would only need to embed the XML messages directly into HTML pages in order to present them in human readable format, which would reduce the complexity of the clients.

Another aspect is that the summary reports which the web service produces need to include extra information and statistics about the performance of students’ group work. Therefore, further work could be done in improving these reports.

Toward improving RESTful web services in general.
A number of aspects were found which relate to RESTful web services in general. One aspect is that during this project, the resources of the web service were designed using my own approach. In other words, the design of these resources was not based on a Uniform Modelling Language. This may not be a big issue in this project, as all of the work was done by one developer, but in projects which are undertaken by more developers it could present an issue, as the design of the resources would need to be understood easily by more than one developer. Therefore, further research and work could be done toward producing a Uniform Modelling Language for designing resources.

In addition, another aspect was found in the fact that RESTful web services do not have a uniform format in terms of documentation. In contrast, JavaDoc can be used in Java programming and its format can be understood by most Java developers. Therefore, further research and work could be performed toward developing a Uniform format for documenting RESTful web services.

Furthermore, challenges and difficulties were found in evaluating the web service during this project. As the web service does not have integral User interfaces, it was necessary to evaluate it by conducting surveys with the end users. In addition, it was
It is difficult to find expert web service developers to evaluate its Application interfaces. These issues could easily arise in other projects in terms of evaluating web services. In order to address this issue, further research and work could be performed toward establishing a framework for evaluating web services.

### 9.4 Conclusion

This report described the development of a RESTful web service aimed at improving the transparency of students’ group work. The requirements of the web service were derived from a review of the relevant literature and from surveys conducted with potential end users. The style and technologies of the web service were chosen after performing research among various styles and technologies.

The development methodology of the web service was based on Incremental development processes. This methodology was selected after investigating a number of methodologies.

As there were challenges in the evaluation of the web service, it was not evaluated as the project desired. However, the evaluation shows that it meets the aims and the objectives of the project in general.

To conclude, I learned the use of a number of tools, technologies and protocols during the project. I resolved a significant number of issues, especially during the implementation. I also faced and overcame a number of challenges, such as the evaluation of the web service. As a result, this project has broadened my knowledge and skills in this subject.
References

Burke, Bill. RESTful Java with JAX-RS. USA: O'Reilly Media, Inc., 2009.


