Awarding Great British Qualification

LEVEL 5

DYNAMIC WEBSITES

Lecturer Guide

Modification History

Version	Date	Revision Description
V2.0	December 2017	Content update

Published by: NCC Education Limited, The Towers, Towers Business Park, Wilmslow Road, Didsbury, Manchester M20 2EZ, UK.

Tel: +44 (0) 161 438 6200 Fax: +44 (0) 161 438 6240 Email: info@nccedu.com
http://www.nccedu.com

CONTENTS

1. Module Overview and Objectives

This unit aims to build on existing knowledge of both databases and web design in order to build dynamic websites connected to databases using technologies such as PHP and MySQL. The unit also covers XML-based services (e.g. RSS) in order to equip the learner with a range of skills to present content dynamically on the World Wide Web.

2. Learning Outcomes and Assessment Criteria

Learning Outcomes: **The Learner will:**	**Assessment Criteria:** **The Learner can:**
1. Understand the various tools and techniques used for Web Application Development	1.1 Define and explain web applications and their functions 1.2 Identify and evaluate appropriate web application development tools for a given scenario 1.3 Identify and evaluate appropriate web application development techniques for a given scenario
2. Be able to develop data-driven websites	2.1 Design and code a web-based user interface appropriate to a given problem 2.2 Design and build a database which interacts with a web page 2.3 Create scripts to facilitate data transfer between a database and a web page 2.4 Evaluate the functionality of a database-driven website in the context of a given problem
3. Be able to apply the various tools and techniques used to build data-driven websites	3.1 Select appropriate web development tools for a given scenario 3.2 Use a development tool to develop a dynamic web solution which addresses a given scenario
4. Understand the functions of web services	4.1 Define and explain a range of web services (e.g. XML, RSS, SOAP) 4.2 Evaluate and select the optimal web service solution for a given problem 4.3 Appraise the potential business benefits of web services
5. Be able to create and deploy web services	5.1 Use one or more web services to build a dynamic website which addresses a given business problem 5.2 Evaluate a dynamic website which utilises web services in the context of business objectives

3. Syllabus

Topic No	Title	Proportion	Content
1	Introduction to The Module	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• Introduction to the unit • Web applications and their functions • Web development tools and frameworks • Client-server applications • Static -v- dynamic websites • Web service solutions ***Learning outcome 1***
2	Designing and Coding A Website	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• Considerations (colours, fonts, images, file sizes, content) • CSS3 and Semantic structure • HTML5 • Responsive design (layout, flexible images and media – dynamic resizing or CSS) • Templates (bootstrap) ***Learning outcome 2***
3	Design and Developing for Mobile Websites	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• CCS3 • Flexible layouts • Resizing and adjustments • Code to redirect mobile users • Location map • Web form ***Learning outcome 2***
4	Design and Build a Database (1)	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• PHP (source code and HTML code, creating tables, manipulating tables and querying databases) ***Learning outcome 2***

5	Design and Build A Database (2)	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• MySQL (what it is, database queries, data types and ranges, SQL statements) • Ruby (what it is, how it can be used, simple coding, basic templates, simple web application)3 ***Learning outcome 2***
6	Using Scripts (1)	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• Interactive elements (consumer suggestions, displays) • Java/jQuery (loops, arrays, arithmetic operations, strings) ***Learning outcome 2***
7	Using Scripts (2)	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• jQuery for mobile devices (HTML 5, CCS3, JavaScript and AJAX) • XML (difference with HTML and examples linked to carrying data) • JSON ***Learning outcome 2***
8	Web Development Tools	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• Cookies and Sessions • Ajax Database Development ***Learning outcome 3***
9	Mobile Application Development Integration	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• Developing mobile applications • DOM, • XSLT (content delivered to mobile devices) • API • Links to mobile applications ***Learning outcome 3***
10	Web Services	1/12 2 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• Examples – WSDL, SOAP • Streaming (RSS) • Web API further ***Learning outcome 4***
11	Building A Dynamic Website	1/12 1 hour of lectures 4 hours of laboratory sessions 1 hour of tutorials	• Consideration of security issues (cyber security/SSL and encryption) • Integration

			• Testing (google mobile and HTML code) ***Learning outcome 5***
12	Evaluating Website	1/12 1 hours of lectures 2 hours of laboratory sessions 1 hour of tutorials	• Use of web application • Functionality of data driven website • Web service solutions • Business benefits of web services ***Learning outcome 5***

4. Related National Occupational Standards

The UK National Occupational Standards describe the skills that professionals are expected to demonstrate in their jobs in order to carry them out effectively. They are developed by employers and this information can be helpful in explaining the practical skills that students have covered in this module.

Related National Occupational Standards (NOS)
Sector Subject Area: 6.1 ICT Professionals **Related NOS:** 4.7.P.1 – Prepare, under supervision, for system/solution/service design activities; 4.7.P.2 – Assist with the design of system/solution/service design; 4.7.P.3 – Monitor the progress of system/solution/service design activities; 5.1.S.2 – Initiate systems development activities; 5.3.S.3 – Manage systems development activities; 5.2.P.2 – Perform software development activities

5. Resources

Lecturer Guide:	This guide contains notes for lecturers on the organisation of each topic, and suggested use of the resources. It also contains all of the suggested exercises and model answers.
PowerPoint Slides:	These are presented for each topic for use in the lectures. They contain many examples which can be used to explain the key concepts. Handout versions of the slides are also available; it is recommended that these are distributed to students for revision purposes as it is important that students learn to take their own notes during lectures.

Student Guide:	This contains the topic overviews and all of the suggested exercises. Every student needs a copy of this and should bring it to all of the taught hours for the module.
Lecturer Code:	There is a large amount of code provided in this guide for use during the laboratory and tutorial sessions. This supplementary document makes this available in electronic format to avoid the need to re-type all of it. It is available from the NCC Education *Campus* (http://campus.nccedu.com)

5.1 Additional Software Requirements

This module requires students to have access to a server with MySQL and PHP installed. The following websites may be of assistance in acquiring the necessary software:

- http://www.mysql.com/downloads
- http://www.php.net/

6. Pedagogic Approach

Suggested Learning Hours					
Lectures:	**Tutorial:**	**Seminar:**	**Laboratory:**	**Private Study:**	**Total:**
23	12	-	23	90	150

The teacher-led time for this module is comprised of lectures, laboratory sessions, tutorials and seminars. The breakdown of the hours is also given at the start of each topic.

6.1 Lectures

Lectures are designed to start each topic and PowerPoint slides are presented for use during these sessions. Students should also be encouraged to be active during this time and to discuss and/or practice the concepts covered. Lecturers should encourage active participation wherever possible.

6.2 Tutorials

These are designed to deal with the questions arising from the lectures and private study sessions.

6.3 Laboratory Sessions

During these sessions, students are required to work through practical tutorials and various exercises. The details of these are provided in this guide and also in the Student Guide.

6.4 Private Study

In addition to the taught portion of the module, students will also be expected to undertake private study. Exercises are provided in the Student Guide for students to complete during this time.

Teachers will need to set deadlines for the completion of this work. These should ideally be before the tutorial session for each topic, when Private Study Exercises are usually reviewed.

7. Assessment

This module will be assessed by means of an assignment worth 100% of the total mark. This assessment will be based on the assessment criteria given above and students will be expected to demonstrate that they have met the module's learning outcomes. Sample assessments are available through the NCC Education Campus (http://campus.nccedu.com) for your reference.

8. Further Reading List

A selection of sources of further reading around the content of this module must be available in your Accredited Partner Centre's library. The following list provides suggestions of some suitable sources:

Ballard, P. and Moncur, M. (2008). *Sams Teach Yourself Ajax, JavaScript, and PHP All in One.* Sams.
ISBN-10: 0672329654
ISBN-13: 978-0672329654

Chisholm, W. and May, M. (2008). *Universal Design for Web Applications: Web Applications that Reach Everyone.* O'Reilly Media.
ISBN-10: 0596518730
ISBN-13: 978-0596518738

DuBois, P. (2006). *MySQL Cookbook.* O'Reilly Media.
ISBN-10: 059652708X
ISBN-13: 978-0596527082

Duckett, J. (2014). *Web Design with HTML, CSS, JavaScript and jQuery Set*. John Wiley & Sons.
ISBN-10: 1118907442
ISBN-13: 978-1118907443

Nixon, R. (2014). *Learning PHP, MySQL and JavaScript: with jQuery, CSS & HTML5 (Learning Php, Mysql, Javascript, CSS & HTML5)* . O'Reilly Media.
ISBN-10: 1491918667
ISBN-13: 978-1491918661

Topic 1: Introduction to the Module

1.1 Learning Objectives

This topic provides an overview of the module. On completion of this topic, students will be able to:

- Understand the implications of web-based deployment of software;
- Understand the role of the different tools discussed during the module;
- Understand the need for N-Tier architectures when developing distributed applications

1.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practice the skills during the tutorial and laboratory sessions.

1.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

1.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- What the unit is all about
- Web applications and their functions
- Web development tools and frameworks
- Client-server applications
- Web service solutions
- Static vs dynamic websites

- Overview of terminology

1.4.1 Guidance on the Use of the Slides

Slide 4 : This sets the context of the module. It outlines what the students will learn during this module. It provides an overview of the different tools and techniques that are used for web application development such as

The module also looks at data driven websites and which allow the students to be able to create their own data driven website so that it can be easily updated to display the latest information in the most-timely manner. Data driven websites are more beneficial to a business than having a static website which is much harder to update. Data driven websites are ideal for ecommerce and ebusiness as it allows for example, prices to be updated quickly and easily.

Students will learn how to apply different tools and techniques to build data-driven websites such as database management systems and being able to track visitors and what they click on during a visit to your website.

Students will be able to understand the functions of web services and how they can be used to exchange data between applications or systems. An example of a web service is a short programme created using ASP.NET which can be used to convert the temperature from Fahrenheit to Celsius and vice versa.

Students will be able to build and evaluate a dynamic website using MySQL, PHP, Ajax with XML and JSON. Students will look at issues of security and cross-browser support.

Slide 5: Over the last ten years the World Wide Web has changed dramatically. Websites were static where the user sent a request for a page from the browser and then the page was displayed. If you needed to interact with a page, you would press a button to send that request to the server and then receive a further page back. This back and forth between browser and server was used to create anything including the simplest instructions.

Slide 6: Users expect more from their web pages and social media applications such as Facebook and Google Apps are entirely interactive, and update dynamically as we use them. Think of Internet shopping and banking, these couldn't be static websites. News media sites are able to refresh and bring you the latest stories as soon as they happen due to dynamic website technologies. The basic need for a client to interact with a server has not disappeared, but the increasing amount of bandwidth available as well as the ability to partially refresh data while a page is loaded has created an environment where genuine interactivity is possible.

Slide 7: Static websites are those that don't allow any interaction and are usually developed for small businesses who maybe want to advertise their service but can't take any online bookings or sales. However dynamic websites allow that real time processing and interaction and allows web pages to be updated very quickly and easily. Imagine a company selling holidays and to remain competitive they need to regularly check and update their prices. With a dynamic website, the data can easily be updated several times during the day. With a static website, this would be almost impossible as by the time one price had been changed an opportunity could have been missed.

Slide 8: Web applications are big business these days, and even desktop applications incorporate the internet into their fundamental design – they update themselves, make available web-based help and allow for embedding of URLs practically everywhere. They run in a web browser. Some examples of web applications are webmail, online retail sales and instant messaging.

Slide 9: Google has even developed an entirely web-based operating system through Chrome; it is now possible to do most every day computer activities through web applications rather than fixed desktop installations. Desktop computers still have advantages through, especially in highly interactive contexts (such as games) where the lag between the client and server is significant. Additionally, many people feel safer when they store their data locally and high-profile incidents, such as the hacking of the Sony PlayStation network, have highlighted some of the concerns people may have with regards to storing significant amounts of personal data in the cloud. However, many software suppliers are now allowing users to purchase "licence" for the product which allows them to access applications such as word processing or spreadsheet software through the cloud.

Slide 10: The realities of internet development though require us to take a particular approach to developing software. Users today expect more interactive and instant experience. Users are used to playing games and having instant responses and therefore websites need to have the same. Think how customer service has moved to online chat for example where you can talk to a "computer advisor" anytime of the day. This would not be possible if the company had a static website.

Slide 11: There are always at least two tiers in dynamic websites: the client tier (which may be as simple as parsing static HTML), and the server tier (which may be as simple as providing the web page on request). Most modern web applications have a more complicated distribution of responsibilities, and so a formal structure such as the PAD architecture shown on this slide is necessary to ensure it is sufficiently scalable. Each of the layers is served by different tools and in this module, we will only look at a small sampling of these. There are dozens of tools available at each layer, but the ones we talk about here are common and generically useful. For the user level, we will be using tools such as Ajax, XML, CSS, jQuery and JSON. For the application level, we will be using tools such as PHP and Ruby and for the data level we will be using MySQL.

Slide 12: The website consists of three components. A front-end web server which handles web page requests from users, middle dynamic content processing and a database management system which stores data and enables the website to be updated quickly, e.g. product prices.

Slide 13: Languages such as JavaScript and CSS are widely used. JavaScript can be used to make the pages interactive and bring the pages to life with animation and interaction such as having a search box or enabling the user to watch a video. Server-side languages such as PHP and RUBY are programming languages which for example allow a user to login to a website and access previous orders.

Slide 14: The presentation layer handles the user interface and nothing else. It is responsible for passing user requests onto the application level where appropriate, and formatting application data for user consumption. It may incorporate some simple levels of functionality such as input validation, but its main and most vital role is to mediate between the user and the server.

Slide 15: The application layer handles the “business logic” and covers everything we would generally think of as “the programming”. While it is possible to handle the data layer here (with bespoke data structures), in this module we focus on using a separate data layer to ensure scalability and optimal performance. The application layer is both a mediator between the presentation and the data, and the main driver for the various pieces of functionality that are associated with the application.

Slide 16: The data layer is responsible for storing and making available data to the application layer. We use MySQL for this because it has been designed around optimal and efficient access, and scales up to large amounts of data very well – much better than we would be able to do ourselves in most cases. The downside of using a Data Management System (DBMS) is that data access can be a chore as it needs us to construct the queries that drive the access.

Slide 17: Communication between layers is usually handled via a platform-independent protocol. A good example of this in action can be seen in the .NET framework, which allows for databases to be used but transparently coverts them into an XML format when needed. Having an independent communication protocol means that layers can be swapped in and out having to worry overly much about compatibility between the new tools used.

Slide 18: This slide introduces some of the specific tools that are going to be used and where they fit into the architecture. This set of tools is a very common combination, and while there are other equally valid choices, the ones that we are using are generically useful in most situations. They may not be the best fit in certain circumstances, but they will be a good fit in almost all and will allow dynamic websites to be created.

Slide 19: Ajax is a set of client-side tools for creating a rich user experience. It is handled by the client, and permits for partial updates of content as well as (with the use of external libraries such as jQuery) engaging interface flourishes. The client-side experience is just as valuable to our dynamic websites as the server side, and so we will spend a couple of topics discussing Ajax later in the module.

Slide 20: PHP is a generalised workhorse for server-side scripting. While it has numerous flaws, it is tremendously common and has flexible support for MySQL built in. Again, we will spend a couple of topics looking at this, especially focusing on how an inherently stateless protocol like HTTP can be used to create a seamless web application for users. We will also look at RUBY which is similar to PHP but it is a different kind of language which is apparently easier to write and read and more website developers are moving over to this language to use alongside their website. Some popular websites such as twitter and scribed use Ruby where Facebook and Wikipedia use PHP.

Slide 21: Our discussion for MySQL will focus primarily on its use through PHP and RUBY, rather than its use as a DBMS. It is anticipated that students at this level will be able to use standard SQL syntax to build queries, although we will introduce the specific MySQL syntax in a later lecture.

Slide 22: XML is a format that allows both the data and the information for interpreting that data to be encapsulated into a single file. XML is the basis for much of what we discuss in the later topics, especially when it comes to JavaScript, HTML and the Document Object Model (DOM). XML Ajax allows web pages to be uploaded without reloading the page.

Slide 23: The machines that provide the services are known as services and the machines that are connected to those services are known as clients. When you connect to google for example to read a page, Google is providing computers to service your request and is providing the server. Your computer as the user is not providing any services to anyone else on the internet and is therefore known as the client machine. Computer networks such as the internet are examples of distributed systems and applications.

Slide 24: Client-server architectures have many benefits – they are easily maintained, easily updated and the data stored on the server can be considered "live data", which is important when many people are using the same sets of data. It is possible to interleave data such as is that in social networking sties, if you have access to everything. However, while centralised servers can be hardened against external penetration, good faith must be assumed on the part of the operator. If your data and your privacy are being mined by the operator, storing on the central server can be a sticking point for users.

Slide 25: However, there are also disadvantages. The server is crucial to the effective functioning of the system, and if it goes down, so too do all the clients connected. Bandwidth can also be an issue, as each user is a consumer of bandwidth and not a provider. Other types of architecture are possible, but these are outside the scope of the module and students will be directed to investigate these as part of their private study tasks.

Slide 26: Web Service Solutions are software packets that can be integrated into your website to save you reinventing the wheel. An example of web service is Google maps. Imagine creating a website and showing your location through Google map? Instead of writing your own complicated code you can use the Google Map web service solution and integrate this into your web site. Google allows you to use their API picker to choose the right API for your website project. Have a look at https://developers.google.com/maps/web-services/overview for further information.

Slide 27: Having decided on an architecture, and the tools needed to implement it, evaluation becomes important. This is an issue that is complicated by the different families of browsers and the different versions of those browsers in common use. Compatibility issues such as these are dominant complication when developing web-based software.

Slide 28: Our user-base is also a consideration, and we will discuss issues of accessibility of web software in a later lecture. Luckily, much of the work we must do to ensure our web applications work properly can be automated via existing online tools – these tools will ensure our HTML is well formed, that we confirm to accessibility guidelines, and that any browser specific issues are identified.

Slide 29: To summarise this module includes a list of things we are going to need to be able to do, we have the generalised tasks of understanding the problems and our users, deciding on what kind of tools are needed and where functionality is to be distributed, developing both the client side and server-side aspects of the web application, and then evaluating our success with regards to verification and user satisfaction.

Slide 30: There are a lot of technologies discussed during this module, and a lot of work needed to make an "interesting" web application. Luckily, existing APIs such as those provided by Google and Facebook can alleviate much of our developer burden. While we will look at building our own user validation as an example, we can easily piggyback onto Google or Facebook to allow our users to use their existing accounts to access our systems.

Slide 31: Conclusion

Slides 32-33: Recap of the terminology

1.5 Laboratory Sessions

The laboratory time allocation for this topic for 2 hours.

Lecturers' Notes:

Students have copies of these activities in the Student Guide. Answers are not given in their guide.

Students should be allocated one of the four topics from Activity 1, and after they have had a chance to research the topic they should be brought together in groups of four. Each of the individuals in the group should have researched a different topic.

It is recommended that you give students an hour to research the set of topics in Activity 1, using books, journal articles and the Internet as necessary. An hour is recommended for the group discussion, building in time for groups to report back any significant insights to the rest of the class if you feel it is appropriate.

Activity 1:

Investigate one of the following concepts as discussed in the lecture:

1. Common web applications
2. Client-Server Architectures

3. Database Management Systems
4. Web service solutions

You should take detailed notes as you conduct your research and prepare to feedback what you have found out to other students. Remember that you will need to organise your notes so that you have included the most important information when you discuss these topics with other students.

Activity 2: Group Discussion

Work in small groups as directed by your tutor bringing together the information you have discovered in Activity 1, each group to produce a presentation showing:

- which web applications are preferable and why
- what a client-server architecture is and why it is important to website development
- what a database management system is, the software and how it is integrated into a website
- what web service solutions are and include 3 different examples showing how they can be used in a website

1.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

Keep a research journal or blog of important notes and concepts from the lecture. Supplement the material from each lecture with the results from your own studies as you progress through the module. These will be your own revision and research notes which will help you when you come to tackle the assessment. For each topic, your Student Guide will also contain activities suggesting some topics for you to research and summarise. Include examples and interesting slides and web pages that you encounter during your research.

For this topic, research the following topics, and investigate where they fit into the N-Tier architecture and website development discussed in the lecture.

- Ruby on Rails
- Perl
- Python

- ASP.NET
- VBScript
- Simple DB

Exercise 2: Presentation

Prepare an 8-slide presentation on the results from your research from Exercise 1 above. Your presentation should give a brief overview of each of the following topics – what they are and how they are used in website development:

Slide	**Content**
1	Ruby on Rails
2	Perl
3	Python
4	ASP.NET
5	VBScript
6	SimpleDB
7	Which you think you would use and why
8	Overall summary of your findings

Exercise 3: Revision

Review the lecture material and ensure that you are comfortable with everything discussed so far.

1.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in the guide. This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, particularly to share information with others in the class but also to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

Exercise 1: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short 8 slide presentation ready to give to the rest of the class. There is no need for this to be especially formal.

Topic 1

Students are reporting on each of the following topics – what they are and how they are used in website development:

Slide	Content
1	Ruby on Rails
2	Perl
3	Python
4	ASP.NET
5	VBScript
6	SimpleDB
7	Which you think you would use and why
8	Overall summary of your findings

Student should also bring along to the class notes to make from their peer presentations.

Topic 2: Designing and Coding A Website

2.1 Learning Objectives

This topic covers the website design considerations. On completion of this topic, students will be able to:

- Understand design principles of a website;
- Understand some of the considerations when building a website;
- Understand responsive web design;
- Understand bootstrap and how it can be used to design a website.

2.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practice the skills during the tutorial and laboratory sessions.

2.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

2.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Designing and coding a web-based user interface
- CSS3
- Semantic HTML elements
- Responsive Web Design
- Viewport and grid view

- Media Queries in Responsive Web Design
- Bootstrap

2.4.1 Guidance on the Use of the Slides

Slide 4 : This sets the context of the module. It outlines what the students will learn during this module. It provides an overview of the different tools and techniques that are used for good web design including user interactivity.

Slide 5: This slide reminds students that HTML and CSS are the main languages for displaying what the user sees when the browser loads the web page. Students may refer to w3schools.com for a quick refresh on the key components of HTML and CSS when creating a web page.

Slide 6: This slide provides students with a reminder about the text editors which can be used to create the web pages. Students should be familiar with these packages but if not they should go back and quickly have a go at creating a simple HTML web page and CSS style sheet using these editors before moving onto the rest of the lesson.

Slide 7: Good website design includes an accessible user interface which is designed on user needs is essential to the success of the website. Websites which are difficult to navigate, include lots of text or enable the user to get lost in the site become impossible to use and result in the company, particularly if it is an e-commerce website of losing sales, receiving poor feedback or customers giving up looking for items on the site and go to other more user-friendly websites. Poor use of colour, poor layout of the different items or navigation which isn't logical will result in poor user navigation. Websites which are not intuitive and cause confusion to the user are less likely to be successful.

Slide 8: Designing a user-friendly interface is essential to the success of an organisation. The designer needs to understand the user needs and they should carry out primary research with the users to find out what they want/need from the website. They should also talk to the company owners/staff to find out what they want/need from the website so that the design can incorporate different user needs and requirements. The designer will need to find out what functionality the website will need eg does it need to be responsive in design so it can be displayed on different devices or will it be an online shop and need a shopping basket, customer database and payment facilities. Research is essential at this stage before coding begins. Websites must comply with requirements and international standards such as ISO9241. Accessibility is important and includes designing a website for users who maybe use screen reader software or maybe navigate other way than using a mouse. The designer needs to consider accessibility features to ensure that the website complies with the law but also doesn't discriminate against disabled users.

Slide 9: Students are encouraged to look at good and not so good websites before designing their user interface. For example, students should have a look at Amazon website and look at the structure of the site, the colours used, the navigation features, how navigation works between different pages and some of the other features such as interactive elements that are used. Students are then encouraged to look at a not so good website such as uglytub.com and ask them to compare this with Amazon and some of the problems with it, eg navigation, layout, text and accessibility features. This could be delivered as group activity. Students should note some of the problems which they can refer to later when they come to design their own website.

Slide 10: Students are encouraged to look at the website uglytub.com (which has been nominated as a very poor website) and a review should take place identifying the poor practice from this website and how the design and features are different to those of the Amazon website, eg lots of text, mainly static site, poor image, poor use of colour, images that have not been resized accurately, links which don't always work etc. It is important that students have a good understanding of the differences between good and not so good websites and accessibility issues as ensuring that pages are designed and created with web content that is accessible to people with disability. W3.org/WAI have some good examples and rules relating to web accessibility and students should be encouraged to have a look at this (see activity below).

Slide 11: Website designers need to consider user interface before they start developing their website. There are many different factors for consideration and some of these are included. Colour needs to be consistent and should follow a house-style, for example if the company logo is lilac, what impact would this colour have on the rest of the pages? Consider how the different colours will be used by users and accessibility eg red and green and users that are colour blind. Consideration needs to be given to how the pages are laid out. How do you want users to navigate the site and if using a mobile site, how will they scroll the content? What are the key items you want users to see and how will you emphasise these? What impact does the layout have on different devices and with different users? Developers need to consider their audience, eg if the site is targeted at a younger audience, the UI considerations will be different to those who are from the older generation? What will the site have that will bring users to the website? We don't cover Google Analytics in this course, however it is something that students could be encouraged to research outside of the lessons as these are key to getting the website found by search robots.

Slide 12: Websites need to be consistent in their design and simple to use and navigate. Websites that are complicated are more likely to attract less customers and visitors to the website or their "journey" time on the website would be short lived. Software is available now that can track the customer journey on your website and we will look at this in future unit. When designing a site, it needs to guide your visits through the website. Comic Sans or Sans Serif fonts are best for websites and most popular and they also comply with some best practice accessibility guidelines for users so always choose a font size/colour that has these considerations. Think about how the design will work with those with disabilities and how users can navigate without a mouse or who use screen reading software.

Slide 13: Colour is important when using a website and developers should consider how it can be used to highlight sections of the site or words or emphasis important sections. Developers are advised to use less than 7 colours as research shows this to be the most effective.

Slide 14: Consider creating the website or design using black and white and when you are happy with it, start adding colour. Also consider the layout of the site and how the layout and colours will work together. Consider using similar colours for related items which might help the user navigate the site.

Slide 15:	This slide provides some information on the different colours and how they are perceived by the eye. This might help when considering the different colours to be used.
Slide 16:	This slide provides some information on colour perceptions and how colours are viewed by users.
Slide 17:	This slide provides further information on the use of colours and considerations during the design phase of the website.
Slide 18:	This slide provides information and examples of colour combinations that can be used on a website and some examples of colours to be avoided.
Slide 19:	This slide provides information on how text can be laid out on the website and some key considerations in terms of good practice.
Slide 20:	Research shows that website users scan rather than read everything on the page and therefore less is usually better and information and text should be provided in short sections. Include key headings which are highlighted and split up the text regularly so that headings point the user in the way you want them to use the website.
Slide 21:	This slide provides a refresher to students in terms of what is meant by a Cascading Style Sheet (CSS) and how it is used to design and create webpages. Students should already be familiar with some of the advantages of using a CSS rather than including styles in a web page and some of these are discussed here such as how it separates the elements from the web page which ultimately enables updating to be quicker and effective. CSS can also improve the accessibility of a web page as it usually ensures all formatting features are defined which could be easily missed when designing a web page. CSS features have increased over the last few years and now presentation techniques such as shadows, gradients and rounded corners can be included.
Slide 22:	This is an example of a CSS file which can be used to add presentation techniques to a web page. Students should be able to identify the different features that will be displayed, eg the background will have a colour red, heading 1 (h1) will be white in colour and left aligned and (p) paragraph (body text) will be font size 16 and will use the font family courier (which is a sans serif font). Sans serif fonts are recommended when creating web pages as these are more accessible to those with sight disabilities.
Slide 23:	Semantic HTML is part of the HTML mark-up language that is used to create web pages and it means that instead of specifically outlining a style in a web page (eg <p> paragraph tag). This type of tag is defined as a <div> tag but there is no indication as to what the content is and is an example of a non-semantic tag. Whereas for example <header>, <form> clearly defines the content, eg the reviewer of the code will know that this tag relates to a header or a form within the page. Non-semantic elements of HTML mark-up should be avoided as these tags cannot always be read or understood by older browsers and students should be encouraged to clearly define each presentation technique in their CSS or webpage to enhance the user experience.

Slide 24: This slide provides some examples of semantic htlm5 elements that could be used in a webpage. Further examples are available at w3schools.com.

Slide 25: This slide provides further examples of HTML5 element tags which can be used to define elements or split web pages into different sections. These are useful when read by screen readers and can enhance the accessibility of pages. It is good practice to separate the page into different elements and sections to enhance the user experience and ensure pages can be read by the latest browser.

Slide 26: As we saw in the first lesson, web pages have moved from static design to more responsive design which allows the web designer to provide a site that can be optimised and viewed on different devices. Nowadays users may use their phone, tablet or computer to view web pages and web sites need to be designed and created for this to happen. Using HTML and CSS allow the web creator to manipulate the content to ensure that it looks good on different devices by including codes which allow pages to be resized, shrunk or enlarged depending on the device. In the next lesson we will look at how to create a website specifically for a mobile device but here we are just looking at how creating web pages that can be responsive are essential in the development of a web site today. Without having responsive web design means that content may not be able to be viewed on different devices and companies may lose out on customers and sales.

Slide 27: Here are with some examples of web site templates that can be downloaded and edited to help them create their website. Encourage students to have a look at these to see which ones they like and feel that they would be useful when creating their own web pages.

Slide 28: Bootstrap is a form of open source software which can be used to create webpages and it was designed to scale up pages from a mobile device to a computer screen. It is particularly useful as it uses HTML, CSS and JS to build responsive mobile websites. It uses HTML5 doctype which is used by most web designers today and it just tells the web browser what version of HTML the page is written in. Bootstrap can be used to help create responsive webpages as it uses grid system and media objects to ensure that content is positioned around media on the web page. Details of how to install bootstrap and the themes are available at: http://getbootstrap.com/. It is free for students to download and use and is quick to learn.

Slide 29: Earlier in the lecture we looked at how pages would be scaled based on the browser and device, however bootstrap scales up from mobile device viewing screen to larger devices and there needs the “shrink-to-fit=no” to be set at no as you don’t want the page to be shrank. This needs to be included in the metatag at the start of the page.

Slide 30: This slide provides an example of a page written using coding linked to Bootstrap software. It includes scripts and an external style sheet. You will see that the page width is set to equal the width of the device viewing screen. This example includes a fixed container. The container class allows webpages to be created using fixed settings but which are still responsive layouts. To allow the rows on a page to be resized to fit the width of the device the row class has to be included to allow the horizontal groups of columns to wrap. The rows must be placed within a .container to allow for padding and proper alignment.

Slide 31:	Bootstrap uses the responsive grid system and it has a number of free example templates that can be used and edited free of charge. These can help students develop their understanding of the software and responsive page layout. An activity has been included for them to complete below.
Slide 32:	Conclusion
Slide 33-34:	Terminology and References

2.5 Laboratory Sessions

The laboratory time allocation for this topic for 2 hours.

Lecturers' Notes:

Students have copies of these activities in the Student Guide. Answers are not given in their guide.

Students have been asked to start to design and create a website for Andy's Autos (a fictitious business) that they will use throughout the remaining lessons. Students will develop the website adding additional features as they progress through their course.

Activity 1:

1. Create an appropriate folder and file structure for Andy's Autos.
2. Create a home page and at least one other web page using HTML with external CSS.
3. Create a CSS file which includes the following elements:
    a. Background colour
    b. Aligned content
    c. Menus (at least links to five – future pages)
    d. Animated banners
    e. Borders
    f. Page layout
    g. Font styles/families/sizes for different elements eg h1, h2, p
    h. Lists
    i. Table with border/padding etc
    j. Viewport
4. You will need to source your own text and images.

Activity 2: Group Discussion

Work in small groups as directed by your tutor bringing together the information you have discovered in Activity 1, each group to produce a presentation showing:

- The web pages that they have created
- Accessibility considerations

- How bootstrap can be used to help design and create a website.

Students should peer assess each other’s webpages considering accessibility and usability and some of the good and not so good features/elements they have identified in this lesson.

2.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers’ Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

Keep a research journal or blog of important notes and concepts from the lecture. Supplement the material from each lecture with the results from your own studies as you progress through the module. These will be your own revision and research notes which will help you when you come to tackle the assessment. For each topic, your Student Guide will also contain activities suggesting some topics for you to research and summarise. Include examples and interesting slides and web pages that you encounter during your research.

Download and install bootstrap software on your computer. Work through the templates and have a look at the code that sits behind these. Make notes on any key features you think would help you when creating your web pages.

For this topic, research responsive web design, viewport, bootstrap and key considerations when designing and creating these pages.

As an additional activity you may wish to research google analytics which can be used to track visitors to your site. This is a helpful way to monitor traffic on your site.

Exercise 2: Presentation

Prepare a 6-slide presentation on the results from your research from Exercise 1 above. Your presentation should give a brief overview of each of the following topics – what they are and how they are used in website development:

Slide	**Content**
1	What is responsive web design?
2	Advantages of responsive web design
3	What is grid view with 3 examples of grid view
4	What is viewport and why is it important in responsive web design?
5	What are the key considerations when designing and creating these pages?

6 What are the advantages of using Bootstrap when creating responsive web pages?

Exercise 3: Revision

Review the lecture material and ensure that you are comfortable with everything discussed so far.

2.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in the guide.

This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, particularly to share information with others in the class but also to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

Exercise 1: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short five minute presentation ready to give to the rest of the class. There is no need for this to be especially formal - you are simply reporting on anything interesting that you found during your research, or pointing out especially useful resources on the topic. Bring your journal along to the class so that you can make a note of anything especially useful that your classmates have mentioned. This is a knowledge dissemination exercise; you are not being formally assessed on the style or content of the presentation.

Topic 3: Designing and Developing for Mobile Websites

3.1 Learning Objectives

This topic provides an overview of producing a mobile website. On completion of the topic, students will be able to:

- Understand the key requirements to design and develop a mobile website

3.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. It is strongly advised that you implement the coded elements of the lecture in front of students so they can see a mobile website application being constructed. Slides with code upon them are good points to stop and show students how they relate to the application you are building.

They will then develop the code during the tutorial and laboratory sessions.

3.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

3.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- CCS3 for mobile websites
- Flexible layouts
- Resizing and adjustments
- Code to redirect mobile users
- Location map
- Web form

3.4.1 Guidance on the Use of the Slides

Slide 4: Traditionally websites were created purely and simply for display on desktop computers, however things have changed and now websites need to be designed and created for use through mobile devices including mobile phones, wearables and tablets. Recent research shows that more users are accessing the web via mobile devices than desktop computers and therefore it is essential that the user experience includes good practice for mobile devices. Some businesses may lose customers and users if their website is not mobile friendly.

Slide 5: A mobile website is designed and scaled to function on the device screen. In Topic 2 we introduced responsive web design and how CSS can be used to resize, hide, shrink or enlarge the content dependent on the device. We also looked at Grid View, CSS and Viewport and how these are important considerations when using mobile devices. In addition, the mobile website needs to be easy to navigate and read – especially on for example a wearable watch which would have a very small viewing screen. Mobile considerations would include navigation buttons with easy-to-read print, click to call features which means that when you touch the telephone number on the screen it would dial the number without having to enter the numbers manually. You could also include GPS and directions from current location to find the business and easy to load streamlined menus. This can be through toggle on/toggle off content which we will cover later in the topic.

Slide 6: There are lots of benefits to the business and the user of having mobile website in terms of considering the design specifically for the user experience. When designing a website, design should incorporate desktop, tablet, mobile and wearable devices. Mobile content from a website needs to be optimised quickly and easy to use. It should load quickly and allow visitors to make a purchase without having to visit the desktop site. All these considerations should be made before the site is developed. Businesses who don't create a website using responsive design are likely to lose users and customers.

Slide 7: CSS can be used to restrict the viewing page on different devices and we had a look at flexible layouts in Topic 2. In addition, JavaScript or jQuery's (Topics 6 and 7) can be used in responsive web design. Traditionally a mobile website would be viewed as one column of data in portrait design. Semantic HTML elements are considered good practice in mobile website development and have a look back at Topic 2 for revision. This shows examples of how pages can be viewed in different mobile devices depending on the code that is used to create the pages. Considerations should also be given to user input. On a desktop computer data would be input using a keyboard and mouse, however on a mobile device it is through touch screen interfaces and therefore consideration needs to be made on how data will be input and how "taps" need to be used when navigating through the site. Another option is to consider dropdown menus as these tend to work better when view on a mobile device is in one column. It is easier for users to click the links on either the left or right side rather than scrolling left to right.

Slide 8: This is an example of a basic mobile page that has been taken from w3schools.com website. This shows that viewport has been used and the width is dependent on the device that has been used. An external style sheet has been linked to the page. Get students to create this page for Zanzibar Zoo as they will create a mobile website throughout the rest of this topic.

Slide 9: This slide provides an example of the same layout but how it would be viewed on different devices. The first provides an overview of how the page could be displayed on a desktop device. By using responsive design this page can resize to fit both tablet and mobile devices. The tablet layout is very similar to the desktop but includes the Social Media Links section underneath the content, whereas the mobile layout shows everything as one column. The next few slides provides some examples of how to include this when coding a website.

Slide 10: Viewport is the name given to the visible area to a user of a web page. The visible area of a web page will be dependent on the device. Viewport is set through the <meta> tags which holds all the HTML metadata coding about a webpage. The metatag identifies in the web page the width and scale of the page. The width=device-width instruction sets the width of the page to follow the screen-width of the mobile device. The initial-scale-1.0 set the zoom level when the page is loaded by the browser. If the viewport is not set the page may not scroll horizontally or the screen may not be shown in full or it may appear too small to read. We will look more at creating pages for mobile devices specifically in Lecture 3.

Slide 11: This slide provides some considerations for web creators, for example users do not want to scroll horizontally across a page to view all the text and therefore the page width should be set against elements to ensure this does not happen. In addition, using viewpoint is helpful by setting a width against using pixels to set the width of the page as this can vary widely between different devices. CSS width elements should not be set to absolute/fixed widths and this will mean that pages cannot be resized when viewed in different devices or different browsers.

Slide 12: This slide discusses the differences between using the traditional box model to design web pages against grid view which is used in responsive web design. The box model was sometimes set using pixels which meant that pages were static and were sometimes viewed differently in different browsers affecting the browser experience. The Grid view allows the webpage to be divided into columns and has a total width of 100% which means that it can be resized or shrunk depending on the browser or device. Grid views are usually created to have 12 columns across the page.

Slide 13: When creating a page using grid view in css all the elements need to have box-sizing properties set to the border-box to include both padding and border in the overall width and height of the elements. When using the box model these were always defined separately, however the next slide shows an example of setting a responsive web page with 3 different sized columns.

Slide 14: This slide provides an example to show how to use grid view and to create a responsive web page with three columns of different sizes, eg one at 20%, one at 50% and one at 30%. This shows that they are all left aligned (float:left).

Slide 15: Media queries are used in responsive web pages and they allow rules to be defined so that css styles are dependent on the device the page is being viewed in. For example, it can consider specific characteristics linked to an output device, eg screen or printer. A media query enables the content to be adapted depending on the device it is viewed on, eg it can adapt the page settings depending whether it is viewed on a computer screen or smartphone screen. Media query checks the size of the browser window from the device and changes the background to meet the query. For example, in this example if the browser is smaller than 700 pixels then the background colour is changed to light blue. Have your students have a go at this and try it on a computer screen and then a mobile device with a small screen.

Slide 16: Remember that width property should be set to 100% for images as well as if an image is set to a specific fixed width then when it is displayed on a mobile device it won't be responsive and then less likely to be viewed properly by the user. The width must be set to 100% for all images on a web page to ensure that the image can be scaled up and down. Have a look at this example on the page. There are other examples on the next page which has a specific height set rather than auto here.

Slide 17: This page shows an image of car being inserted as a background image. It has the width set of 100% which means that it can be resized horizontally on the page, however the height has been set at 400 pixels which means that this is fixed at this size. If it was set to 100% then it may stretch to cover the content area which means it would be unproportionable or it may cover over content that needs to be visible.

Slide 18: An alternative way to resize images for mobile devices is to use an image resizer such as Co-operative image resizer and Independent image resizer. A Co-operative image resizer is an example of when HTML is included in the template and image is within the <img src> tag and the attribute points to the resized image. An independent image resizer uses Javascript to scan for <img> tags and rewrites the <img src> tag through the resizer. Alternatively, css style can be used to set the width as a percentage of the screen width, eg width: 33%; max-width: 180px;

Slide 19: In addition to page/image resizing which is dependent on the device that is being used, consideration also needs to be given to resize text. Text which is too large or not large enough to read on a mobile device is going to affect the user experience. Consideration also needs to be given to how text can be resized when the user switches their device from portrait to landscape view. @media queries are helpful here as they can resize the text dependent on the screen width. The @media rule can be used to set different style rules for different devices. They can check different things such as height and width of the viewport, height and width of the device, whether the orientation is portrait or landscape, the resolution and more. Have a look at the following website which provides examples of different media types that can be used for different devices https://www.w3schools.com/cssref/css3_pr_mediaquery.asp.

Slide 20: Another consideration for mobile device is the content which can be toggled on or toggled off. Consider a website which has been designed originally for a desktop and includes dynamic menus or web image slides which may take time to load or may not be displayed effectively in different mobile devices. Users should therefore have the option to turn on or turn off some of the content. This can be completed through using CSS or JavaScript which can edit the content through a toggle button which is created. In addition jQuery library can be used (see Topic 7) to download and edit the code for your own website using the .toggle() function.

Slide 21: Media Queries are important when designing a mobile site. The @media rule can be used for different styles to define the rules that are used for different media types and devices. Media Queries are linked to CCS3. Media queries look at the device eg screen size or screen resolution and then deliver the page to match these settings. There are a number of @media queries that can be used to help the web developer develop a mobile website.

Slide 22: This slide includes an example of Media Queries that can be used and shows how these are tested as logical expressions showing either true or false.

Slide 23: Another way to ensure that the mobile user experience is good is to ensure that the website is directed to a site for mobile users. There are various methods that can be used to do this, for example using a “user-agent” or by assessing screen resolution. A mobile browser sends data in the form of a “user-agent” to your web server. The “user-agent” tells the web server which type of device is trying to access and open the page. The device is detected and the mobile site is opened. Alternatively, the screen resolution is detected. Mobile devices tend to have higher screen resolutions than desktop computers and therefore the “user-agent” can identify the type of device and return the appropriate content.

Slide 24: There are various methods that can be used to redirect mobile users to a mobile website. Three of the most popular are shown here eg JavaScript, PHP and CSS using media queries.

Javascript code can be used to detect either screen size or the user agent. The following slide shows an example of the code that can be used to detect the screen width and then direct to the mobile site. However, consideration should be given to the use of JS and mobile devices. If the device does not use Javascript then this will not work.

Another alternative is PHP and this can work with browsers that do not rely on JavaScript. An example of the code to be used can be found at: http://wallydavid.com/simple-php-mobile-website-redirect-code - remember to replace m.site.com with your own website.

Finally, the last example is media query. CSS now has a built-in method to detect mobile browsers and displays the CSS style based on the size of the browser window. This doesn’t need a separate mobile site just two different style sheets linked to the web pages. The first style sheet would be for desktop monitors and the other would be for mobile devices. An example can be found at: https://css-tricks.com/snippets/css/media-queries-for-standard-devices/.

Slide 25: This provides an example of JavaScript which can be used to demonstrate how the page would be directed to a mobile site dependent on the screen width.

Slide 26: An important feature of a mobile website would be the inclusion of google map. There are different ways that google map can be included on a mobile site. One uses JavaScript and the second could use a static map through an API. If you are wanting to display the map on high end devices that use Javascript then Google Maps API would be used, however for low end devices then Static Map API would be chosen. The first thing is to set the properties to be detected on the device which will then render the map according to the browser property. Including the static map is really easy and all you need to do is build a URL which links to the Google Static Maps API. To use the high-end version then the Google JavaScript API is included as well as

HTML div element which will hold the map. Map options are set eg such as location using grid co-ordinates for the specific area of the map you want displayed in your web page.

Slide 27: Including and completing mobile forms are another important consideration when using mobile devices. For example, inputting data into a form on a desktop computer would be through keyboard and mouse whereas on a mobile device it would be input through the touch-screen keyboard. Consideration needs to be made in terms of ensuring that the screen is large enough for touch screen and checkboxes are easier to tap. We have already discussed tapable links to detect telephone numbers but this is also something to consider in the form. What about autocorrect and autocapitalise? Although these are sometimes useful features, on web input forms, code should be used to turn these off as they can be quite annoying for the user. Using HTML and CSS elements to style the form will make the input and control easier and better experience for mobile device users. Code can be used to make checkboxes easier to tap or make the input fields large enough for target areas on mobile device.

Slide 28: This is an example of code used to create a form for a mobile website.

Slide 29: Social media links are essential on both desktop and mobile sites. However, adding social media buttons and integration are great for mobile sites. Some users prefer to use just the one login to different websites, for example, users will use their Facebook login on shopping websites and therefore if you integrate login to your own site through a social media login, then for some their experience will be better. However, you may wish to provide this as an option as not all users will want to do to this. Including social media links allows content easily to be shared with other users and across different sites. For example, a job advert on your company website could be shared through social media by allowing users to click on the social mediate link and share. Ideally social media buttons are placed at the bottom or along the side of the page so any resizing allows them easily to be found. Facebook for example, has many social plugins (widgets) that can be used to enhance your website such as save button, like, share, send and quote, comments and follow button. Have a look at the developer's page on facebook.com to see how these plugs can be included in your own site.

Slide 30: Once the website has been created then it needs to be tested to ensure it is mobile friendly. For example, Google has a mobile-friendly test tool which can be used to test the web page and see how it would perform on a mobile device.

Slide 31: Conclusion

Slide 32: Terminology

Slide 33: References

3.5 Laboratory Sessions

The laboratory time allocation for this topic for 2 hours.

Lecturers' Notes:

Students have copies of these activities in the Student Guide. Answers are not given in their guide.

Students have been asked to start to design and create a mobile website for Andy's Autos (a fictitious business) that they will use throughout the remaining lessons. Students will develop the website adding additional features as they progress through their course. They can use the information from Topic 2 to help them here.

Activity 1:

1. Create an appropriate folder and file structure for Andy's Autos Mobile Website.
2. Create a home page and at least one other responsive web page which can be displayed on at least two different sized mobile devices.
3. Create a CSS file which includes the following elements which can be viewed on mobile devices:
    a. Background colour
    b. Menus (at least links to five – future pages)
    c. Borders
    d. Page layout
    e. Font styles/families/sizes for different elements eg h1, h2, p
    f. Viewport
4. You will need to source your own text and images.

Activity 2: Group Discussion

Work in small groups as directed by your tutor bringing together the information you have discovered in Activity 1, each group to produce a presentation showing:

- The mobile web pages that they have created

Students should peer assess each other's webpages considering accessibility and usability and some of the good and not so good features/elements they have identified in this lesson.

3.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

Keep a research journal or blog of important notes and concepts from the lecture. Supplement the material from each lecture with the results from your own studies as you progress through the module. These will be your own revision and research notes which will help you when you come to tackle the assessment. For each topic, your Student Guide will also contain activities suggesting some topics for you to research and summarise. Include examples and interesting slides and mobile web pages that you encounter during your research. Make notes on any key features you think would help you when creating your mobile web pages.

For this topic, research different codes that can be used to resize and adjust text and layout on a mobile website, inclusion of different APIs that can enhance the user experience.

Exercise 2: Presentation

Prepare a 5-slide presentation on the results from your research from Exercise 1 above. Your presentation should give a brief overview of each of the following topics – what they are and how they are used in website development:

Slide Content

1 Why create a mobile site?
2 What are the challenges when creating a mobile website?
3 How flexible were "flexible layouts"?
4 How easy was it to incorporate google maps in your mobile website?
5 What did the website look like in two different devices?

Exercise 3: Revision

Review the lecture material and ensure that you are comfortable with everything discussed so far.

3.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in the guide.

This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, particularly to share information with others in the class but also to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

Exercise 1: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short five-minute presentation ready to give to the rest of the class. There is no need for this to be especially formal - you are simply reporting on anything interesting that you found during your research, or pointing out especially useful resources on the topic. Bring your journal along to the class so that you can make a note of anything especially useful that your classmates have mentioned. This is a knowledge dissemination exercise; you are not being formally assessed on the style or content of the presentation.

Topic 4: Design and Build A Database (1)

4.1 Learning Objectives

This topic provides an overview of PHP. On completion of the topic, students will be able to:

- Create scripts to facilitate data transfer between a database and a web page
- Evaluate the functionality of a database-driven website in the context of a given problem.

4.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. It is strongly advised that you implement the coded elements of the lecture in front of students so they can see a MySQL/PHP application being constructed. Slides with code upon them are good points to stop and show students how they relate to the application you are building.

They will then develop the code during the tutorial and laboratory sessions.

4.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

4.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Introduction to PHP;
- Input and output in PHP;
- Common structures in PHP.

4.4.1 Guidance on the Use of the Slides

Slide 4: PHP is the first of the tools we are going to look at when we build our toolkits for dynamic websites. It is entirely possible to create a dynamic website using nothing but PHP, but for all the reasons outlined in the previous lecture, it is best to consider it as part of a larger framework – it fits on the application layer of the model we have previously discussed. This topic assumes certain things of all students – that they are comfortable with simple programming techniques, and that they can make use of and understand HTML mark-up. If your students come from a business IT background, you may need to spend some additional time explaining some of the basic programming to ensure that students have sufficient grounding to understand the lecture and complete the tasks.

Slide 5 and 6: The largest difference between a server-side language and a desktop language is the workflow required to write, deploy and test web applications. Integrated Development Environments (IDEs) exist for PHP, but more commonly the process is handled in separate, discrete steps – writing the code in a text editor, uploading the code to a server, and then executing the code with a browser. The exact steps that are gone through for students working through this topic will be dependent on how they are to run their PHP applications – the steps will be the same, but the exact details cannot be provided in the lecture. We do not use an IDE in this module, because it is important that the role each piece of software is playing is understood – where tools fit in the architecture is significant, and an IDE can obscure that. Students are encouraged to choose a basic editor such as Notepage++ or Jedit. These should be used instead of a word processor.

Slide 7: PHP fits into the application layer of the N-Tier architecture because it is a server-side language – all PHP code is processed on the server, and all that the client sees is the fully rendered HTML. While this does impose a certain burden on the server, it allows for a tight integration between the application and data layers in a way that is not possible with client-side scripting languages.

Slide 8: When the GET protocol is used, the data that is sent from the form elements get encoded directly into the URL. Spaces get replaced with a special code (%20), and a few other symbols are adjusted (which has importance when making use of web services). Each piece of data is delimited with an &, and the parameter list itself follows a ? in the URL. In dice_roll.get.php?num=6&faces=7, what we have is two parameters (num and faces), which have the values 6 and 7 accordingly. Our PHP scripts can make use of these variables by using the $_GET associative array – we need to know how the data is coming in, because our PHP code must either be changed to accommodate the format, or be written in such a way that it does not matter which is used.

Slide 9: This slide shows the dice roller web application using the GET method.

Slide 10: Correspondingly, this is the PHP code using $_GET to process the dice roller web application shown on slide 9. However, we can make a web application prepared for both of these GET and POST method by checking first to see if a key is available in one array, and if it is not then using the other.

Slide 11: The POST protocol is more useful on a day to day basis, because it has no limitations in terms of data size or data formats – anything that can be represented in PHP can be sent via a POST request. Unlike GET which encodes the data into the URL, POST encodes it directly into the HTTP header which obscures it from casual view.

Slide 12: This shows our first, very simple PHP script. It is embedded in an HTML document – when a web browser requests this PHP page, the server will render the PHP (the bit stored between the <?php And the ?>) and return only a plain, flat HTML file to the client.

Slide 13:	A PHP file can contain many separate blocks of PHP code, and each will be processed on the server. Some large PHP programs are made up of many interacting PHP files; we will look at some examples of these as we go through the module.
Slide 14:	This slide shows the HTML that is received by our browser – there is no hint of the PHP, and as far as the browser is concerned, there never was. There are several benefits to this, especially when regarding intellectual property. Client-side code, because it must be available to the browser, can be viewed and read by anyone who can access the HTML source. Server-side languages permit for the code to be kept secret, which can be important when talking about potentially valuable business logic.
Slide 15:	All programming requires a way to interact with users. PHP provides for this through the use of HTML for the most part, although there are other ways too (we will talk more about this when we discuss web services and APIs). For our first example, we are going to do it through the use of an HTML form element – we create a form with a couple of textboxes, and provide names to those textboxes. When the button is pressed, the contents of the textboxes will be sent to the PHP script that we define in the **action** of the form.
Slide 16:	This shows the HTML front-end of this interaction; no PHP at all is defined here. The link between our PHP and this HTML form is instantiated through the form action – there we provide the script name (we assume here that both files are stored in the same remote directory, so we do not need to worry about relative or absolute path references). We use the POST method here, which has been discussed in the Topic 3.
Slide 17-19:	Variables do exactly the same thing in PHP as they do in other languages, except that we do not worry about typing – all variables are identified with a $ symbol, and we do not need to explicitly declare them before they are used. When the button of the form element is pressed, the contents of each of the elements in that form are sent as part of the POST request to the PHP script. These get stored in a special variable called $_POST – this contains all of the data that was sent as part of the HTTP request.
Slide 20:	This slide shows the PHP code for the test variables file. In the PHP, we create a variable called $name and its contents come from this $_POST variable. Name is what we called the text in Slide 16, and so that is how we get access to it, and similarly for the question. The rendering of strings in PHP allows us to embed a variable inside a string and have it output properly. In a later slide, we will look at a second way of presenting variables in output to the user.
Slides 21-22:	PHP has a number of benefits that make it popular. Input and output are especially easy to do because it is handled via HTML. Creating a user interface in a desktop language, or a language such as ASP .NET, can be very time consuming and requires much design and thought. In PHP we offload this work onto the existing HTML protocol, which gives us access to the full range of tools that any web page has. It is also popular because it has been designed with database connectivity in mind – it is very easy to integrate a database into a PHP application as we will see

in later topics. It is also a simple language to use – much of the complexity that comes from other languages has been stripped out to create an easy syntax.

Slide 23: As with anything, there are downsides. In particular, while it is possible to work around the statelessness of HTTP, it involves a considerable burden on the developer to ensure that stateless elements of a large PHP program work well together. Architecturally, it is not especially well designed (if you want to swap data layers, you often need to actually change the underlying code statements used). Due to the simplicity of the language, it is often used by hobbyists to create large programs that are badly structured and yet freely available for exploration. Additionally, versioning differences mean that unless you know the specifics of the environment in which a PHP application is functioning, you can never know if the PHP code you find will work on your system. Finally, actually creating 'real world' applications requires the use of a second layer of software, most usually MySQL.

Slide 24: PHP is a loosely typed language, which means there is no syntactic difference between strings, ints, floats and all the other different kinds of data. They are all just held in a 'variable'. Arrays are no different, except that they are populated using a special function called **array**. In our next example, we are going to look at a PHP application that returns a random answer when we ask a question. Students may be familiar with the 'magic eight ball' toy which works in a similar way, hence the name of our program. You might like to show them a picture of one of these online and explain the general idea if they are unfamiliar with this toy.

Slide 25: Here we show the syntax for creating an array in PHP. We declare a variable ($responses) and then we populate it with the result of the array() function – in this case, a list of four strings.

Slide 26: All our application is going to do is give a random string from this array as an answer. We do not care what the question was at all, so we discard that input. As with most c-type languages, arrays in PHP start counting from 0.

Slide 27: PHP gives us a function that provides a random numerical index that is valid for the provided array. This function is called array_rand – the line of code shown will set the variable $random_response to have a random number between 0 and 3 (the valid indexes permitted by the size of the array).

Slide 28: This slide shows the program code for our magic eight ball. We setup our array as shown on Slide 25, and then we get a random numerical index as shown on Slide 27. The answer text is given by using that index to pull a value from the array. As with our first PHP example, we pull the name from our $_POST array, but we don't do anything in this instance with the question – that's just a pretext for the interaction and is not used to generate the output.

Slide 29: PHP offers the full range of loops, and syntactically these are identical to how they work in C. This slide shows the syntax for a while loop in PHP.

Slide 30: This slide shows the syntax for a for loop. Note that within the for loop, we provide the content of the array in a slight different way – rather than "the number is $i", we instead use the string concatenation operator to add the variable to the string. This is not a requirement of a for loop, it is just to show students that PHP offers multiple ways of accomplishing the goal of combining variable values and strings.

Slide 31:	Due to a lack of time to go over the details of PHP syntax, much of the exploration of the language will have to be done by students in their private study time. Students should already be familiar with the do-while loop from other languages, but the for each loop may be new depending on their previous exposure to the languages that support it.
Slide 32:	Selection in PHP similarly works as it does in other C-Type languages. In this example, we show an if structure nested within a for loop, providing output depending on the oddness or evenness of the number contained within the counter variable.
Slide 33:	PHP has a peculiarity when it comes to performing comparisons on variables, and this is a direct consequence of its loosely typed structure. The standard equivalence operator (two equals) is called the **loose comparison operator** in PHP. It attempts to perform a process called 'type juggling' whereby it tries to find a happy medium comparison value between two variables of different content types. Strict comparison operators should be used where this should not be done, and in the context of this lecture that is almost every string comparison.
Slide 34:	While PHP is a loosely typed language, it is still the case that typing is important (one of the reasons being the type juggling that PHP performs). As such, it is often the case that we need to formally switch the type of data contained within a variable – that is done using type casting, where the variable is prefaced with the name of the type it should become.
Slide 35:	We are going to be using PHP version 5 throughout this module. There are numerous differences with regards to this version and versions that have come before, and students should be mindful of this when looking at the example code that is available on the internet. Backwards compatibility is not a particularly strong feature of PHP.
Slide 36-38:	Conclusion, Terminology and References.

4.5 Laboratory Sessions

The laboratory time allocation for this topic is 2 hours.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide. Depending on the browser installed, PHP syntax errors may not be presented to users when they access the PHP pages. You should refer to the browser specific documentation to enable PHP debugging if this is the case. Not all the syntax needed for these exercises has been included in the lecture. Students should be encouraged to investigate the functions available in PHP to help solve these problems.

Making use of HTML and the appropriate PHP, write programs that accomplish the following exercises:

Exercise 1:

Show the result of rolling a set of dice, with the number of dice and the number of faces on each dice decided by the user.

Suggested Answer:

The following programs are examples only – students should be encouraged to explore and extend as necessary. The key thing they should incorporate is the use of the rand() function to generate the random numbers, although other solutions should be accepted too.

HTML

```
<html>
  <head>
    <title>Dice Form</title>
  </head>
  <body>
  <form action = "dice_roll.php" method = "post">
     <p>How many dice</p>
     <input type = "text" name = "num">
     <p>How many faces?</p>
     <input type = "text" name = "faces">

     <input type = "submit" value = "Roll">
     <input type = "reset" value = "Clear values">
  </body>
</html>
```

PHP Code

```
<html>
  <head>
    <title>Dice Roller</title>
  </head>
  <body>
  <?php
    $num = $_POST["num"];
    $faces= $_POST["faces"];
    $total = 0;
    $roll = 0;

    for ($i = 0; $i < $num; $i++) {
      $roll = $random = (rand()%$faces) + 1;
      echo "<p>Dice roll " . ($i+1) . " is $roll.</p>";
      $total += $roll;
    }

    echo "<p>Total roll is $total</p>"
  ?>
  </body>
</html>
```

Exercise 2:

Take in some user provided text, and reverse it.

Suggested Answer:

HTML

```
<html>
  <head>
    <title>Reverse Form</title>
  </head>
  <body>
  <form action = "text_reverse.php" method = "post">

     <p>Text to reverse</p>
     <input type = "text" name = "text">

     <input type = "submit" value = "Reverse">
     <input type = "reset" value = "Clear values">

  </body>
</html>
```

PHP Code

```
<html>
  <head>
    <title>Text Reverser</title>
  </head>

  <body>

  <?
    $text = $_POST["text"];
    $reversed= "";

    $text = strtolower ($text);

    for ($i = strlen ($text); $i >= 0; $i--) {
      $reversed = $reversed . $text[$i];
    }

    echo "<p>The reversed word is $reversed</p>"
  ?>
```

```
    </body>
  </html>
```

Exercise 3:

Using Andy’s Autos website, create a new webpage and add a feature which will allow the user to select one of four operations from a drop-down menu – add, subtract, multiply, or divide. Allow them to provide two numbers, and perform the appropriate operation on them.

Suggested Answer:

HTML

```
  <html>
    <head>
      <title>Calculator Form</title>
    </head>
    <body>
    <form action = "calc.php" method = "post">

       <p>Number 1</p>
       <input type = "text" name = "num1">
       <p>Number 2</p>
       <input type = "text" name = "num2">
       <p>Operation</p>
        <select name = "operation">
          <option>Add</option>
          <option>Subtract</option>
          <option>Multiply</option>
          <option>Divide</option>
        </select>

       <input type = "submit" value = "Calculate">
       <input type = "reset" value = "Clear values">

    </body>
  </html>
```

PHP Code

```
  <html>
    <head>
      <title>Calculator</title>
    </head>

    <body>
        <?
      $num1 = $_POST["num1"];
```

```
          $num2 = $_POST["num2"];
          $op = $_POST["operation"];
          $ans = 0;

          if ($op === "Add") {
              $ans = $num1 + $num2;
          }
          else if ($op === "Subtract") {
              $ans = $num1 - $num2;
          }
          else if ($op === "Multiply") {
              $ans = $num1 * $num2;
          }
          else if ($op === "Divide") {
              $ans = $num1 / $num2;
          }


          echo "<p>The answer is $ans.</p>";
        ?>

        </body>
      </html>
```

4.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

As part of your ongoing journal exercise, you should research the following topics:

- Loose typing
- PHP syntax
- PHP functions

Exercise 2: Calculator Extensions

Using your calculator code that you wrote for exercise 2 in the laboratory exercises, expand it so that it allows for the following:

- Calculates the power of one number to the other
- Calculates the remainder left after diving one number by another
- Puts the result of a calculation automatically into the first textbox when it has been done.

Suggested Answer

The first two modifications to the calculator program can be done adding some cases to the if at the start. The last of these requires students to realise that they cannot (yet) alter the content of an input box after it has reached the browser, and that their PHP page will need to generate the appropriate HTML content itself like so:

```
<html>
  <head>
    <title>Calculator</title>
  </head>
    <body>

  <?
    $num1 = $_POST["num1"];
    $num2 = $_POST["num2"];
    $op = $_POST["operation"];
    $ans = 0;

    if ($op === "Add") {
        $ans = $num1 + $num2;
    }
    else if ($op === "Subtract") {
        $ans = $num1 - $num2;
    }
    else if ($op === "Multiply") {
        $ans = $num1 * $num2;
    }
    else if ($op === "Divide") {
        $ans = $num1 / $num2;
    }
    else if ($op === "Power") {
        $ans = pow ($num1, $num2);
    }
    else if ($op === "Remainder") {
        $ans = $num1 % $num2;
    }

  echo "<html>";
  echo "<head>";
  echo "<title>Calculator Form</title>";
  echo "</head>";
  echo "<body>";
```

```
        echo "<form action = \"calc.php\" method = \"post\">";

        echo "<p>Number 1</p>";
        echo "<input type = \"text\" name = \"num1\" value = \"$ans\">";
        echo "<p>Number 2</p>";
        echo "<input type = \"text\" name = \"num2\">";
        echo "<p>Operation</p>";
        echo "<select name = \"operation\">";
        echo "<option>Add</option>";
        echo "<option>Subtract</option>";
        echo "<option>Multiply</option>";
        echo "<option>Divide</option>";
        echo "<option>Power</option>";
        echo "<option>Remainder</option>";
        echo "</select>     ";

        echo "<input type = \"submit\" value = \"Calculate\">";
        echo "<input type = \"reset\" value = \"Clear values\">";

        ?>

        </body>
      </html>
```

Exercise 3: Presentation

Prepare a short, five-minute presentation on the results of your research from Exercise 1 above. If you have found out anything particularly interesting, you should focus on that as a priority.

Exercise 4: Revision

Review the lecture material and ensure that you are comfortable with everything discussed thus far.

4.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide. This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, partially as a way to share information with others in the class but also as a way to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

Exercise 1: Discussion of Private Study Exercise

Discuss as a group your solution to Exercise 2 in the private study.

Exercise 2: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short five minute presentation ready to give to the rest of the class. There is no need for this to be especially formal - you are simply reporting on anything interesting that you found during your research, or pointing out especially useful resources on the topic. Bring your journal along to the class so that you can make a note of anything especially useful that your classmates have mentioned. This is a knowledge dissemination exercise; you are not being formally assessed on the style or content of the presentation.

Topic 5: Design and Build A Database (2)

5.1 Learning Objectives

This topic provides an overview of mySQL and the database connection to a web page. On completion of the topic, students will be able to:

- Design and build a database which interacts with a web page.

5.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. It is strongly advised that you implement the coded elements of the lecture in front of students so they can see a MySQL/PHP application being constructed. Slides with code upon them are good points to stop and show students how they relate to the application you are building.

They will then develop the code during the tutorial and laboratory sessions.

5.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

5.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- POST and GET
- An introduction to SQL
    - Database queries
    - Data types and ranges
    - SQL statements.

5.4.1 Guidance on the Use of the Slides

Slide 4: This lecture introduces MySQL into the N-tier architecture we have been building. It slots into the data layer, and PHP acts as an intermediary between the presentation and the data layer. It should be the case that the presentation layer never has cause to directly communicate with MySQL (it should all occur through the mediation of the application layer), and that the data layer never directly communicates with the presentation layer. This ensures the requisite amount of abstraction between the three parts of the architecture.

Slide 5: This diagram shows how MySQL fits into our growing architecture. We will continue to populate this diagram as time goes on and we encounter new protocols, tools and systems.

Slide 6: It is a requirement of this module that students have the resources to access their own MySQL databases – it does not matter if these are set up for them or they are given access to an account on which they can create their own. However, all of the future content is based on the idea that students have accounts with which they can explore the concepts that are discussed. If your arrangements are such that localhost is not appropriate, you should make whatever corrections you require on the basis of your institutional provision.

Slide 7: For the examples we go through here, the username and password are as they actually exist on a real web server. Students should replace the values outlined here with their own data when it comes time to translate the concepts into functioning web applications. Their databases will be assumed to be empty, containing no tables – there is no need for them to have access to phpMyAdmin or any other MySQL management application, but these can be valuable tools if they can be provided without undue inconvenience.

Slide 8: This slide outlines the code needed to connect to a MySQL database – all four of the variables are dependent on your institution's configuration and the student's database details. The mysql_connect function is what handles the job of making a connection with the database located on the host. The die() function causes a program to terminate (because the database connection failed to be created, in this case).

Slide 9: Different browsers and different server settings vary wildly in the level of detail they provide for PHP or MySQL errors. You should consult your platform documentation to see how the maximum amount of information can be provided to students for these, because debugging database applications can be frustrating if enough data is not provided to the developer. In the case of a connection though, there are only three possible ways in which this statement can fail, and these are listed on this slide.

Slide 10: Creating the connection to the MySQL process does not select a database to work with, and this is done through the mysql_select_db function as is shown on this slide.

Slide 11: Manipulating a MySQL database is done through queries. It is the assumption of this topic that students will already be familiar with simple SQL from previous modules – if not, you may wish to take some time here to provide a short refresher. The amount of SQL knowledge assumed is not high, and covering the principles of CREATE, INSERT, and SELECT will cover all that is required for this topic. Queries are executed through the mysql_query function, as we see in the next slide.

Slide 12: HTTP permits data to be sent to web pages in two different ways. This can be by using "GET" or "POST". "GET" is the method that is used by the browser to ask the server to send back to a given source and uses URL. "POST" is the method the browser uses to talk to the server when it is asking for a response and uses HTTP. Anything that is sent via GET is shown in the browser's address bar. However, anything sent by POST can only be seen by any hackers. Post is the best method to use when sending data between pages.

When the GET protocol is used, the data that is sent from the form elements get encoded directly into the URL. Spaces get replaced with a special code (%20), and a few other symbols are adjusted (which has importance when making use of web services). Each piece of data is delimited with an &, and the parameter list itself follows a ? in the URL. In dice_roll.get.php?num=6&faces=7, what we have is two parameters (num and faces), which have the values 6 and 7 accordingly. Our PHP scripts can make use of these variables by using the $_GET associative array – we need to know how the data is coming in, because our PHP code must either be changed to accommodate the format, or be written in such a way that it does not matter which is used.

Slide 13: The POST protocol is more useful on a day to day basis, because it has no limitations in terms of data size or data formats – anything that can be represented in PHP can be sent via a POST request. Unlike GET which encodes the data into the URL, POST encodes it directly into the HTTP header which obscures it from casual view.

Both of the protocols fulfil the same role and both suffer from the same limitation – HTTP statelessness, as we have discussed earlier. If we move outside the confines of a single PHP script we lose the data unless we manually carry it around with us (which we can do, for example, by embedding the data in the URLs that our PHP page constructs, but this is an extremely cumbersome method with the same limitations outlined in Slide 10). They serve the role of getting information from the user to the server, but not from one part of the server to another.

Slide 14: As with all software there are some limitations of POST and GET and this slide identifies these.

Slide 15: This slide contains the code needed to create a table in our empty database – in this case, the table is called **test_table** and has two fields. We create the query as a string, and then pass both the query and the connection we gain as part of mysql_connect to the function mysql_query which then executes the query on the appropriate database.

Slide 16: Creating a table is a process that can only be done once (unless the table is deleted before recreated), and as such setting up the tables used by a dynamic website is usually handled as part of a special case **setup.php** page that is not part of the usual application workflow – the idea of this is much like that of a standard application installer. When working with a dynamic application, the user installing it will browse to this setup.php page, follow the steps instructed there, and then the page will be deleted or moved to prevent accidental reinstallation.

Slide 17: There is no need for a student to touch the database directly. PHP will allow for almost every manipulation to be done provided the user associated with the database has the appropriate privileges. These slide shows the code for inserting a pair of values into the database – every time we browse to this page, the data will be entered and that is unlikely to be the desired goal. In a later slide, we will look at incorporating user input to allow for this to be done more elegantly.

Slides 18 - 19: Data that is returned from queries in PHP are handled as **associative arrays** – we have already used these in the past few lectures. They are sometimes also known as hash tables or mappings. An associative array is one in which the **key** is associated (usually as a result of some kind of arithmetical manipulation) with a **value**. In the case of the PHP code we have, the key would be the database record field, and the value would be what is contained within that field for a record.

Slide 20: The associative arrays that are provided as the results of a query take the form of an **array** of **associative arrays** – each element of that array is an associative array that represents a single record of the database. Getting the data out of the database is handled by an SQL query, but it is necessary for us to manipulate that data in PHP to incorporate it in whatever algorithms or workflows we are providing. The Associative Array is an array that uses named keys that are assigned by the user. This can be used to give each item a specific value eg someone's age.

Slide 21: We will now move onto how we manipulate the results and query the database using PHP scripts.

Slide 22: PHP gives us a number of useful functions for manipulating these records. The first of these is mysql_num_rows which takes the results of a mysql_query function as its parameter and returns an integer that represents the number of records returned. This is valuable information because we often want to iterate over the elements in this array.

Slide 23: This slide shows the code for executing a simple select and then displaying the number of rows that were returned for the query. It does not output any of the actual data that is associated with the query, just the number of records.

Slide 24: The number of records is important, but the data that is contained within the results is more important still. There are two ways to extract this information – the first is through the mysql_result function which allows for precise access to particular records and particular fields. An example of this function in use is shown at the bottom of this slide.

Slide 25: mysql_fetch_array allows for more flexible querying and manipulation of the records. The result set contains an internal counter that is incremented after each invocation of mysql_fetch_array, so calling it repeatedly will result in the counter being incremented. At each invocation, it returns the record currently being pointed to by the counter, or a null value if it has reached the end of the set.

Slide 26: This slide shows the code for making use of mysql_fetch_array. It uses the number of results gained from using mysql_num_rows, and bases a for loop around that to iterate over each element in turn. The HTML code handles displaying this data in a simple table for presentation purposes.

Slide 27: This slide shows the incorporation of user input into a simple query. The results can then be presented, for example, in a table.

Slide 28: This slide shows an example of a query that can be used to query the data where data is specifically requested from the database which means the Select parameters.

Slide 29: It is a good rule as a developer to never trust anything that comes from the user. We cannot guarantee that data we ask for will come in the proper format, or will not be provided in a way that is intended to be malicious. One of the ways users can provide malicious input is through the **sql injection**, whereby the input includes well-formed SQL designed to be executed on a database. Modern browsers do some of the work in preventing this, but a determined mischief maker can easily write a simple socket-based program to send an HTML header that gets around browser based safeguards.

It is important that students **sanitize** the data that comes into their application from a user – the mysql_real_escape_string function will do this for them, they just need to remember to make use of it. This function will make sure that potentially dangerous characters such as quotation marks and apostrophes are properly escaped so that they get rendered as their ASCII code and not in their symbolic role.

Slide 30: This shows the code for sanitised input. You should insist that all input that comes into contact with user data is sanitised in all student submissions.

Slide 31: The nature of the integration between PHP and MySQL raises questions of what part should be performing which operations. You could simply have PHP request a huge data set (select * from <table>) and then use PHP code to iterate over this to do filters and more specialised queries, or you could use the mechanisms provided in MySQL to do it instead. Both are equally plausible approaches, but students should be mindful of what it is they are trying to do, and the scalability of their approaches.

Slides 32: The work that MySQL does best is storage, retrieval and querying of data. It is a platform designed around extremely efficient processing of these three things, and it is hugely unlikely that students will be able to match that level of optimisation. However, SQL strings tend to become very large and complicated when sophisticated queries must be performed, and these can be difficult to construct, difficult to read, and difficult to manipulate. On the other hand, PHP can be excellent for fine control over data when selection is not necessarily straightforward (for example, select all salaries under £50k a year until we have a total of £1.5 million, and then after that only select the ones that are over £250k).

Slide 33: In all cases, it depends on a balancing act of traits. MySQL is almost always going to be more efficient, but that is not necessarily the only outcome you want to incorporate in your program. Readability and maintainability of code is important, especially if the efficiency elements of MySQL are unlikely to be an issue (these start to become a major consideration at the scale of thousands / tens of thousands of queries per second). Portability is important too, especially if you are potentially going to be shifting between data layers – SQL is not a standardised language, and different vendors use different syntax.

Slide 34: This slide talks about the web development frameworks.

Slide 35: Ruby is an open source software programme that has become very popular over the last few years and it has become very popular particularly with the Ruby on Rails framework for the web. This software is completely free to use, copy, modify and distribute. Ruby is different to other software programmes as everything is an object and information and code are given their own properties and actions. Ruby is an example of object-orientated approach and includes some great features such as exception handling to handle errors from Java or Python. Ruby on Rails is a web application that runs on the Ruby programming language. You will need to have Ruby, RubyGems and a SQLite3 Database – have a look at the following web link which provides information on how to get started. http://guides.rubyonrails.org/getting_started.html. One of the advantages of using rails is that it is fun and should allow you to write less code than using other languages and frameworks. Rails is based on the philosophy that you don't repeat yourself and convention over configuration is preferred.

Slide 36: This slide provides an example as to how Ruby can be used to show the text Hello Andy! If you are going to use Ruby on Rails you will need to learn the language. You will see that this is very different from using HTML and there are no <tags> as such in the code. Some designers feel that Ruby on Rails is much better than using PHP, however they are different as PHP is a programming language, whereas Ruby on Rails is a Framework developed in the Ruby language. You will need to decide which language to learn as it is difficult to learn two languages and implement them successfully. Rails comes with different templates that allow you to customise applications and you may decide to use Ruby on Rails once you are confident using PHP and it allows much faster web development. The difference with Rails is that to create a static home page you need a route, a controller and a view. Rails works well with Bootstrap and these styles can be imported into the Rails Gemfile. Ruby has lots of plugins that can be used eg for creating tabs and navigation menus instead of creating these from scratch.

Slide 36-38: Conclusion, Terminology and References

5.5 Laboratory Sessions

The laboratory time allocation for this topic is 2 hours.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

There is only one exercise for this laboratory session, as it is relatively intricate and should take students the full allotment of time.

Exercise 1:

Using the techniques that have been discussed in the past four lectures, and using Andy's Autos website, create the MySQL and PHP architecture for a login page. It should permit the following operations:

1. Register your account
2. Prevent people registering usernames that already exist
3. Report on successful logins
4. Prevent access from invalid username/password combinations
5. Create a database to store customer details including the following fields:
    a. customerID
    b. firstName
    c. lastName
    d. address1
    e. Town
    f. County
    g. postcode
    h. telNo
    i. mobNo
    j. eMail
6. Create a php script that will allow you to input data into the database from a webpage which includes a registration form. The form should match the fields in the database.

Suggested Answer:

Registration Html Page

```
<html>
  <head>
    <title>Example Registration</title>
  </head>
  <body>
  <form action = "attempt_reg.php" method = "post">

     <p>Username?</p>
     <input type = "text" name = "username">
     <p>Password</p>
     <input type = "text" name = "password">
<p>Email?</p>
<input type = “Email” name = “Email”>
     <input type = "submit" value = "registration">
     <input type = "reset" value = "Clear values">

  </body>
</html>
```

Registration PHP Script

```
<html>
  <head>
    <title>Simple Registration</title>
  </head>
  <body>

  <?

    $username = $_POST["username"];
    $password= $_POST["password"];
    $email = $_POST["email"];

    $db = mysql_pconnect ("localhost", "monke13_nccuser", "ncc1");
    mysql_select_db ("monke13_ncc");

    $query_check = "select * from User where Username=\"$username\"";

    $results = mysql_query ($query_check, $db);

    if (!$results) {
      echo "<p>" . mysql_error() . "</p>";
    }

    $num_results = mysql_num_rows ($results);

    if ($num_results != 0) {
      echo "<p>That username already exists</p>";
      echo "<a href = \"example_reg.php\">login</a>";
      exit;
    }

    $query = "insert into User (Username, Password, Email) values
(\"$username\"
    , \"$password\", \"$email\")";

    mysql_query ($query);


    if (!$query) {
      echo "<p>" . mysql_error() . "</p>";
    }
echo "Registration successful";
    echo "<a href = \"example_reg.php\">login</a>";


  ?>

  </body>
</html>
```

Login HTML Page

```
    <html>
      <head>
        <title>Example Login</title>
      </head>
      <body>

      <form action = "attempt_login.php" method = "post">

         <p>Username?</p>
         <input type = "text" name = "username">
         <p>Password</p>
         <input type = "text" name = "password">

         <input type = "submit" value = "login">
         <input type = "reset" value = "Clear values">

      </body>
    </html>
```

Login PHP Script

```
    <html>
      <head>
        <title>Simple Login</title>
      </head>
      <body>

      <?

        $username = $_POST["username"];
        $password = $_POST["password"];
        $db = mysql_pconnect ("localhost", "monke13_nccuser", "ncc1");
        mysql_select_db ("monke13_ncc");

        $query = "select * from User where Username=\"$username\" and
    Password=\"$password\"";

        $results = mysql_query ($query, $db);

        $num_results = mysql_num_rows ($results);
        if ($num_results == 0) {
          echo "<p>Invalid login</p>";
          echo "<a href = \"example_login.php\">Register</a>";
          exit;
        }

        echo "<p>Login successful</p>";

      ?>

      </body>
    </html>
```

Customer Detail HTML

```
<!DOCTYPE html>

<html>  <head>  <meta  charset="UTF-8" />

<title>Input Form</title><link rel="stylesheet" type="text/css" href="demoStyles.css">

</head>

<body>

<form action="inputData.php" method="post" name="inputFrm">

                        <label>Insert Data into mySQL Database</label>

                        <br />

                        <br />

                        <label>First Name:</label>

                        <input name="firstName" type="text" />

                        <br />
```

```
<br />
<label>Last Name:</label>
<input name="lastName" type="text" />
<br />
<br />
<label>Address 1:</label>
<input name="address1" type="text" />
<br />
<br />
<label>Town:</label>
<input name="Town" type="text" />
<br />
<br />
<label>County:</label>
<input name="County" type="text" />
<br />
<br />
<label>Post Code:</label>
<input name="postCode" type="text" />
<br />
<br />
<label>Tel No::</label>
<input name="telNo" type="text" />
<br />
<br />
<label>Mobile no::</label>
```

```
                <input name="mobNo" type="text" />

                <br />

                <br />

                <label>Email Address:</label>

                <input name="eMail" type="text" />

                <br />

                <br />

                <input name="Submit" type="submit" value="Submit" />

                <input name="Reset" type="reset" value="Clear" />

</form>

</body>

</html>
```

Customer Details PHP Script

```
<html>
  <head>
    <title>Input Form</title>
  </head>
  <body>

  <?

    $username = $_POST["username"];
    $password= $_POST["password"];
    $email = $_POST["email"];

    $db = mysql_pconnect ("localhost", "monke13_nccuser", "ncc1");
    mysql_select_db ("monke13_ncc");



     $firstName=$_POST['firstName'];

     $lastName=$_POST['lastName'];

     $address1=$_POST['address1'];
```

```
        $Town=$_POST['Town'];

        $County=$_POST['County'];

        $postCode=$_POST['postCode'];

        $telNo=$_POST['telNo'];

        $mobNo=$_POST['mobNo'];

        $eMail=$_POST['eMail'];



        $sql="INSERT                                                INTO
        $tbl_name(firstName,lastName,address1,Town,County,postCode,telN
        o,mobNo,eMail)VALUES('$firstName','$lastName','$address1','$Tow
        n','$County','$postCode','$telNo','$mobNo','$eMail')";

        $result=mysql_query($sql);

        if($result)

        {

            echo "Successfull";

            echo "<br />";

            echo "<a href='input.php'>Back to main page</a>";

        }

        else

        {

            echo "ERROR";

        }

          ?>

       </body>
   </html>
```

5.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Topic 5

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

As part of your ongoing journal exercise, you should research the following topics:

- SQL Injection Attacks
- SQL syntax for MySQL
- Advantages and disadvantages of MySQL versus other database platforms
- Review of Ruby on Rails and at least three positive points and evaluation as to why you would use it.

Exercise 2: Login Lockouts

Most login systems won't let you simply keep trying until you get in. Implement a lock-out in the login code that you have developed above. It should keep a track of how many unsuccessful attempts were made to log into an account, and if that number goes above three it should automatically report any future logins as failures (even if they're not) for a total of five minutes.

Suggested Answer:

There are various ways to approach this problem, but the most straightforward is to include two new fields in the database (attempts, and time). When making a check, the login code should check the number of attempts, and if's over a certain threshold it should check the time. The code for the PHP side of this is presented here:

```
<html>
  <head>
    <title>Simple Login</title>
  </head>
  <body>

  <?

    $username = $_POST["username"];
    $password = $_POST["password"];

    $db = mysql_pconnect ("localhost", "monke13_nccuser", "ncc1");
    mysql_select_db ("monke13_ncc");

    $query = "select * from User where Username=\"$username\"";

    $results = mysql_query ($query, $db);
    $num_results = mysql_num_rows ($results);


    if ($num_results > 0) {
      $attempts = mysql_result ($results, 0, "Attempts");

      echo "Attempts is $attempts";

      $pass = mysql_result ($results, 0, "Password");

      if ($pass == $password) {
      $time= mysql_result ($results, 0, "Timestamp");
        if ($attempts >= 4 && $time >time()) {
          echo "<p>Invalid login</p>";
          echo "<a href = \"example_login.html\">Try again</a>";
        }
        else {
          $query = "update User set Timestamp=\"0\" where Username =
\"$username\"";
          mysql_query ($query, $db) or die (mysql_error());
          $query = "update User set Attempts=\"0\" where Username =
\"$username\"";
          mysql_query ($query, $db) or die (mysql_error());

          echo "<p>Login successful</p>";
        }
      }
      else {
        $attempts = $attempts + 1;
        $time = time() + (60 * 1);
        echo "Attempts is $attempts";

        $query = "update User set Attempts=\"$attempts\" where
Username = \"$username\"";
        mysql_query ($query, $db) or die (mysql_error());

        $query = "update User set Timestamp=\"$time\" where Username
= \"$username\"";
        mysql_query ($query, $db) or die (mysql_error());


        echo "<p>Invalid login</p>";
        echo "<a href = \"example_login.html\">Try again</a>";

      }
    }
    else {
      echo "<p>Invalid login</p>";
      echo "<a href = \"example_registration.html\">Register</a>";
    }
  ?>

  </body>
</html>
```

Exercise 3: Presentation

Download Ruby on Rails software http://rubyonrails.org/ and create a new rails project. This should take you about an hour to complete. Create at least two pages for Andy's Autos which includes a home page and contact us page (which includes a form).

Have a look at the following webpage to find out how to get started:

http://guides.rubyonrails.org/getting_started.html

Produce a presentation which includes screen shots of the two pages you have created and a reflection on how you found the software and an evaluation as to why you would use it over other software.

Exercise 4: Revision

Review the lecture material and ensure that you are comfortable with everything discussed thus far.

5.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, partially as a way to share information with others in the class but also as a way to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

Exercise 1: Discussion of Private Study Exercise

Discuss as a group your solution to Exercise 3 in the private study.

Exercise 2: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short five minute presentation ready to give to the rest of the class. There is no need for this to be especially formal - you are simply reporting on anything interesting that you found during your research, or pointing out especially useful resources on the topic. Bring your journal along to the class so that you can make a note of anything especially useful that your classmates have mentioned. This is a knowledge dissemination exercise; you are not being formally assessed on the style or content of the presentation.

Topic 6: Using Scripts (1)

6.1 Learning Objectives

This topic provides an overview of Ajax and JavaScript. On completion of the topic, students will be able to:

- Make use of JavaScript to create simple client-side applications;
- Make use of Ajax to implement simple server communication.

6.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. It is strongly advised that you implement the coded elements of the lecture in front of students so they can see a MySQL/PHP application being constructed. Slides with code upon them are good points to stop and show students how they relate to the application you are building.

They will then develop the code during the tutorial and lecture sessions.

6.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

6.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- The structure of Ajax
- The document object model
- JavaScript
- Simple Ajax applications

6.4.1 Guidance on the Use of the Slides

Slide 4: In this lecture, we begin to expand on the presentation layer in our N-Tier architecture. We do this by talking about client-side technologies designed to create a responsive and effective front-end to our systems. Ajax is the mechanism by which this is done nowadays, and comprises of a range of related technologies held together by the scripting language JavaScript. Ajax is a complex topic, and will require at least two lectures to cover the necessary fundamentals. This lecture is intended as an introduction to Ajax, JavaScript, and the simplest kind of server communications.

Slide 5: The Ajax family of technologies references XML specifically in the name, but it is not actually a mandatory part of the framework (many people prefer to use JSON instead of XML, and we will look at that later in the module). Ajax, despite being often referred to in terms that might make people think it is something new, is actually just a formalisation of a number of mature technologies that were already in common usage. When any web page is loaded the browser creates the Document Object Model of that page. This HTML DOM model is constructed as a tree of different objects. Javascripts are the scripts which can change all HTML elements in the page or it can add new HTML elements to the page or it can create events in the HTML page.

Slide 6: Most of what Ajax does is based on a single object called the XMLHttpRequest object. This object can be configured to provide the communication between client and server through callbacks and the GET/POST methods that were discussed in an earlier lecture. All of this though is done over HTTP – all communication with the server is done via this protocol, and the Ajax is contained within an HTML document.

Slide 7: The basic lifecycle of an Ajax application is that, when the function is invoked, it creates the XMLHttpRequest object, configures it, and then uses that as the basis of the request made of the server. The server (or, the application layer) then processes that information and sends back the response either as flat text or as an XML/HTML/XHTML document. The browser takes the response and does something with it, usually through the medium of JavaScript, and the browser then updates the page content to reflect the change. The result is that the outcome can be seen in the same page, and there is no need to wait for the browser to parse another static page from the server.

Slide 8: This slide shows Ajax inserted into our N-Tier architecture. As it never directly interacts with the server (it does this only through the HTML document in which it is housed), its communication is only with HTML. You can think of it then as a layer **on top** of the existing HTML that we have used so far.

Slide 9: JavaScript is the core language that links all of the various Ajax technologies together. Despite its name, JavaScript has virtually nothing to do with the Java programming language; the name was a marketing ploy rather than a reflection of structural similarities. JavaScript is a **client-side** scripting language which we can use to offload some of the processing responsibilities from the server. It is not always appropriate or possible to do this, but when it comes to data validation or presenting large amounts of structured content, it is best if the client does as much of the work as is possible to reduce the burden on any central points of failure. Some examples of JavaScript are alert boxes, showing the date/time the page loads, animating the background colour of a document or animating texts.

Slide 10: JavaScript is embedded in HTML documents much like the PHP scripts we have written thus far. The main difference is that since it is interpreted in the browser, the HTML page that the user gets contains all of the client-side code. We use the **script** tag of HTML to incorporate JavaScript code, as is shown in the code sample on this slide. In this case, it simply writes out ‘Hello World’ and the current date to the page. Scripts can also be placed in external files and have the file extension .js. There are several advantages to this in that it separates the HTML and the Code and it makes the HTML and JavaScript easier to read and any cached JavaScript’s are speedier to load.

Slide 11: Before we can begin to unravel much of JavaScript, we must talk about what is at the core of its presentation engine – the Document Object Model. This is a way of representing XML documents as easily navigated tree structures. As HTML can be thought of as a specialisation of generic XML, it can also be used to create an easily navigated tree of HTML mark-up elements. There are various ways to parse XML documents. In an earlier topic we looked at handling it via callbacks that triggered when tags or content were encountered. That is a system properly known as SAX (Simple API for XML). In SAX it is the responsibility of the parser to place meaning on XML tags. DOM Parsing on the other hand builds a tree of the XML document and allows the developer to explore it as a single cohesive entity.

Slide 12: Tree structures are common in computer science, and that is because they are relatively easy to change, modify and expand. Within our HTML documents, context can be identified (provided some hint is given in the HTML mark-up through use of attributes such as ‘id’) and elements manipulated as nodes on the tree. Each of these nodes may contain subnodes, and as part of our manipulation of a DOM tree, we may have to iterate over several children of a node to achieve our goals.

Slide 13: This slide shows some simple XML mark-up for a library. We already know how to parse it through SAX, but the next slide shows what we end up with when it is parsed through DOM.

Slide 14: The tree structure here allows us to start at any node and explore down into that node for all its constituent subnodes. If we wanted to get the surname of all the authors in the tree for example, we would iterate over each of the Book nodes, then drill down to the Author node of each, and then drill down into the Surname node of each author. It is easy to drill down into nodes to get access to whatever may be contained within.

Slide 15: JavaScript makes heavy use of the DOM model to permit manipulation of web pages. We do not need to do anything special, it is done for us. When we access the **document** in JavaScript, we are accessing the DOM representation of the HTML page in which we are working. Inside this object is a wide range of useful methods, which we do not have the time to fully explore (students will be encouraged to do so during their private study). For now, the most important method is **getElementById**, which allows us to access an element based on the ID attribute it is given in the HTML. In this way, we do not even need to manually traverse the tree, we just call the function and get out the element we want.

Slide 16: This slide shows the code for a JavaScript program that manipulates the contents of the DOM. It uses getElementByID to get access to the <p> element that is contained in the HTML, and then sets its InnerHTML property (more on this in the next slide) to be ’Hello World’. The result of this is that the browser dynamically changes `<p id = ”test></p>` into `<p id = ”test>Hello World!</p>`. It is this ability to change the content of web pages after they have been downloaded that drives much of the flexibility of JavaScript.

Slide 17: Inner HTML represents the textual content of an element (that is, the text between the tags). It is not actually part of the formal DOM specification, but is so useful that all the major browsers support it. However, JavaScript that exists only to be executed in a linear manner by the browser has limited scope – what we want is to be able to have JavaScript that is executed when certain things happen. We want to be able to bind our JavaScript to **events,** such as buttons being pressed, pages being loaded, or text fields being changed.

Slide 18: This is an example of a JavaScript function that is executed when the button is pressed. **Alert** functions as a message box, and just displays whatever was typed into the textbox. Note here two differences though from our previous example – one is that a parameter is passed into the function (this.form – form is an object, like document, that JavaScript maintains for us), and that we use the **value** property of the **myText** object to get the contents of the textbox.

Slide 19: This is an example of a JavaScript function which opens a prompt box so that users or customers can enter a value before entering the page. The prompt box opens when the page is loaded and user clicks “ok” or “cancel” and enters a value. If the user clicks ok then the box returns the input value. If the user clicks cancel then the box returns null.

Slide 20: JavaScript is core to understanding how we incorporate Ajax into our architecture. Expanding our system to include Ajax is slightly complicated, because browsers handle things differently, and our client-side scripts need to be ready to deal with older browsers as well as newer browsers. Thus, on the next slide we give a bit of **boilerplate** code that can be incorporated into any Ajax request that is made.

Slide 21: This is a multi-platform mechanic for getting an XMLHttpRequest object from the browser as older browsers don’t support the XMLHttpRequest object. This code checks the browser and provides an alternative for older browsers.

Slide 22: Communication with the server is done through GET and POST, as was discussed in a previous lecture. Once we have our Ajax request object, we can use **open** to setup the connection we want to make, and **send** to make the request of the server. The only parameter here that is unexpected is the third, which is whether the request should be handled asynchronously. This parameter is what handles whether or not the rest of the JavaScript should continue to execute while the server communication is ongoing – it should almost always be set to true. Asynchronous communication means that we do not need to wait for the server; the callback that we define will be triggered for us when the data is ready.

Slide 23: . For students who are used to being the ones who define what functions get called and when, callbacks can be a slightly confusing system to get into their mind. All Ajax requires us to do though is define a function that will be executed when the server provides us with a response. We have to do this before we send a request, and we usually do it by **inlining** the function directly into the object.

Slide 24:	When the callback function is called, the request object will contain some useful data for us regarding the state of the server communication. We are looking for a ready state of 4 (completed), and a request status of 200 (server reports okay). There are situations where we'd want to handle other values for these, but not in the context of this module. For now, we only want to make sure that the request worked before we trigger the code that handles the rest of our interaction.
Slide 25:	In our first Ajax application, we have a small text file on the server. When the button is pressed, we're going to retrieve that text file and replace the current contents of the "contents" tag with what we got from the server.
Slide 26:	This is the code that handles that transaction, minus the browser compatibility code (that would need to be in place too). The main part of the code here is the setting of the **onreadystatechange** property to have our callback function. We check the ready state and the request status, and if these are appropriate we change the inner HTML of the tag as we did with JavaScript. The key difference here is that we get the specifics of the interaction from the server and not from the client. This is the key of what makes Ajax such a powerful framework.
Slide 27:	There are many events that are triggered during an HTML document's lifetime, and Ajax handling code can be attached to all of these. Students will be encouraged to investigate the events of which they can make use during their private study time.
Slide 28:	In our second Ajax application, we are going to have the Ajax code trigger on a mouseover rather than on a click, and we are going to base the text file we request on the specific image that we move the mouse over (a term traditionally referred to as a 'mouseover'). The result is a simple graphical encyclopaedia that changes the text based on where our mouse is. Note in the event handlers here that we are passing in the filename of a text file. This is then handled in the JavaScript.
Slide 29	The structure of this function is largely identical to the last, except that it has a parameter (URL), and that parameter serves as the basis for the server communication. The result is that the text file we get depends on which image we have moved our mouse over.
Slide 30:	This slide provides an example of how you can create a script which includes Array with different variables. An array can hold more than one value at a time, however in our example each item has a single variable. However, if you want to loop through the cars to find a specific one then you can create a JavaScript Array.
Slide 31:	In this example the array holds more than one value and is the easiest way to create a JavaScript Array.
Slide 32:	This script will calculate the price multiplied by the quantity and can be added to a button on the web page. For larger products and sales this would not be the most effective way to calculate prices but for simple calculations this can be useful.
Slide 33:	As an architecture for a dynamic application, this is very simple – in a future topic we will look at how we can incorporate proper data persistence via PHP/MySQL. Most of the time, we will not be looking to incorporate flat text into our front end (that is

what is held in responseText – the raw content of whatever page we were served), and instead of using responseText we will make use of responseXML which gives us a more easily manipulated XML document.

Slide 34: The key benefit we get out of all of this is that we do not need to wait for page refreshes to get information that is updated via a server. Using PHP, we could easily have made an application that let us click on a link and get text information about the image. However, each interaction would have required a page refresh, and that is slow and unresponsive. With Ajax, we create a front-end that is genuinely responsive and this has the effect of making our dynamic websites seem like seamless entities rather than a mixture of different, albeit compatible, technologies.

Slide 35: Conclusion

Slide 36 -37: Terminology and References.

6.5 Laboratory Sessions

The laboratory time allocation for this topic is 2 hours.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Making use of JavaScript, write web pages that perform the following tasks:

1. Implement a dice roller as in Exercise 1 in the laboratory session for Topic 4

2. Add a script to Andy's Autos website which allows you to validate that a date of birth provided by a user is in a valid form (dd/mm/yyyy). You do not need to validate that the date is sensible (i.e you do not need to worry about the number of days in months etc), just that it adheres to that form.

Suggested Answer:

Solution 1

```
<script language="JavaScript">

      function rollDice (form) {
        var num = form.numDice.value;
        var faces = form.numFaces.value;
        var total = 0;
```

```
          var i;

          for (i = 0; i < num; i++) {
            total = total + Math.floor(Math.random()*faces) + 1;
          }

          document.getElementById ("results").innerHTML = "You rolled:
" + total;
        }

    </script>

    <form name = "testForm">
      <p>Enter the number of dice</p>
      <input type = "text" name = "numDice">
      <p>Enter the number of faces</p>
      <input type = "text" name = "numFaces">

      <input  type="button"  value="Roll  Dem  Bones"  name="button"
onClick="rollDice (this.form)">
    </form>

    <p id = "results">Total goes here</p>
```

Solution 2

```
    <script language="Javascript">


      function validateNumbers (txt) {
         var i;

         for (i = 0; i < txt.length; i++) {
          if (isNaN (txt[i])) {
            return false;
          }
         }

         return true;
      }

      function validateSegment (txt, num) {
        if (txt.length != num) {
          return false;
        }

        if (!validateNumbers (txt)) {
          return false;
        }
```

```
          return true;
        }

        function validate (form) {
          var dob = form.dob.value;
          var i;
          var tokens = dob.split( "/" );
          var result = true;

          if (tokens.length != 3) {
            return false;
          }

          if (validateSegment (tokens[0], 2) == false) {
            return false;
          }

          if (validateSegment (tokens[1], 2) == false) {
            return false;
          }

          if (validateSegment (tokens[2], 4) == false) {
            return false;
          }

          return true;
        }
   function validateDoB (form) {
          var result = validate (form);

            document.getElementById("results").innerHTML
             = "Validation: " + result;
        }
      </script>

      <form name = "testForm">
        <p>Enter a date of birth as dd/mm/yyyy </p>
        <input type = "text" name = "dob">

        <input     type="button"     value="Validate"     name="button"
  onClick="validateDoB(this.form)">
      </form>

      <p id = "results">Total goes here</p>
```

6.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Topic 6

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

As part of your ongoing journal exercise, you should research the following topics:

- JavaScript syntax
- The Document Object Model
- Events in JavaScript

Exercise 2: Andy's Autos

Making use of the architecture shown in the lecture, add to your web-application for Andy's Autos. You should have images that link to various external sites, and then use ajax events to add information about the cars that are shown on the site. You should have mouseover explanations about the value the cars, and also images that change when the mouse is moved over them.

Add in at least one

- calculation script,
- one loop
- one string
- animation of the background display.

Suggested Answer:

Highly dependent on students and student interests – no suggested answer.

Exercise 3: Presentation

Prepare a short, five-minute presentation on the results of your research from Exercise 1 above. If you have found out anything particularly interesting, you should focus on that as a priority.

Exercise 4: Revision

Review the lecture material and ensure that you are comfortable with everything discussed thus far.

6.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, partially as a way to share information with others in the class but also as a way to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

Exercise 1: Discussion of Private Study Exercise

Discuss as a group your solution to Exercise 2 in the private study.

Exercise 2: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short five minute presentation ready to give to the rest of the class. There is no need for this to be especially formal - you are simply reporting on anything interesting that you found during your research, or pointing out especially useful resources on the topic. Bring your journal along to the class so that you can make a note of anything especially useful that your classmates have mentioned. This is a knowledge dissemination exercise; you are not being formally assessed on the style or content of the presentation.

Topic 7: Using Scripts (2)

7.1 Learning Objectives

This topic provides an overview of client-side scripting with jQuery. On completion of the topic, students will be able to:

- Make use of jQuery to enhance their front-ends;
- Select elements using jQuery selectors and filters;
- Manipulate and animate HTML elements through jQuery.

7.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. It is strongly advised that you implement the coded elements of the lecture in front of students so they can see a MySQL/PHP application being constructed. Slides with code upon them are good points to stop and show students how they relate to the application you are building.

Students will then develop the code during the tutorial and lecture sessions.

7.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

7.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- An introduction to jQuery
- Effects in jQuery
- Selectors and Filters
- HTML manipulation with jQuery

7.4.1 Guidance on the Use of the Slides

Slide 4: In this topic, we return to the presentation layer of our architecture and discuss jQuery. jQuery is the most popular JavaScript library, and offers huge benefits to developers in terms of increasing the effectiveness of their code and how attractive their interfaces can be. Combined with CSS, it can create very attractive interactive front-ends without requiring huge investment of effort in creating bespoke JavaScript. The jQuery library is really just one huge JavaScript object that has been built and integrated into the DOM for people to use, and as such, it is not a language in itself, just a wrapper around standard JavaScript.

Slide 5: There are several ways to make use of jQuery. It is freely downloadable from the jQuery website (http://jquery.com/), and external repositories are also available whereby the library can be linked into a web application without needing it to be made available locally. For maximum flexibility, we will assume at this point that it is being accessed from an external repository (and this is the style that will be used in the lecture notes).

Slide 6: jQuery is a wrapper around JavaScript and the DOM. It provides a new system for interacting with the underlying HTML document by simplifying the interface and providing new mechanics for developers. It is a very expressive framework, permitting developers to achieve much with only a modest amount of code.

Slide 7: As jQuery is a wrapper, we need to place it within script tags as we would with normal JavaScript. The nature of the framework though is that most code is handled via an event function that is attached to the base document – specifically, the **ready** event. This event gets triggered after the DOM has been fully parsed, but before any of the page has been loaded.

Slide 8: This slide shows our first jQuery function. We access the library using an external repository, and then we bind a click event handler to the <a> tag on the page. When that element is clicked, the page will flash up a message box before taking the user to the destination web page.

Slide 9: One of the most instantly arresting features of jQuery is its library of **effects**. We can use these to add presentational flourishes to our front-end by animating elements and changing their properties in relation to events that occur. Of course, any presentational flourishes of this nature are likely to be distracting if used to excess, but much of the web has incorporated these effects into their design to great effect – think of navigation menus that expand on mouseovers as an example. We can attach effects of this nature to any HTML element, and bind them to any combination of events. The result is tremendously flexible.

Slide 10: This slide shows a second example of effects attached to a hyperlink. In this case, we use preventDefault to stop the link opening its destination web page, and instead fade it out and then back in again. The number passed as a parameter is how long the animation takes to accomplish (the larger the number, the longer the animation lasts)

Slide 11: We do not lose access to normal JavaScript just because we are using jQuery. This slide shows a standard JavaScript loop that works with jQuery to create a link that flashes in and out of visibility.

Slide 12: Our ability to effectively manipulate the HTML elements in our applications is based on how easily we can identify the correct elements and then attach appropriate event handlers. jQuery uses a dollar sign notation, with $(document) referring to the container HTML page and $("a") meaning 'every <a> tag on the document'. However, we are not limited to this – jQuery comes complete with a sophisticated selection system for finding the elements with which we want to work.

Slide 13: This slide shows the code that is required to find an element with the <p> tag and an ID of "information". The system that jQuery is using to locate the matching elements is called its **selector** system, and by combining its special symbols, we can identify any number and combination of elements.

Slide 14: The # symbol is used to locate by ID (think of it as the jQuery equivalent of getElementsByID), whereas an element that has had a CSS class applied to it is identified by using the . symbol.

Slide 15: The flexibility of jQuery is such that we can combine these selectors by separating them with a comma; we can continue to do this until we have provided a suitable distinction to identify the elements we require. We can mix and match selectors also, simultaneously searching by class and by ID.

Slide 16: To go along with the selectors, jQuery also provides special symbols called **filters**. These are functions we can apply to all the elements we received from our selector, so that we can weed out any that are inappropriate. We may wish to get rid of elements that contain no HTML, we may want access to only those elements that have odd numbers in our array of matches, or any number of other things. These filters often come in handy for fine grained control over elements – odd and even for example will be seen in a later slide in the context of zebra-striping a table for extra readability.

Slide 17: There are around 20 different filters supported by jQuery, and students should be encouraged to research them and see how they can be used to support the building of dynamic websites. Filters can be thought of as special functions that get called on each member of the array of elements that our selector provides – as such, they can be computationally expensive when dealing with large arrays.

Slide 18: jQuery allows us to bind a number of events to the elements we receive from our selectors – the usual suite of these is provided, with the ones listed on this slide being especially common. Other events exist for more specialised needs, and again, students should be encouraged to investigate the full range of the library.

Slide 19: This slide shows an example of applying an effect to an element based on matching a particular style. The CSS style is defined in the <head> of the document, and when the hyperlink is clicked, the contents of the <div> tag will either appear or disappear depending on their current visibility. The slideToggle function can be used to swap between these visibility states without requiring specific handling by the developer.

Slide 20: While jQuery provides a library of animation effects, we are not limited to these when creating our interfaces. jQuery allows us to manipulate any arbitrary element of CSS (provided it has a numerical value) to create a wide range of graphical effects. This is handled via the **animate** method, which works identically to the other methods we have seen so far except that it requires an associative array of CSS elements as a parameter. By default, the animation will modify the current value of the element to match the provided value, using the duration to handle how long it will take. However, jQuery also provides for dynamic adjustment whereby we take a current value and modify it by a relative number. Most of jQuery's effects can be provided with a callback that occurs when the animation has concluded.

Slide 21: This slide shows an example of absolute animation, whereby the animation will cause the font size to change to the specified value. If we run the animation twice, the second time will cause no changes to be observed.

Slide 22: This slide shows an example of relative animation – each time the event is triggered, the current font size will be increased by four. Repeated invocations will cause the element to continue growing.

Slide 23: Callbacks are used to deal with JavaScript's line by line execution. If we want code to be executed after an animation has finished, it will not work to simply locate it on the next line – in that case, it will be executed while the animation is running. Instead, we use a callback to ensure that it properly synchronises to our animation. We are not required to provide callbacks (we can either provide a stub as on Slide 21 and Slide 22, or simply leave the parameter blank), but they can be useful when we have related functionality that should be triggered.

Slide 24: This slide shows an example of using the callback to flash up a message box when the animation has concluded.

Slide 25: As with JavaScript and Ajax, our jQuery libraries are designed to allow us to easily manipulate the HTML of our document. jQuery provides the html function which maps onto the innerHTML property of an element in the DOM. We can use this to either query the contents of the innerHTML for an element, or to replace it with a new set of contents as we have seen done with JavaScript.

Slide 26: jQuery makes it similarly easy to alter the attributes of HTML tags. While we have suggested that students making use of XML should avoid attributes if at all possible, we cannot ignore them in HTML. The code shown at the bottom of the slide shows using a selector to get all of the img tags in a document, and then filtering that array of elements to return only those that have no alt tag defined. On the resultant array, we then use the attr function to supply an alt tag, albeit only a placeholder.

Slide 27: We have already seen that we can identify elements based on their CSS styles but we can also alter the style of elements 'on the fly' when a document has been loaded. We simply use the css() function to directly change the CSS that is being applied to an element. The example code here shows zebra-striping on an HTML table (it alternates colours across rows of the table to increase the readability as well as attractiveness of the data).

Slide 28: There are further points to discuss regarding jQuery), but it is worth spending a little time here to explain why jQuery is so useful. That it makes things easier to do is undeniable, and the effects it provides to developers are fun and entertaining. Its most important benefit though is that it is a **cross browser** library – all of the compatibility issues that we have encountered so far are simply not an issue when we use this library.

Slide 29: There are a number of other benefits too; despite its sophistication, it is a library that has a small footprint. There are CSS headers out there that are bigger than the entirety of the jQuery library. Additionally, because jQuery handles a lot of the intricacies of selecting and manipulating elements, it frees us as developers to concentrate on those parts of the system that can best use our attention. Events are much easier to handle than in plain JavaScript, and the presentational flourishes provided by effects and animation can result in more engaging and useable interfaces. However, while there are few technical drawbacks to jQuery, it is still important to realise that it is an extension to JavaScript; students should always consider how to accomplish the same goals in JavaScript, just in case they end up in a situation where jQuery is either not available, or has become unsupported.

Slide 30: Here we show how our technologies to date link together. JavaScript is the mechanism by which we interact with the Document Object Model, and both Ajax and jQuery are built atop of JavaScript. They need JavaScript in order to accomplish any of their design goals. By this point, our diagram is becoming more and more simplified by virtue of how interrelated all the technologies are. XML is the foundation of the DOM model, and yet in our diagram it is firmly contained in the data layer. The diagram that we provide here shows where the technologies fit as far as their role in the N-Tier architecture, rather than where they impact on particular layers. XML is a data representation format, and thus it is contained in the data layer. Ajax may make heavy use of XML (via DOM), but for the applications we design, it is using the XML that represents what is within our data layer.

Slide 31: There are many jQuery's that can be used to create mobile web applications and we looked at this in Lecture 3. However, jQuery mobile also offers various templates and is another example of a web framework. It is based on HTML5-based interface designed to make responsive web sites and apps that are accessible on smartphones, tablets and desktop devises. This framework allows you to design a responsive web pages and includes a ThemeRoller which allows you to create different themes for web pages.

Slide 32: This is an example of jquery for mobile and how the scripts are included in the http page. These are linked to external jquery style sheets and the http://code.query.com/mobile website includes a library of different themes that can be used when building your site.

Slide 33: JSON stands for JavaScript Object Notation which is used to store and exchange data over the internet. It is written as text with JavaScript object notation which begins with a left brace and ends with a right brace and each name is followed by a colon and name/value pairs are separated by a comma. The array is an ordered collection of values. A value can be in a string. JSON is independent to any programming language and is simpler to use than XML.

Slide 34: You may be unsure as to why you would use JSON over XML. XML is written in both open and close tags where JSON doesn't have closing tags. This can speed up development but is also easier for the human eye to read. JSON can bypass the

XMLHttpRequest object by making web serve requests inside a <script> tag. As we will see on the next slide JSON is easier to read than XML, however XML information can be “fetched” from the server with AJAX where the loadDoc () function creates an XMLHttpRequest object and sends the request to the server.

Slide 35:	This slide shows an example written in both JSON and XML and the different use of tags. JSON has “firstName” and “lastName”, whereas XML has <firstName> </firstName> <lastName> </lastName>.
Slide 36:	Conclusion
Slide 37-38:	Terminology and References.

7.5 Laboratory Sessions

The laboratory time allocation for this topic is 2 hours.

Lecturers’ Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Making use of jQuery, create at least two further web pages in Andy’s Autos website that provide the following effects:

1. A form that slides out when a button is pressed.
2. A hyperlink that ‘runs away’ from the cursor when the user attempts to click on it.
3. A series of nested bullet points that fade away, one after the other, when a button is clicked.

Suggested Answer:

Part 1:

```
<html>
  <head>
    <title>Example Registration</title>
  </head>
  <body>
  <script src="http://code.jquery.com/jquery-1.6.1.js"></script>

  <script>
```

```
    $(document).ready(function(){
      $("form#slideout").hide();

      $("input#slide").click (function(event) {
          $("form#slideout").slideToggle ("slow");
      });

    });

    </script>

    <input type = "submit" id = "slide" value = "register">

    <form id = "slideout" action = "attempt_registration.php" method =
  "post">
 <p>Username?</p>
       <input type = "text" name = "username">
       <p>Password</p>
       <input type = "text" name = "password">
       <p>Email?</p>
       <input type = "text" name = "email">

       <input type = "submit" value = "register">
       <input type = "reset" value = "Clear values">

    </body>
  </html>
```

Part 2:

```
  <html>
    <head>
      <title>Example Run Away Link</title>
    </head>
    <body>


    <script src="http://code.jquery.com/jquery-1.6.1.js"></script>

    <script>

    function move() {
      var top = Math.random() * $(window).height();
      var left = Math.random() * $(window).width();

      $("a#runaway").animate({"top" : top + "px", "left" : left + "px"},
  500);

    }
    $(document).ready(function(){
      $("a#runaway").mouseover(move);
    });

    </script>

    <a  style="position:  absolute";  id  =  "runaway"  href  =  "#">Click
  me!</a>
    </body>
  </html>
```

Part 3:

```
  <html>
    <head>
      <title>  Fade Away Bullet Points</title>
    </head>
    <body>


    <ul>
      <li>Point 1<l/i>
      <ul>
        <li>Point 2</li>
        <ul>
          <li>Point 3</li>
          <ul>
            <li>Point 4</li>
          </ul>
        </ul>
      </ul>
    </ul>

    <script src="http://code.jquery.com/jquery-1.6.1.js"></script>

    <script>
    $(document).ready(function(){
      $("a#cascade").click(function() {
        var ind = 3;
        var i, del = 0;
        for (i = ind; i >= 0; i--) {
          del += 1;

          $("ul:eq(" + i + ")").delay (1000 * del).fadeOut (1000);
        }
      });
    });

    </script>

    <a id = "cascade" href = "#">Click me!</a>
    </body>
</html>
```

7.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

As part of your ongoing journal exercise, you should research the following topics:

- jQuery filters
- jQuery selectors
- Events in jQuery
- JSON Objects, Arrays and PHP

Exercise 2: Sprucing Up Your Checklist

The web pages you have created for Andy's Autos can now be made more interactive through the use of jQuery. Add jQuery elements to make your website more visually striking. As some examples of what you might do: Extend the content so as to present further reading in a table, and use jQuery to 'zebra-stripe' the table. Modify it so that when you mouse-over an element from your checklist, it grows in size to make it more distinct from others on the list. Have the table of further reading slide in on a mouse-over of its main point.

Suggested Answer:

No suggested answer – highly dependent on how students have designed their checklists.

Exercise 3: Presentation

Prepare a short, five minute presentation on the results of your research from Exercise 1 above. If you have found out anything particularly interesting, you should focus on that as a priority.

Exercise 4: Revision

Review the lecture material and ensure that you are comfortable with everything discussed thus far.

7.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, partially as a way to share information with others in the class but also as a way to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

Exercise 1: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short five-minute presentation ready to give to the rest of the class. There is no need for this to be especially formal - you are simply reporting on anything interesting that you found during your research, or pointing out especially useful resources on the topic. Bring your journal along to the class so that you can make a note of anything especially useful that your classmates have mentioned. This is a knowledge dissemination exercise; you are not being formally assessed on the style or content of the presentation.

Topic 8: Web Development Tools

8.1 Learning Objectives

This topic provides an overview of different web development tools that can be used to create an interactive website. On completion of the topic, students will be able to:

- Use cookies to provide persistent data for PHP applications;
- Use sessions to provide persistent data for PHP applications;
- Use Ajax to build a database.

8.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. It is strongly advised that you implement the coded elements of the lecture in front of students so they can see a MySQL/PHP application being constructed. Slides with code upon them are good points to stop and show students how they relate to the application you are building.

They will then develop the code during the tutorial and laboratory sessions.

8.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

8.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Cookies
- Sessions
- AJAX and Database development

8.4.1 Guidance on the Use of the Slides

Slide 4: In this lecture you will be introduced to cookies, sessions and RSS feeds. You will also expand your understanding of Ajax so that you can create front-ends to our database that we created in Topics 4 and 5.

Slide 5: Cookies are stored on a user's computer, and have limitations in terms of their size. Sessions are stored partially on the user's computer (in a cookie that contains only a session ID and perhaps some authentication information), but mostly on the server itself. We solve the problem through the use of cookies or sessions. As with POST and GET, these are two ways of accomplishing the same task – permitting persistence of data through a protocol that has no inbuilt support for it. They differ only in their implementations, not in the role that they fill in the construction of a dynamic website. When we use cookies, we must use them in a way that is different to the PHP we have built so far – specifically, we must include the call to the cookies before any output is sent to the client (even the <HTML> tag). This because cookies get interpreted as part of the header of an HTML page and not as part of the page itself. Cookies are available on the next page load, which means if you set and access a cookie in one pass from the server, you will find the values of the cookies are based on what they were before the cookie was previously set.

Slide 6: This slide shows the example code of a PHP script that makes use of cookies. From now on, we will not show the HTML forms that are used as the front-end to these scripts, because they are all variations on a theme, and our discussion of HTML's role in our user interface has come to an end for the moment. The key part of this example program is the bold code which creates a cookie called "texttokeep", gives it the value defined in the variable $thetext, and then sets the cookie to expire in 10,000 seconds after the current timestamp.

Slide 7: This slide shows the next PHP script. In this one, we have access to the cookie that was sent via the $_COOKIE associative array. The first line will show that there is no data stored in $_POST for the value that was sent, because it was not carried through to the next PHP script from the first. Only the cookie will permit access to the text the user entered.

Slide 8: There are only three procedures beyond what has been discussed already that we usually need to perform on cookies – we need to be able to change what is contained in a cookie, which we do by accessing the value directly from the associative array. We need to delete cookies, which we do by setting an expiry date to be some point in the past, and we sometimes need to see if a cookie was accepted (so we can tell whether or not we should be using cookies or sessions).

There are numerous limitations to cookies and thus they have acquired a bad reputation. Some of this is justified in terms of the way they have been used as secret tracking of user behaviour. However, when used in the proper way, which is for small pieces of infrequent communication between client and server, they can be tremendously effective and low-impact.

Slide 9: Sessions fill the same basic role as cookies, but they work in a slightly more sophisticated way, as is expected from a system that was introduced to address the limitations of cookies. When sessions are used, the server requests the cookie and uses that session ID to access the session specific information. All of this is handled automatically via PHP. As with cookies, we must provide the directives to use sessions before we send any HTML at all to the client. We do this through the use of the session_start function, as shown in this slide.

Slide 10: Once a session has been opened, we register new session variables by assigning them directly into the associative array. Once we have done this, they are available to any PHP script that opens a session, and any changes made in one script will result in the session data being changed for all other pages. However, the limitations here are that sessions are much more short term than cookies, and sessions can expire after a certain amount of time has passed.

Slide 11: This slide shows the example of a PHP script that is using sessions to store some data. Again, this data is provided by the user and is accessible through the $_POST associative array, but the key elements (marked in bold) are the session start at the top, which ensures that any changes made to $_SESSION will propagate through the rest of the web application and the setting of the data itself using the $_SESSION variable.

Slide 12: As with the example for cookies, this slide shows what happens on the next page, and as with cookies, we will see that what we entered into the form gets sent along, via $_SESSION, to all script pages that make use of the session_start() directive at the start.

Slide 13: Sessions are mostly managed directly via the $_SESSION array, but two useful manipulations are the use of the unset function which will delete a piece of session data (and indeed, will delete anything from any associative array in the same way), and the session_destroy function which lets you end a session on demand. In the end, because they do the same thing, working out which of these two techniques to use is based on a pair of simple questions – does the client accept cookies, and is there a need to store persistent data on a longer basis than is permitted by sessions? It does not matter for this module which students choose to use, so they should use the one with which they feel most comfortable (although you may ask them to use a specific type for individual exercises to ensure that they can use both when needed).

Slide 14: The database that we use for this example is trivial, consisting of only an ID and a description. These two fields however will permit us to build an architecture for a web-based front-end that allows for dynamic querying and updating of the data. To simplify our development, we create a PHP page (setup.php) that deletes and creates our test table as needed. Students should be encouraged to create similar pages for projects that they create, as they allow for us to work with live data without having to spend time fixing inconsistencies that get introduced through careless coding.

Slide 15: This slide shows the PHP code for a simple setup page. It deletes the Things table if it is already there, then creates it (this way, we start off clean with no legacy data). It then inserts some basic testing data with which we can work. Browsing to this page will essentially do a 'clean install' of our database, which is valuable when developing data driven applications.

Slide 16: It is a trivial matter to have our Ajax frontend communicate with a PHP back-end, provided we are willing to ignore the benefits that come from ensuring a clear separation between presentation and content. We have not been separating these two thus far, with our PHP scripts often taking the role of asserting certain presentational details (such as displaying databases in tables). From this lecture onwards, we stop conflating those roles. Before we do though, we look at how to incorporate PHP presentational detail into an Ajax front-end.

Slide 17: This is the first part of a PHP script that queries the database we just created. Note that it makes use of the GET method to acquire user input – this is because it makes it simpler to create our Ajax queries.

Slide 18: This is the second half of the PHP script, and it handles providing the output from the database in the form of an HTML table.

Slide 19: This slide shows the Ajax front-end to this database – the response from the PHP script is simply placed on the web page in the fashion we saw in the previous topic. This works, but lacks elegance and maintainability; we should never assume in our application layer that presentation is going to be in a particular form. There is nothing to say that our PHP script might not be accessed via a desktop application which has no mechanism for rendering or interpreting the HTML we provide. We insist on the separation of roles for various reasons, the most important of which are increasing maintainability (if we change the way the application works, we do not need to also adjust presentation code unless it impacts on the communication between the two), and ensuring flexibility of access. Desktop applications are internet enabled these days, and many will make use of internet web services to provide extra content. We do not know in advance who will be making use of the web applications we build, and so it is our responsibility to ensure that the data provided is in as generic a format as is feasible. As such, while the architecture we have just shown will work, it is insufficient for building well-structured applications.

Slide 20: The XML discussion from a previous lecture is core to building an effective strategy for communication in N-Tier architectures. We communicate between the presentation and application layers through the use of XML to ensure generic representation and easy parsing. In PHP, we must manually construct a DOM representation of our data in order to transmit it (there are external libraries that do this, but we should learn how to do it ourselves first). This is done through the DOMDocument class.

Slide 21: The process of building a DOM tree is not complex, but it is **intricate** – we are appending child nodes to parent nodes and relating them to the root node. As such, for the process that is discussed, it would be valuable if you can construct a diagram for your students showing how each of the variables in the example program relate to the others. It is a diagram that can only make sense when constructed live, and so it is not possible to provide a suitable diagram in the lecture slides.

Slide 22: We begin the process by creating a DOMDocument object – this happens before we do anything else. Once we have that, we create a root element (that is done through the $doc variable that we have created) and then relate each of the fields in our database result set to that.

Slide 23:	This slide shows the construction of a DOM tree. For each record in our result set, we create a node (the XML tag for this will be 'thing', and we refer to it in the code as $node. This is essentially what will hold the main container for the record). We then create two separate subchild nodes – name, and description. To these, we append a text node (a node that contains only text) and append that to the subchildren. We then append these subchildren to the main node we created, and then that node to the root. As mentioned above, this is an intricate process and students will benefit from you actively diagramming each step.
Slide 24:	Once we have our DOM tree constructed, we call the saveXML method on our DOMDocument to create a text representation of the XML. This can then be served to our Ajax front-end in a form that is amenable to parsing.
Slide 25:	Before we can finish with our server-side coding, we need to incorporate a line of code that will tell Ajax to interpret the results we send as an XML document. We do this by using the **header** function before any other output is sent to the browser.
Slide 26:	From the starting state we get from setup.php, this is the XML document that is served up by our PHP script when it is accessed. This document will be available in the responseXML property of the Ajax request object once it has finished communicating with the server.
Slide 27:	We are now going to build the user interface using XML and Ajax rather than in PHP. In order to do this, we will need to manually parse out the various data elements that were provided, but the fact that DOM Is a tree makes it straightforward to do. To begin with, we will just duplicate the interface we had previously – the data will be presented in a simple HTML table.
Slide 28:	Ajax, like JavaScript, handles parsing out the DOM structure of the document for us. The XML document we use is the one we are provided with from responseXML of the request object. We can thus pass this into a function designed to construct our HTML table.
Slide 29:	This is the function that handles the construction of an HTML table to contain our data. Students have already been directed to read up on the DOM specification as part of a previous private study exercise, so we should not have to explain why we manipulate the elements in the way that we do. The key aspects though are that we create a list of nodes by using getElementsByTagName, using the tag 'thing' (this is the parent container for our XML records). We then iterate over each node, getting the ID and Description subnodes. As you will recall when we constructed the tree, the contents of these nodes had text nodes appended to them, so we use firstChild to get hold of the text content, and nodeValue to tell us what the actual content of the text node is. It is straightforward to hook our function to the Ajax object, because we simply pass the responseXML property into the function, and then use the return value of that to set the innerHTML of an element elsewhere on the HTML page.
Slide 30:	In order for this to qualify as a real web application, it has to let us do something. Because this is only an example application, our options are limited, but any database driven application should permit us to modify the content, and so that is what we are going to do. The basic structure is identical to what we have just explored – use a

PHP page to handle the database manipulation and querying, and then we update our front-end accordingly.

Slide 31:	First of all, we are going to improve the user interface a little by providing a list of valid IDs in a combo box, and then automatically populating a textbox based on that ID. In order to do this, we need to extend our PHP script a little so that we can simply request everything from the table.
Slide 32:	Populating a combobox is done in the same way we built our table, but we only need to get the ID tags from the XML data. In our HTML document, we have a select form element and it has the name "data". We are going to set its innerHTML property to incorporate the HTML that represents the list of valid ID entries.
Slide 33:	This is the function that populates our combo box. It is very similar in style to the table construction that we discussed in Slide 19.
Slide 34:	We bind this function into the onLoad event of our HTML page so that the first thing that happens when a page loads is that the combo box is populated with the valid ID values. The next step is to provide a function that lets us query the description associated with an ID. Rather than creating a new function each time that must handle the XMLHttpRequest initialisation, we will generalise setupAjax a little so that it can be provided with an external URL as a parameter. That way it can serve as the basis for several functions.
Slide 35:	This is the modified code for setupAjax and the navigateDatabase function – the latter constructs the URL, which is then passed to setupAjax. SetupAjax in turn sends the XML document that is received onto the updateFrontend function which will populate the textbox with the new description.
Slide 36:	This is the updateFrontend function, and it parses out the XML of the query, and places the contents of the description node for the first element (we ignore any further elements) into the textfield we have in our form.
Slide 37:	In order to reflect changes made to the database in our front-end, we need both a new JavaScript function, and a new PHP page that will handle performing the query. This slide shows the updateDatabase function that creates the query for setupAjax.
Slide 38:	This is the PHP script that will handle updating the database. We provide an ID and a Description as part of the request, and the script then updates the database accordingly.
Slide 39:	The last part of the system is the HTML itself which acts as the container. Note that we have three event handlers defined here – one when the document loads (defined in <body>), one where the combo box contents are changed, and another when the text box is **blurred** (this happens when an element loses focus).
Slide 40:	The end result of this is a web application that offers a seamless querying and editing experience. We never need to wait for new pages to be served up; we can work with it as if it were a standard desktop application.

Slide 41- 42:	Conclusion, Terminology and References

8.5 Laboratory Sessions

The laboratory time allocation for this topic is 2 hours.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Making use of the architecture discussed in the lecture, create an Ajax enabled web application linked to Andy's Autos that allows for creating, querying, browsing and manipulating a database of your own choice.

Each record should have a minimum of four fields and your Ajax application should permit users to modify any of these.

In addition, your front-end should allow users to create a new record, and delete the record they are currently browsing (the latter requiring confirmation via a message-box).

Suggested Answer:

As students have free reign to decide on the database they wish to create for this, no specific solution code is provided. The structure of their applications though should follow the format given in the lecture, with Ajax being used to create a dynamic front end, and PHP being used to provide data to the front end in XML format.

The full set of code from the lecture example is given here for reference.

Ajax-Enabled Frontend

```
<html>
  <head>
    <title>Javascript Stuff</title>
  </head>
  <body onLoad="updateComboBox ()">
    <h1>My First Javascript</h1>

    <script language="Javascript">


    function updateFrontend (XML) {
      var form = document.getElementById("mainForm")
      var elements =
XML.documentElement.getElementsByTagName("thing");
```

```
        var id, description;

        if (elements.length == 0) {
          document.getElementById ("id").innerHTML = "";
          form.description.value = "";
        }
        else {
          description = elements[0].getElementsByTagName
("description");
          form.description.value =
description[0].firstChild.nodeValue;
        }

      }

      function setupAjax(url) {
        var url;


        if (window.XMLHttpRequest) {
          // Code for modern browsers
          request=new XMLHttpRequest();
        }
        else {
          // code for older versions of Internet Explorer
          request = new ActiveXObject("Microsoft.XMLHTTP");
        }

        request.onreadystatechange=function() {
          if (request.readyState==4 && request.status==200) {
              if (request.responseXML) {
                updateFrontend (request.responseXML);
              }
          }
        }

        request.open ("GET", url, true);
        request.send();

      }

      function navigateDatabase (form) {
        var url;
        var id;

        id = form.data.value;

        url = 'query_content_xml.php?thing=' + id;
        setupAjax (url);
      }
```

```
    function updateDatabase (form) {
      var url
      var desc;
      var id;

      id = form.data.value;
      desc = form.description.value;

      if (desc.length == 0) {
        return;
      }

      url = 'update_content_xml.php?id=' +  id + "&description=" +
desc;

      setupAjax (url);
    }

    function updateComboBox() {
      var url;

      url = "query_content_xml.php";

      if (window.XMLHttpRequest) {
        // Code for modern browsers
        request=new XMLHttpRequest();
      }
      else {
        // code for older versions of Internet Explorer
        request = new ActiveXObject("Microsoft.XMLHTTP");
      }

      request.onreadystatechange=function() {
        if (request.readyState==4 && request.status==200) {
            var text = "";
            var elements;
            var id;
            elements =
request.responseXML.documentElement.getElementsByTagName("thing");

            for (i = 0; i < elements.length; i++) {
              id = elements[i].getElementsByTagName ("ID");
              text += "<option>" + id[0].firstChild.nodeValue +
"</option>";
            }
            document.getElementById ("data").innerHTML = text;
        }
      }
```

```
          request.open ("GET", url, true);
          request.send();

        }


        </script>

        <form name = "mainForm" id = "mainForm">
          <p>ID</p>
          <select id = "data" onChange="navigateDatabase(this.form)">
          </select>

          <p>Description</p>
          <input type = "text" name = "description"
  onBlur="updateDatabase (this.form)">
        </form>
      </body>
    </html>
```

PHP Script for Querying

```
<?
  header('Content-Type: text/xml; charset=utf-8');

  $host = "localhost";
  $user = "monke13_nccuser";
  $pass = "ncc1";
  $database = "monke13_ncc";

  $thing = $_GET["thing"];

  $connection  = mysql_connect($host, $user, $pass)
    or die ("Couldn't connect to database");
  mysql_select_db ($database);

  if ($thing) {
    $query = "SELECT * from Things where ID='$thing'";
  }
  else {
    $query = "SELECT * from Things";
  }

  $ret = mysql_query ($query, $connection);


  $num_results = mysql_num_rows ($ret);

  $doc = new DOMDocument();
  $doc->formatOutput = true;
```

```
    $root = $doc->createElement( "all_things" );
    $doc->appendChild( $root );

    for ($i = 0; $i < $num_results; $i++) {
       $row = mysql_fetch_array ($ret);

      $node = $doc->createElement( "thing" );

      $name = $doc->createElement( "ID" );

      $name->appendChild($doc->createTextNode($row["ID"]));

      $node->appendChild( $name );

      $description = $doc->createElement( "description" );

      $description->appendChild($doc-
>createTextNode($row["Description"]));

      $node->appendChild( $description );

      $root->appendChild ($node);

    }

    mysql_close($connection);

    echo $doc->saveXML();

?>
```

PHP Script for Updating

```
<?

    $host = "localhost";
    $user = "monke13_nccuser";
    $pass = "ncc1";
    $database = "monke13_ncc";

    $id= $_GET["id"];
    $description = $_GET["description"];

    $connection  = mysql_connect($host, $user, $pass)
      or die ("Couldn't connect to database");
    mysql_select_db ($database);

      $id = mysql_real_escape_string ($id);
    $description = mysql_real_escape_string ($description);
```

```
  $query = "UPDATE Things SET Description = '$description' WHERE
ID='$id'";

  $ret = mysql_query ($query, $connection);

  mysql_close($connection);

?>
```

8.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

As part of your ongoing journal exercise, you should research the following topics:

- XML parsing in Ajax.
- XML DOM construction in PHP.

Exercise 2: Encyclopaedia of You Revisited

Taking the content, you developed for exercise two in your private study last week, modify the code so that all the data is contained within a mySQL database. This database should then be queried via a PHP script, and converted across to XML for display in a front-end.

Suggested Answer:

Highly dependent on student content – no suggested answer.

Exercise 3: Presentation

Prepare a short, five-minute presentation on the results of your research from Exercise 1 above. If you have found out anything particularly interesting, you should focus on that as a priority.

Exercise 4: Revision

Review the lecture material and ensure that you are comfortable with everything discussed thus far.

8.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, partially as a way to share information with others in the class but also as a way to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

At this stage of the module, you should also introduce the assessed assignment to students. Assignments for the relevant assessment cycle are available from the NCC Education Campus (http://campus.nccedu.com). You will need to ensure that each student has a copy of the assignment and understands the requirements. Assignments would normally be submitted for marking during Topic 9 or 10, depending on how much time you feel you need for marking.

Exercise 1: Discussion of Private Study Exercise

Discuss as a group your solution to Exercise 2 in the private study.

Exercise 2: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short five-minute presentation ready to give to the rest of the class. There is no need for this to be especially formal - you are simply reporting on anything interesting that you found during your research, or pointing out especially useful resources on the topic. Bring your journal along to the class so that you can make a note of anything especially useful that your classmates have mentioned. This is a knowledge dissemination exercise; you are not being formally assessed on the style or content of the presentation.

Topic 9: Mobile Application Development Integration

9.1 Learning Objectives

This topic provides an overview of mobile application web development integration. On completion of the topic, students will be able to:

- Mobile application development and integration with a website
- HTML5 and mobile applications
- DOM framework
- Developing mobile apps.

9.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. It is strongly advised that lecturers implement the coded elements of the lecture in front of students so they can see a MySQL/PHP application being constructed. Slides with code on them are good points to stop and show students how they relate to the application you are building.

They will then develop the code during the tutorial and lecture sessions.

9.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

9.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- What are mobile apps?
- Why use a mobile application over a website?
- UI for mobile apps
- HTML5 and DOM
- Creating mobile apps
- APIs and linking mobile apps to mobile sites.

9.4.1 Guidance on the Use of the Slides

Slide 4: In this lecture, we are going to look at how we can make use of mobile applications to enhance the user experience instead of or as well as using mobile website. Users of mobile devices are generating demand for mobile applications rather than navigating through URLs on websites. There has been a huge increase in companies having websites, mobile sites and apps eg Amazon, eBay, Next and Debenhams are examples of businesses that have gone down this route. In Topic 3 we looked specifically at how to create a mobile website, here we are going to look at creating mobile applications which can be used as well as or instead of mobile websites.

Slide 5: A business would need to consider whether it wants a mobile application as well as a mobile website and it should seek user demand. When designing the initial feedback with user input the questions should be asked and at that point the business and developer should look at the value of also having a mobile application. Remember that mobile website can be accessed through responsive web design which was covered in Topic 3. The difference between a mobile website and mobile application is that the app is downloaded and installed onto a mobile device whereas a mobile site is just viewed through the device. Information and content can be downloaded and accessed through the app without an internet connection and it can also provide good quality information back to the information on how the app is used.

Slide 6: There are many benefits of having a mobile app, however the developer will need to consider whether they have the skills to develop this and what the cost of updating would be. A mobile app can be personalised and can also be used for complex calculations which maybe more difficult to code in a website. They could also be more secure eg banks now have apps with additional security features. If an app is used rather than a mobile website then features on the mobile device can be shared or accessed such as the camera and as seen above, off-line content can be provided without being on the network.

Slide 7: User Interface design is important when designing an app. In topic 2 we looked at some of the key considerations for a website and these should also be reviewed in this topic. UI for mobile applications also need to be considered (this is sometimes known as UX/design). Templates can be used to help with development or they can be created from scratch. There are lots of templates available for mobile app development and these can be found through searching on the web.

Slide 8: As with topic 2, there are lots of benefits of using good UI in mobile applications. One of the main considerations is that the app needs to meet user needs and be intuitive. The app needs to be usable and the UI needs to be responsive when using touch screen. The UI needs to be efficient when carrying out the functions and it needs to create minimum effort for the user. The UI needs to be clear and logical and the user needs to be able to see the functions and location the action buttons so that they can quickly navigate around the app. The UI should be familiar, it should use the same house style as the main site and it should be linked directly with content on the main site.

Slide 9: HTLM5 can be used to create an app in addition to other website functions. The advantage of using HTML is that it can be used across different platforms eg iPhone, android phone, windows phone etc. In addition to using HTML there are mobile frameworks which can be used to support development and these will be reviewed later in this topic. HTML5 is easy to use and reduces costs as payments are not being made to use specific mobile frameworks. There are some drawbacks of using HTML5 in that it can bring performance and security concerns and there are sometimes problems with accessing API interfaces. HTML5, however has a collection of pages which can be optimised specifically for mobile devices linked to semantic elements and multimedia components. Frameworks can be used to handle some of the issues that arise when developing an app. An example of a framework is DOM model.

Slide 10: DOM is the Document Object Model and this was discussed in an earlier topic. This model represents the structure of a web page. As the DOM becomes more complex this can take much longer to navigate and modify the app. A framework which allows DOM elements to be recycled will help when building HTML5 mobile applications as the DOM needs to be as simple as possible.

Slide 11: XSLT is a language which transforms xml documents into other formats and this can be useful both on mobile websites and also mobile applications. XSL stands for eXtensible Stylesheet Language and can be used as a style language for XML. This can change XML documents into HTML pages or plain text documents or pdf which can be viewed in different software applications. XSLT is a language which is used for expressing style sheets and is similar to CSS as it describes how to display an XML document on a website or in a mobile application. XSL can be used for complex formatting where the document can be displayed in different devices.

Slide 12: XSLT is more sophisticated than CSS and it is expected that both will remain. XSLT allows elements and attributes to be added and removed quickly. CSS as you recall is used to add styles to HTML elements and HTML uses predefined tags. XML does not use predefined tags and XSL describes how the XML element should be displayed on the mobile application. XSL consists of four parts:

- XSLT – the language which transforms XML

- Xpath – language used by XSLT to access different parts of an XML document
- XSL-FO – the vocabulary for specifying formatting semantics
- Xquery – language which combines databases, documents and web pages.

W3.org 2017. W3C [online] Available at [Accessed 26 November 2017]

Slide 13: This slide shows an example of a style sheet which has been taken from w3.org. This shows how XSL is used in the stylesheet and how different elements can be transformed depending on the element.

Slide 14: This slide provides an example of an XML document for a CD catalogue. This is referenced to the www3.schools.com website and if you navigate to this page you can see what this looks like through “try it”.

Slide 15: This slide provides an example of a XSL style sheet linked to the CD catalogue and XML document on the previous slide.

Slide 16: This slide provides an example of how the style sheet is linked to the XML document.

Slide 17: Another way to create the mobile application is through the use of an app builder. There are hundreds of different app builders available and these are some that are popular. These can be accessed and downloaded from the relevant websites (details on the references slide).

Slide 18: iBuildApp is a mobile application builder which contains a number of different app templates that can be used either free of charge or for a small fee. The software allows professional apps to be developed by editing the layout and design of the templates and by adding additional content. Once the app has been built then this can be published on the App Store which then means that users can download and access it.

Slide 19: PhoneGap is an Adobe software package which is open source framework which allows apps to be built using web technology. It is a familiar language as it can be used linked to HTML, CSS and JavaScript. The software includes apps which are already app-store ready which means that they can be easy to use/edit and simple for novice user to develop. It has a wide range of plugins that can be accessed to increase the capabilities of the mobile applications and it has lots of help and support through the developer community.

Slide 20: This slide provides information on how the software is installed onto a desktop computer and how a new project can be created. At this point you may wish to demonstrate this.

Slide 21: AppsGeyser is a simple software package which can be used to create mobile version of your existing website. It is really easy to use. All you need to do is add your URL into the software, choose colour templates and click ok. You can then preview what your website would look like on a mobile device. Have a look at this as it is one of the easiest ways to see what our site would look like on a mobile device.

Slide 22:	Application Programming Interfaces (APIs) are small software packages which contain programming instructions which are used to access web-based software or tools. These packages are produced by software companies to the public so that other developers can edit and create products that are powered by its service. There are lots of APIs available including from Facebook, eBay, Amazon and google maps. Amazon have an API which allows web developers to add into their website to allow their customers to access Amazon's product information. The web designer can use the API to link Amazon products, prices and buy-now options to their own website. This can be advantage for some ecommerce websites.
Slide 23:	This slide provides further examples of APIs that can be used in a website, such as GoogleMaps, BingTranslate, YouTube, Facebook and eBay. The link provides a full list of examples of APIs which are available in 2017. These can enhance the website and provide additional features which the web developer may have struggled to use in the past.
Slide 24:	The slide shows an example of an API which provides a Facebook login. Imagine a company has a login to register for the site. The company may offer the customer the option of registering directly with them or logging in through Facebook login. There are advantages and disadvantages of doing this, however it is a useful tool and for some users it means they only have to remember one login and password! This slide shows the code that would be used through the API to login.
Slide 25:	This slide goes a little further and shows the code to be used to add the Facebook login button to the website. This can also be incorporated through an API.
Slide 26:	An important consideration is how your mobile site and mobile application will talk to each other. For example, you may wish to link the mobile application to other mobile apps rather than URL through a website. It is much more user friendly and professional for the mobile app from your own company to link to content in other mobile apps (eg Amazon) rather than visiting the Amazon URL website.
Slide 27:	Users want to be directed to mobile app rather than a URL. AppLinks (Facebook product) is one method that these links can be created. We haven't got time really to look at this in any detail, however have a look at Applinks and see how the content in a webpage can be directed to a mobile application and viewed in this. Code can be added to pages which then drive the customers directly to other apps rather than the URL web pages.
Slide 28:	An example of when you could want to link to another mobile app would be above eg Amazon or it could be through a fee sample game. A developer might have created a game which is free to distribute through the app store. However certain features may only be accessed through purchasing additional content. Links can be created in the app to direct the user to the app store and specifically to the app to purchase the additional content. This is likely to result in more sales than not having a direct link and the risk of the user finding the additional content.

Slide 29:	Conclusion
Slide 30:	Terminology
Slide 31:	References

9.5 Laboratory Sessions

The laboratory time allocation for this topic is 2 hours.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Make a mobile application for Andy's Autos and try to link this to the main web page you have created.

Suggested Answer:

No suggested answer. The exercise is entirely dependent on the student's choice of web service.

Exercise 2:

Based on the mobile application you have created add in at least one API and create a direct link from the mobile application to another mobile application of your choice through the app store.

Suggested Answer:

No suggested answer. The exercise is entirely dependent on student's choice of web service.

9.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

As part of your ongoing journal exercise, you should research the following topics:

- Mobile application frameworks
- AppLinks
- APIs

Exercise 2: Mash-Up

Making use of two mobile application frameworks of your own choice, create a mash-up of these. Incorporate at least two different APIs that could be useful linked to the mobile apps.

Suggested Answer:

Highly dependent on what students select for their mobile applications, and thus it is not possible to give a suggested or even representative answer.

Exercise 3: Presentation

Prepare a short, five-minute presentation on the results of your research from Exercise 1 above. If you have found out anything particularly interesting, you should focus on that as a priority.

Exercise 4: Revision

Review the lecture material and ensure that you are comfortable with everything discussed thus far.

9.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, partially as a way to share information with others in the class but also as a way to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

Exercise 1: Discussion of Private Study Exercise

Discuss as a group your solution to Exercise 2 in the private study.

Exercise 2: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short five-minute presentation ready to give to the rest of the class. There is no need for this to be especially formal - you are simply reporting on anything interesting that you found during your research, or pointing out especially useful resources on the topic. Bring your journal along to the class so that you can make a note of anything especially useful that your classmates have mentioned. This is a knowledge dissemination exercise; you are not being formally assessed on the style or content of the presentation.

Topic 10: Web Services

10.1 Learning Objectives

This topic provides an overview of consuming web services. On completion of the topic, students will be able to:

- Understand Web API;
- Make use of SOAP-based web services;
- Make use of REST-based web services;
- RSS feeds
- Integrate data from two web services into a mash-up.

10.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. It is strongly advised that lecturers implement the coded elements of the lecture in front of students so they can see a MySQL/PHP application being constructed. Slides with code on them are good points to stop and show students how they relate to the application you are building.

They will then develop the code during the tutorial and lecture sessions.

10.3 Timings

Lectures:	2 hours
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

10.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- An overview of REST and SOAP
- SOAP based web services
- REST based web services
- RSS feeds
- Web API

10.4.1 Guidance on the Use of the Slides

Slide 4:	In this lecture, we are going to look at how we can make use of web services to create web applications that leverage existing data sets. Web services are prevalent on the Internet, and many of them are backed by substantial reserves of information. As individual developers, we cannot hope to compete with the accumulated data of organisations such as Google, so instead we make use of their web-services and incorporate them into our own applications.
Slide 5:	Web services tend to be based on the SOAP protocol which we look at further in this module. However, you may have heard of Web API. A Web API is an application programming interface and it is used to write code which interfaces with other code. A web service is a type of API that tends to use HTTP, whereas a web SOAP service doesn't always use HTTP, it may use SMTP instead. In this lecture we will be looking at SOAP and REST frameworks.
Slide 6:	There are two main systems for deploying web services on the internet: SOAP, which is the more mature of the two technologies, and the more lightweight REST framework. The former of these is heavily standardised, with web services being programmatically queried in a consistent way. For the latter, mostly what we get is an XML document without any specific formatting conventions.
Slide 7:	Both are based on using XML to transmit data between remote systems, but SOAP standardises the XML that is produced, right down to the name, type and order of the elements that are contained within a document. In this way, it is possible to programmatically extract data about a web service, such as what functionality it exposes, the type and order of parameters, and the kind of data that is to be returned. All of this is done as part of the SOAP 'envelope', which is a specific structure for XML documents consisting of the header and the body of a SOAP response.
Slide 8:	This slide shows an example of a simple XML document conforming to the SOAP envelope protocol. This represents a request sent to a server, and it consists of the method to make use of (GetStockPrice), and the parameter to that method (StockName). A similar XML document will be provided by the server as part of the SOAP negotiation between client and web service.
Slide 9:	We can mostly ignore the details of what is happening in the SOAP envelope – the important thing is that it exists because it is this that chiefly distinguishes SOAP from REST as a mechanism for remote access. SOAP documents are usually more complex than the example on Slide 5, and often include facilities for enhancing security and reporting on faults. The protocol supports all of this, and also supports layering other protocols on top. Additionally, SOAP does not presume a particular protocol will be used to transmit the document – we do not need to go through HTTP in order to make a SOAP request. This is a flexibility that can be tremendously useful in certain circumstances where implementing a full HTTP layer is not feasible.
Slide 10:	REST, on the other hand, is far more akin to the PHP scripts we have been using so far – accesses are primarily made via URLs with encoded parameters. REST makes use of the existing HTTP protocol, using its hard-coded methods (GET, PUT, POST, DELETE) to provide its interaction regime. Each of these verbs maps onto a particular kind of action. Some authors like to relate the four operations to the standard database CRUD system, where PUT = CREATE, GET = RETRIEVE, POST

= UPDATE, and DELETE = DELETE. While this is useful shorthand, it misses out much of the nuance of what the HTTP definitions actually mean.

Slide 11: GET is meant as a way to obtain information from the server where there should be no side effects (as in, all it does is return data – it does not impact on the state of the server at all). PUT is its polar opposite, and is supposed to add or modify data on the server without returning data back to the client.

Slide 12: Delete is straightforward – it is supposed to delete data on the server. The role of POST, however, is a little more complicated. Strictly speaking, the role of POST is related to the **idempotency** of the operation. An operation is **idempotent** if performing it once or performing it multiple times will result in the same state of the resource being manipulated. GET, PUT and DELETE are idempotent operations, while there is no obligation for POST to be so. These distinctions do not matter especially for this module, but students may have questions about the difference between the operations.

Slide 13: When we access a REST service, the type of action we perform (GET / POST / PUT / DELETE) will influence what functionality is triggered. However, because REST is not a fully specified protocol, there is no standardisation as to what is supported and how strictly it maps onto the HTTP definitions of the actions.

Slide 14: SOAP and REST co-exist quite peacefully as they both have their strengths and weaknesses. REST has a much lower overhead in terms of how much information must be transmitted (no need for the envelope, and no need for services to expose standard functionality since the REST architecture itself can be assumed to exist). SOAP, however, is a standard protocol and thus can be programmatically queried and manipulated in ways that are much more difficult for REST. Additionally, SOAP does not require a particular transport layer, meaning that in niche circumstances where having a full HTTP implementation is not possible, it can be used when REST cannot.

Slide 15: Both REST and SOAP are, from the perspective of the user, virtually identical (provided you do not need to manually manage SOAP envelopes). For the most part, REST is used as a lightweight framework whereby day-to-day API access can be provided, while SOAP has the edge in enterprise conditions where its additional power and flexibility are most beneficial.

Slide 16: The web service provided and our client are entirely separated from each other. As such, there needs to be a common way for us to find out what can actually be done with a web service. WSDL is that common mechanism – it defines the set of exposed methods, the parameters, the return types, and so on. When we wish to consume a web service with SOAP, we pass to it the link to the WSDL of the service and this is then used to build the interaction context we later access.

Slide 17: For our first example of making use of a web service, we are going to use SOAP to access the stock price of a company. The URL given on this slide takes you to a container page for the web service, including WSDL definitions and forms that permit you to try out the web service without first coding the interaction. While it would not be beyond us as individuals to keep track of prices in the stock market, using a web service to get this information means that we do not need to worry about writing the code or keeping the values up to date. Such is the benefit of web services – they allow us to make use of the data sets that other people have constructed.

Slide 18: PHP provides support for creating SOAP connections – all we really need to do is tell our SoapClient object where to find the appropriate WSDL file, and then we need to call the methods and parse the output. Parameters to the method call are provided as an associative array, where the name of the parameter (which is provided in the WSDL) is the key that maps onto our actual value.

Slide 19: This slide shows the code for making the connection to the SOAP service. Two features in particular need to be mentioned. The method called on the SoapClient object matches a method that is exposed in the WSDL for the service (GetQuote), and the GetQuoteResult field of the object that is returned follows the naming convention that it is <nameOfMethod>Result. If the method we called was 'QueryStock', then the results field would be called QueryStockResults. What comes out of this query is an XML document which we load into a DOM tree and then echo as our output. Not all results that are provided by a SOAP web service will be in XML, but most will.

Slide 20: The XML that is returned from the service looks like this – from this point on, we know exactly how to manipulate the document to achieve our ends. It is simply a variation on what we have done for the XML we created from our database queries.

Slide 21: However, the problem we have is that the format of the XML document is not defined in the WSDL. We have to manually inspect the returned document to see how it should be traversed. Luckily, because we now insist on a clean separation between presentation and application, we can simply invoke our PHP script manually and examine the XML document that we receive. As far as DOM is concerned though, it does not matter. DOM builds its representation directly from the XML it receives and it does not need a specification to do that.

Slide 22: This slide shows the code that constructs a table representation of the stock data we receive – or at least, a subset of it. This fits into the Ajax framework in exactly the way it has in earlier lectures, with the return value of this being used to set the InnerHTML property of an element on the HTML page.

Slide 23: To consume a REST web service, we follow the same basic system. However, because of the non-standard nature of REST, there is not a default library of objects provided by PHP to do this. Instead, we use a different set of tools that are designed to simplify remote access generally.

Slide 24: The library that permits us to access remote resources is called **CURL**, and we use this to negotiate our REST requests as if we were simply making use of an external web page. CURL permits generic access to all manner of remote resources, from web pages to FTP sites. To see this in action, we are going to use a different web service – one provided by Google.

Slide 25: We are going to use the Google Directions API for this example. This needs us to provide three pieces of information (we need to provide all three or we will get a REQUEST_DENIED error). We need to say where we are coming from, where we are going, and whether our data is coming from a device with a location server (it is not).

Slide 26: This is the code for creating the REST connection to the Google API. In this case, we hard-code two cities into the query string, and then parse what we got out of curl_exec to get our DOM tree. It is as straightforward as that, and in many respects identical to the PHP scripts we ourselves have been using.

Slide 27: The XML document that comes from this API is very large (you may want to show it to your students in the class, but the editing required for it to fit onto a PowerPoint slide results in it being simultaneously unreadable and unrepresentative). For our purposes, we are going to extract three pieces of information – the html_instructions (these are the descriptions of what people should do), the Distance (which has two children, 'value' and 'text') and the Duration (which also has two children: 'value' and 'text'). As with our SOAP service, we already know how to traverse this.

Slide 28: Part of the power of web services is that they expose lots of useful data to us, but another part of their power is that they can be used to integrate data in a way that adds genuine value. Think of applications that layer interesting locations onto Google Earth, or others that combine reviews of products from many different sites, or others still that do price comparisons across the entire internet. All of these are done by combining data from multiple sources and integrating them in a way that provides new value. These applications are called **mash-ups**, and we are going to do a simple mash-up here to show the power of the concept. We are going to combine a web service that provides the real-world location of an IP address, and feed that into the Google Maps API to give the directions from the user's location to some fixed point (in the example, it is London).

Slide 29: This slide shows the first step of this process – getting the IP address of the user, and then making a REST request of the geo-location web service by passing to it that IP address. We then parse out the longitude and latitude from the XML document we are given.

Slide 30: We take that latitude and longitude and we feed it into the Google API. The result is that these two disparate web services integrate in a way that neither were specifically designed to do.

Slide 31: This is the code for creating the table of our directions. This should be familiar to the students by now; the only new feature is that we must traverse the tree a little deeper in order to explore the sub nodes of Distance and Duration.

Slide 32: All of this flexibility is due to two things – proper architecture in our applications, and the relative ubiquity of XML as a data protocol. It is to support distributed interaction of remote elements that we make use of these things. If people wrote web services the same way we wrote our first PHP scripts, none of this would be possible because it would be too difficult to extract the information we want from the HTML encasing it. Additionally, if we managed to do it for one web application, we would need to do it all over again for the second. The N-Tier architecture and XML ensure that there is compatibility between all the elements of what can be very complex multi-part applications.

Slide 33: However, web services do not come as an unmitigated good – they are often costly to access (for the higher end services), and those that are free usually have access restrictions on them. However, there is a lot of 'value added' that comes from making use of good, reliable web services.

Slide 34: Some of the functionality we may need to perform on a day-to-day basis is only feasibly available through web services. Very few of us have access to credit card validation routines, or a database that precisely maps IP addresses to cities. As such, if we want this functionality, we often need to outsource it. The danger though is that we cannot necessarily rely on web services. Sometimes they are moved or deleted. Sometimes they have connection or capacity issues. Sometimes the terms and conditions under which we access them change. There are many web services out there, from the quirky and personal to the heavy-duty commercial deployments of the big-name internet companies. Most of the everyday names on the internet have web APIs that are available and you can often make use of their functionality rather than coding it yourself. It is very easy to permit Facebook to authenticate users on your websites for example, but in the process, you have to ask yourself how much control you are willing to give up, and how much you want to bind yourself to another organisation.

Slide 35	A full range of free and commercial web services is available at www.programmableweb.com and this should be reviewed to see the different services which are available.
Slide 36:	RSS (Rich Site Summary) are messages which are delivered from different websites include news-related sites and online publishers. These allow users to stay up to date with information as the user does not have to visit the site to read the information as the “feeds” are delivered to show the latest content. The information that is distributed is read through a news reader and therefore this needs to be available to read any articles. Most browsers include this feature then the user can choose which content to add to their site. For example if you wanted to add a RSS feed from the BBC News Entertainment stories then you would visit the relevant section of the BBC website and you would click on the orange RSS feedbutton.
Slide 37:	Conclusion
Slide 38-39:	Terminology

10.5 Laboratory Sessions

The laboratory time allocation for this topic is 2 hours.

Lecturers’ Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Making use of a plug-in of your choice from this list of REST-based web services, create a PHP script that consumes this resource, and an Ajax front-end that presents it to the user. Also add an RSS feed from the BBC website into Andy’s Autos Website.

- http://www.programmableweb.com/apis/directory/1?protocol=REST&format=XML

Suggested Answer:

No suggested answer. The exercise is entirely dependent on the student’s choice of web service.

Exercise 2:

Making use of a plug-in of your choice from this list of SOAP-based web services, create a PHP script that consumes this resource, and an Ajax front-end that presents it to the user. It should be a web service that performs a different task to the one you selected for the first exercise.

- http://www.programmableweb.com/apis/directory/1?protocol=SOAP&format=XML

Suggested Answer:

No suggested answer. The exercise is entirely dependent on student's choice of web service.

10.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

As part of your ongoing journal exercise, you should research the following topics:

- REST versus SOAP
- Web services
- Mash-Ups
- WSDL

Exercise 2: Mash-Up

Making use of two web services of your own choice, create a mash-up of these whereby the data from both plug-ins work together to produce a result greater than any can achieve on their own.

Suggested Answer:

Highly dependent on what students select for their web services, and thus it is not possible to give a suggested or even representative answer.

Exercise 3: Presentation

Prepare a short, five-minute presentation on the results of your research from Exercise 1 above. If you have found out anything particularly interesting, you should focus on that as a priority.

Exercise 4: Revision

Review the lecture material and ensure that you are comfortable with everything discussed thus far.

10.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

This tutorial is designed to give students a chance to present the more interesting of their findings from their private study work, partially as a way to share information with others in the class but also as a way to ensure that they are indeed keeping a journal and researching the topics provided. You should encourage students to bring along their journals to these sessions and make notes on any points of interest that are raised by their classmates.

Exercise 1: Discussion of Private Study Exercise

Discuss as a group your solution to Exercise 2 in the private study.

Exercise 2: Reporting Back to the Class

As a result of the research you did during your private study time, you should have a short five-minute presentation ready to give to the rest of the class. There is no need for this to be especially formal - you are simply reporting on anything interesting that you found during your research, or pointing out especially useful resources on the topic. Bring your journal along to the class so that you can make a note of anything especially useful that your classmates have mentioned. This is a knowledge dissemination exercise; you are not being formally assessed on the style or content of the presentation.

Topic 11: Building A Dynamic Website

11.1 Learning Objectives

This topic provides an overview of previous content in context.

On completion of the topic, students will be able to:

- Analyse a real-world scenario;
- Integrate tools to develop a solution to the scenario;
- Consider security issues;
- Test the website.

11.2 Pedagogic Approach

The pedagogic structure for this topic includes some lecture but most of the lecture is built around the students practicing the skills they have developed so far. The Lecture gives an overview of the scenario and the tools that have been previously introduced, but it also looks at some of the security issues and testing of the website to ensure functionality. Students should be given paper or electronic copies of the scenario to work from but they should not be given copies of the answers – those should be discussed within the tutorial. The scenario will not cover all of the required functionality – some of this will be left to students in their private study time.

11.3 Timings

Lecture:	1 hour
Laboratory Sessions:	4 hours
Private Study:	7.5 hours
Tutorials:	1 hour

11.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Analysing a real-world scenario and creating a design solution;
- Integrating tools to develop a solution to the scenario;
- Consider security issues when building the solution;
- Testing the website.

11.4.1 Guidance on the Use of the Slides

Slide 4: This lecture and topic allows the student to practice and draw together what they have learnt so far. They are provided with a real-life scenario which allows them to practice the skills and solve a problem for a client. They will integrate different web tools and techniques that they have viewed in the previous lectures. They will also gain an understanding of some of the security issues to be considered during web development. They will be introduced to testing the website in different browsers and for technical correctness.

Slide 5: This slide outlines the scenario that the students have been provided with and this can be handed to the student. Discuss with the students the scenario and ask them to mindmap what they think should be included in the site. Ask the students to work in groups and present their findings back to the main group. From here they should then consider the design of the website.

Slide 6: This slide outlines the different tools that the students have been introduced to so far in the course and they should look to include as many of these tools and techniques as possible in their “mock up” website. Students should be given plenty of time to create their website.

Slide 7: This slide introduces a number of security issues and common vulnerabilities with websites. This list is not exhaustive and students should regularly review and look for some of the latest security issues and vulnerabilities as they develop their website. This identifies common vulnerabilities and problems in 2017.

Slide 8: SQL injection is a common website vulnerability in year 2017. SQL injection is a code injection technique which is used to attack applications that are data driven and they are inserted into entry fields. These usually work through exploiting security vulnerability in application software. This type of attack allows data to be tampered with, disclosed or deleted. In 2007 Microsoft UK website was attacked and defaced using SQL injection.

Slide 9: Another website vulnerability is malicious file upload. This can occur when websites allow users to upload files. There are different ways to try and reduce this vulnerability. One way is to ensure that the user has signed in and has authentic access to the system and upload the files. Another way is to sanitise the file name or contents. We looked at how to sanitise in an earlier lecture. Another way is to restrict the filetype to be uploaded, for example if you only want users to upload images then you can restrict the mime through an array to only accept .jpg, .gif or .png.

Slide 10: Restricting access is one way to protect the information on your website. The main methods to restrict access is through sign in using UserID and password or by IP address. Restricted pages can be stored in a single directory and can be controlled by a file called .htaccess which is placed in this directory to restrict access to certain pages. Another way is to restrict access by IP address eg by region.

Slide 11: Website logging actions record user actions when they have logged in. Each user logins in with their details and actions are logged to ensure that users are accountability for their own actions. Some websites would create a table in a database to log user activity and would write a function which filters user, time and activity types. Activities which can be seen as malicious actions can be flagged and then the user could be logged out.

Slide 12: This slide introduces cybersecurity. This is a massive topic in 2017 and is likely to grow with specific units and qualifications specifically linked to cybersecurity. We will just be having a short introduction to this topic but students should be encouraged to find out the specific regulations and legislation that operate within their own country. Cybersecurity is becoming a massive issue for organisations across the world. Due to recent cyber-attacks which have disabled large organisations or has allowed terrorist activities to develop then lots of policies and procedures have been introduced to try to overcome some of these problems. Cyber actors deliberately go out of their way to steal information and money to destroy or threaten the delivery of services. In May 2017, the NHS in UK was hit by a cybersecurity attack where many staff within the NHS were unable to access their IT systems with some being shut down as a precautionary measure. This had a major impact on NHS with hospitals and GP being unable to access patient data after their computers were locked by ransomware and the cyber actor demanding a payment worth £230m. Cybercrime is growing and traditional crimes such as child pornography are being perpetrated through cyberspace.

Slide 13: Cyberspace is difficult to secure due to the links between cyberspace and physical systems and because the malicious actors can operate anywhere in the world using malicious software to damage cyber infrastructures. In September 2017 a meeting was held in New York to review stopping the spread of terrorist content. New technology solutions are to be developed to stop content being uploaded to the internet. Facebook, Microsoft and Twitter have agreed to establish a new Global Internet Forum on Counter Terrorism which we will see more of in 2018.

Slide 14: The UK government and other governments across the work have created Cyber Security Strategies. The UK has a National Cyber Security Strategy which has identified four main measures to try and address cyber security. Protected DNS (Domain Name Service) is the phonebook on the Internet. A DNS filtering service has been created to protect public sector bodies by blocking users. DMARC Anti-spoofing allows government departments to block communications which are not genuine. DMARC authenticates organisations communicate as genuine and malicious and fraudulent emails are blocked. Web Check is a free service that has been built to scan public sector body websites and produce reports on what needs fixing and also how to fix any vulnerabilities. Phishing and malware protection has been introduced to UK government departments through Netcraft, who are a private sector company who can issue takedown notifications to hosts of email and physing sites. In 2017, this amounted to 62,849 attacks (nscs.gov.uk, 2017).

Slide 15: Secure Socket Layer is technology that is used to establish a secure link between a web server and browser. This technology has been created to allow sensitive information such as credit card information, national insurance numbers and login credentials to be protected when it is sent over the internet. Both the browser and sever need a SSL certificate to ensure that a secure connection is established. Millions of pieces of sensitive data is transmitted over the internet every day through online transactions and one way to check the security is to ensure websites begin with https rather than http.

Slide 16: Secure Socket Layer certificates are used to establish secure connection. A website displaying a secure connection shows a lock icon or a green bar. SSL certificates have a key pair: a public and a private key. These keys work together to establish the encrypted connection. Servers need to be running Internet Information Services with certificate authorisation which can be achieved through a globalsign. Web Developers can create their own certificates by creating a folder on their server where all secure files are kept. Web hosting companies can also provide a SSL secured server – some have a charge and some are free of charge. Developer's should check that they are provided with a URL reference that routes through their secured server to a secured folder.

Slide 17: Encryption is a method that is used to encode a message. Websites that do not encrypt information are likely to lose customers as users want to feel that their information is kept safe. SSL encryption is necessary for any ecommerce website to ensure that data and personal information is exchanged securely. As we have seen in the previous slides SSL certificates are one way of providing this encryption. RSA (Rivest-Shamir-Adleman) is a widely used secure data transmission encryption technology.

Slide 18: If you have your own server then you can setup SSL within the server environment. This slide reinforces the information from slide 16 and brings together a summary of encryption and security.

Slide 19: This slide provides an overview of the different activities to be completed through the scenario and this should be discussed with the students and how they will design these activities.

Slide 20: This introduces the next section of this topic which looks at website testing and three key requirements in terms of technical correctness, browser compatibility and standards compliance.

Slide 21: In many cases, ensuring technical correctness is simply a variation on what is done with a standard desktop application; we conduct formalised testing on units before we then test the integration of these units. Students should already be familiar with this from previous modules, and it will not be covered again here. However, websites also introduce a wide variety of browser specific issues that must be identified and then addressed.

Slide 22: Browser compatibility is the main concern when building dynamic websites, because while the situation has improved over the past decade, it is still not the case that a web page will look and behave correctly across all browsers without significant developer attention. The more complex a web application is, the more intricate the browser corrections will be. Almost 90% of the market share for browsers is taken up by the Big Three: Internet Explorer, Firefox and Chrome. As such, checking the correctness of a web application should ensure at a minimum the correct functioning across these three browsers. By incorporating testing for Safari and Opera, coverage can be take up to around 98%, with niche browsers such as Opera Mini, Lynx and such making up a tiny percentage.

Slide 23: Much of what defines Ajax and JavaScript is standardised; it is not fundamentally different to write applications for these different browsers. However, where the incompatibility comes in is in the fine detail – how browsers handle things that are not explicitly stated in the specifications. The white space issue is of particular concern when working with XML documents, because the size of nodes in a DOM tree will differ between browsers and that can result in inaccurate functionality or even entirely broken subsystems of a web application.

Slide 24: In addition to the browser specific issues, there are other concerns that will affect people making use of a web application. If we mandate the use of certain fonts, they may or may not be available on the user's system. Monitor resolutions can cause huge problems with presentation if the mark-up does not compensate for it in some way. Additionally, add-ons may cause complications in those browsers that support them. Browsers use different engines to render HTML, which can cause subtle (or substantial) differences in how the web pages look and behave. As mentioned in a previous lecture, some browsers handle user input differently, such as automatically escaping quotation marks and apostrophes before sending them via POST or GET. It is impossible to test for all combinations of problems across all browsers, but we must aim to test as much as we reasonably can.

Slide 25: When doing the testing of a web application, we have to ensure that we test with at least the three major browsers, across multiple resolutions, and both with and without common add-ons. The list of subtle browser incompatibilities is considerable, and we could profitably spend several lectures talking about them. However, all we can do in this lecture is to highlight the issues and then direct students in their private study time to explore the subject in more detail.

Slide 26: We can minimise our risk of browser problems though by following a few simple rules. Bleeding edge technology (such as HTML 5) usually takes a couple of years before it settles down into a useful standard. There are browsers that support the prototype specifications for HTML 5, but that standard is still under development. Anyone building serious applications on top of the specification is liable to find problems down the line. As such, we should keep away from technology that is experimental or untried. We should also make sure that we use only well supported functionality, and shy away from proprietary extensions. When the world wide web was still young, Microsoft adopted an 'enhanced' version of HTML for its Internet Explorer browsers in the hope that the extra functionality would ensure browser dominance. If we want to avoid browser incompatibilities, we must avoid introducing them ourselves by making use of partially supported functionality.

Slide 27: We use a lot of standards in the web applications we are developing thus far. Good standards are the core of communication between web applications and subsystems. However, it is possible to get the impact of following a standard without actually following it correctly – we can output XML that is 'good enough' for our purposes, but the power of a standard is that we can interpret it everywhere and in every context. If fixed standards are followed only partially or inconsistently, we lose that benefit.

Slide 28: There are all sorts of reasons why this happens. Sometimes standards leave significant sections for implementers to decide on (this is known as being **implementation defined**). Sometimes (such as with Microsoft's proprietary HTML tags) the standards are not followed because they are actually a competing superset of the standard and not the standard itself. Sometimes they are too complex, and people simply ignore this complexity in favour of having something that works, rather than something that is 'correct'.

Slide 29: For a long time, the HTML standard was only half-heartedly enforced. Browsers would be very forgiving of problems, up to the point where they would 'assume' what you meant if your mark-up did not actually make any sense. Unfortunately, that process of interpretation varied wildly – some browsers did not try to interpret, and those that did often interpreted the mark-up differently from other browsers. Browsers are still relatively forgiving, but we are moving more forcefully towards an internet where standards are considered inviolable. The problem that this forgiveness created was that developers, on the whole, look to other developers to see how to do things. Much of programming is inspired by the developers around us, whereby we look to find code that does what we want and then adapt it to meet our needs. That is a perfectly healthy strategy, but it does mean that incorrect solutions tend to propagate. The flawed code that is only partially correct is used by other developers as the basis for their own creations that perpetuate the flaws.

Slide 30: Most of the standards are too complex for us to manually ensure compliance, so we use external validators to ensure that we properly adhere to the specification. There are many of these, but the most useful on a day-to-day basis is the W3C validator, which will take a URL and provide a list of the mark-up flaws (there are usually a huge number of these!).

Slide 31: The process for ensuring standards compliance requires us to be critical each time we make use of a standard, and make use of whatever tools we can to ensure its correctness. There are dozens of web validators for all sorts of issues, but most standards also include some method of ensuring validity – XML for example has DTDs and schemas that can be used to ensure that the XML documents we create and produce conform to a set standard. Additionally, when developing we need to make sure that our browser is as strict as it can be in interpreting standards. Many browsers have a toggle for this, and it is usually set as permissive as it can be. As a developer, we want the browser to catch as many errors as it can as we go along so that we can fix them.

Slide 32: Conclusion

Slide 33: Terminology

Slide 34: References

11.5 Laboratory Sessions

The time allocation for the laboratory sessions for this topic is 4 hours.

Lecturers' Notes:

Students have copies of the laboratory activities in the Student Guide. Answers are not given in their guide.

During the laboratory sessions, students should work on the activities outlined below in groups of two or three. Students have the responsibility for subdividing labour, but you should encourage them to ensure that every member of the group has a task in each of the activities. Do not give any answers during this session, although you can provide guidance and advice. Much of the example code given in previous topics will solve parts of the case study, but the responsibility for integrating all of these tools lies with each of the groups.

All the suggested answers associated with this topic are skeleton implementations – students will be expected to add in the fine details such as data validation and presentational flourishes.

Case Study

Stocks 'r' Us is a stock trading site. Recently, clients have asked for a mechanism by which they can check the current value of their stock portfolios without having to call their broker. You have been commissioned to develop the site that would permit people to record their stock purchases and find their value at any given time.

The sensitivity of a financial portfolio is such that you have been asked not to develop your own bespoke login system for this site, but instead use the OpenID standard – specifically, users should authenticate against the Google OpenID server and the ID information that is sent back should be used to uniquely identify users in the system. You have been asked to use the PHP class available at https://developers.google.com/identity/protocols/OpenIDConnect to create your connection to the Google OpenID servers

The first pass over the software is intended to be a prototype so that it can be trialled by senior managers, and so only a limited set of functionality is required. Later extensions will be requested, but not for this class exercise. The set of functionality you must provide is as follows:

- Allow people to log in to the system via OpenID
- Allow people to add stocks to their portfolio
- You will need to store this information in a database
- Provide users with a real-time update of stock prices (using the web service that you have used in previous topics).

This is intended to be a flagship product for the company, and as such you are asked to ensure that the system works appropriately across browsers and uses the most current technologies to accomplish your goals. Appropriate use of jQuery and Ajax have been marked down as a key deliverable of the product.

Activity 1:

Create a setup page for your stock portfolio application.

Suggested Answer:

The table for the setup should contain a field for using the OpenID of the user, but should also store at a minimum the number of stocks an individual has and the company in which they have those stocks. The following is an outline implementation of a setup page:

```
<html>
  <head>
    <title>Case Study Setup</title>
  </head>

  <body>
    <?

    $host = "localhost";
    $user = "monke13_nccuser";
    $pass = "ncc1";
    $database = "monke13_ncc";
    $connection  = mysql_connect($host, $user, $pass)
      or die ("Couldn't connect to database");
    mysql_select_db ($database);

    $query = "DROP TABLE StockData";

    $ret = mysql_query ($query, $connection);

    $query = "CREATE TABLE StockData( OpenID varchar (100), Stock
varchar (15), Amount INT)";

    $ret = mysql_query ($query, $connection);

    if ($ret) {
      echo "<p>Table created!</p>";
    }
    else {
      echo "<p>Something went wrong: " . mysql_error(); + "</p>";
    }
    ?>
  </body>
</html>
```

Activity 2:

Create the PHP code required to query the database for a particular user. This should return the data in XML format and should include no presentational detail of its own. This page should also be responsible for querying the stock price web service for the current value of the stock.

Suggested Answer:

```
<?

  header('Content-Type: text/xml; charset=utf-8');

  $host = "localhost";
  $user = "monke13_nccuser";
  $pass = "ncc1";
  $database = "monke13_ncc";

  $thing = $_GET["id"];

  $connection  = mysql_connect($host, $user, $pass)
    or die ("Couldn't connect to database");
  mysql_select_db ($database);

  $query = "SELECT * from StockData where OpenID='$thing'";
  $ret = mysql_query ($query, $connection);


  $num_results = mysql_num_rows ($ret);

  $soap = new
SoapClient("http://www.webservicex.net/stockquote.asmx?WSDL");

  $doc = new DOMDocument();
  $doc->formatOutput = true;

  $root = $doc->createElement( "all_stocks" );
  $doc->appendChild( $root );

  for ($i = 0; $i < $num_results; $i++) {

    $row = mysql_fetch_array ($ret);

    $node = $doc->createElement( "stock" );

    $name = $doc->createElement( "stock_name" );

    $name->appendChild($doc->createTextNode($row["Stock"]));

    $node->appendChild( $name );

    $result = $soap->GetQuote(array('symbol' => $row["Stock"]));

    $stock_xml = new DOMDocument();

    $stock_xml->loadXML ( $result->GetQuoteResult );
    $price = $doc->createElement( "stock_price" );

    $last = $stock_xml->getElementsByTagName ("Last")->item(0)-
>nodeValue;

    $price->appendChild($doc->createTextNode($last));

    $node->appendChild( $price);

    $amount= $doc->createElement( "stock_amount" );

    $amount ->appendChild($doc->createTextNode($row["Amount"]));

    $node->appendChild( $amount);

    $root->appendChild ($node);

  }

  mysql_close($connection);
```

```
       echo $doc->saveXML();


     ?>
```

Activity 3:

Create the PHP page for updating the stock portfolio. If someone enters a stock that they already possess, it should update the entry rather than adding a new record insertion.

Suggested Answer:

```
<?

  $host = "localhost";
  $user = "monke13_nccuser";
  $pass = "ncc1";
  $database = "monke13_ncc";

  $id= $_GET["id"];
  $stock= $_GET["stock"];
  $amount= $_GET["amount"];


  $connection  = mysql_connect($host, $user, $pass)
    or die ("Couldn't connect to database");
  mysql_select_db ($database);

  $query = "SELECT * from StockData where OpenID ='$id' and Stock
='$stock'";

  $ret = mysql_query ($query, $connection);

  $num_results = mysql_num_rows ($ret);

  if ($num_results > 0) {
    // We need to update a stock number here.
    $row = mysql_fetch_array ($ret);

    $amount = $amount + $row["Amount"];

    echo "Amount is $amount";

    $query = "UPDATE StockData SET Amount = '$amount' WHERE OpenID='$id'
AND Stock ='$stock'";

    $ret = mysql_query ($query, $connection);
  }
  else {
    $query = "INSERT INTO StockData (OpenID, Stock, Amount ) VALUES
('$id', '$stock', '$amount')";

    $ret = mysql_query ($query, $connection);

  }

  mysql_close($connection);

?>
```

Activity 4:

Create the front-end for the application.

Suggested Answer:

```
      <?
        session_start();
        $user_id = $_SESSION["user_id"];
      ?>

      <html>
        <head>
          <title>Case Study Main Interface</title>
        </head>
        <body>

          <script src="http://code.jquery.com/jquery-1.6.1.js"></script>
          <h1>Logged in as <? echo $user_id; ?></h1>

          <h1>Your Stocks</h1>

            <p id = "stock"> </p>


         <form id = "new_stock">

           <p>Stock Code</p>
           <input type = "text" id = "stock_name">
           <p>Amount</p>
           <input type = "text" id = "amount">

           <input type = "submit" value = "add">

            <script>

        function parseContent (xml) {
          var text = "";
```

```
    text += "<table>";
    text += "<th>Stock Name</th>";
    text += "<th>Stock Amount</th>";
    text += "<th>Stock Price</th>";
    text += "<th>Stock Value</th>";

    $(xml).find("stock").each (function() {
      text += "<tr>";
      text += "<td>" + $(this).find ("stock_name").text() + "</td>";
      text += "<td>" + $(this).find ("stock_amount").text() +
"</td>";
      text += "<td>" + $(this).find ("stock_price").text() +
"</td>";
      text += "<td>" + $(this).find ("stock_price").text() *
$(this).find ("stock_amount").text()+ "</td>";


      text += "</tr>";
    });

    text += "</table>";
    $("p#stock").html (text);
  }

  function updateContent() {
      $.ajax ({
        type : "GET",
        dataType: "XML",
        url : "query_stocks.php",
        data : {"id" : $user_id},
        success : parseContent
      });
  }


    $(document).ready(function(){
      updateContent();

      $('#new_stock').submit(function() {
          $.ajax ({
            type : "GET",
            dataType: "XML",
            url : "add_stocks.php",
            data : {"id" : $user_id, "stock" :
$("#stock_name").val(), "amount" : $("#amount").val()},
            success : updateContent
        });
      })

    });
    </script>
  </body>

</html>
```

Activity 5:

Incorporate the OpenID authentication system and the PHP class you were provided.

Suggested Answer:

First Page

```
<html>
  <head>
    <title>Case Study Login</title>
  </head>
  <body>

    <a href="login.php">Login with your Google Account</a>
  </body>
</html>
```

Login.php

```
<?
  require_once 'opened';

  $googleLogin                                                    =
GoogleOpenID::createRequest("ncc/case_study/return.php");
  $googleLogin->redirect();
?>
```

Return.php

```
<?
  session_start();

  require_once 'openid.php';

  $googleLogin = GoogleOpenID::getResponse();
  if($googleLogin->success()){
    $user_id = $googleLogin->identity();
    $_SESSION["user_id"] = $user_id;
  }

  echo "<a href = \"frontend.php\">Go to Main Interface</a>";
?>
```

Activity 6

Test the website for:

- Technical Correctness
- Browser Compatibility

- Standards Compliance

Keep relevant screen shots and notes of your testing.

11.5 Private Study

The time allocation for private study in this topic is expected to be 5.5 hours.

Lecturer's Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Write a short reflection on the web site you have created.

- What are the strengths?
- What are the weaknesses?
- What security measures have you taken into account?
- What were the results of testing the website?

Exercise 2:

There are many extensions you can make to the stock tracking application. Implement the following:

- Apply a jQuery theme via ThemeRoller
- Incorporate data validation
- Ensure that stocks actually exist before they are added
- Allow users to delete stocks from their list
- Use the Google Charts API to create graphs showing the proportion of total value each of the stocks take up.

11.6 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.
During this tutorial, you should discuss the worked solution to the scenario, providing the code that has been provided. This solution will differ from that of the students, but it will serve as a baseline for the discussion and you can profitably incorporate student feedback into your discussion. You may also wish to discuss the extensions students put together during their private study. This tutorial is also designed as an opportunity for students to raise any last questions they may have about the module. During their private study, students should have been making a note of questions to ask, and you should deal with those questions as they are raised.

Exercise 1: Overview of Scenario Solution

Discuss the solution to the exercise that your lecturer provided.

Exercise 2: Reporting Back to the Class

As a result of the work done for your private study, you should have implementations for each of the tasks outlined. Be prepared to discuss your solutions with others in your class, and to help them to refine and correct the code they have developed.

Topic 12: Evaluating Websites

12.1 Learning Objectives

This topic provides an overview of the issues of evaluating websites. On completion of the topic, students will be able to:

- Reflect on the use of the web application for a specific purpose;
- Reflect on the functionality of data driven website;
- Reflect on the web service solutions;
- Test for user satisfaction;
- Reflect on the business benefits of the web site/services;
- Understand the issues of accessibility support.

12.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the laboratory sessions and tutorials.

12.3 Timings

Lectures:	1 hour
Laboratory Sessions:	2 hours
Private Study:	7.5 hours
Tutorials:	1 hour

12.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- User satisfaction
- Suitability of web application
- Business benefit of the web site/services
- Accessibility

12.4.1 Guidance on the Use of the Slides

Slide 4: This lecture introduces the topic of evaluation by looking at the things that must be considered when reviewing a website. The lecture has three main parts, focusing first on the technical correctness (primarily browser compatibility), then on standards compliance, and finally on user satisfaction. While this is not a full treatment of the topic (any one of these has enough nuance to be worthy of a module in itself), it will highlight the issues and give students the necessary context to incorporate these elements into reviews they conduct of websites.

Slide 5: In the previous lecture we look at browser compatibility and technical correctness but in this lecture we will look at user satisfaction and how well the website meets the needs of users. If a website is technically excellent but does not meet user needs then it is not likely to attract users and customers, particularly if it is an ecommerce site. It is essential that the website does what the user needs and is accessible to users.

Slide 6: A relatively modern design strategy is that of 'User Centred Design'. In this system, users are included at each stage of development and asked to evaluate whenever possible. They are part of the planning, and are consulted after each milestone is reached. This requires a relatively large burden of involvement on the user, but the benefits are considerable – users get to see progress being made on a regular basis, and get a real chance to influence the direction of that progress. Developers get to evaluate on a regular basis how effective their applications are, and repeated testing usually has more benefit than larger tests that are conducted only once. Finally, the earlier changes are made the less costly they are – identifying problems early results in software that is developed more quickly and at less cost in developer time.

Slide 7: Conducting such tests in a rigorous manner is a specialised skillset, but even an amateur web developer can get a lot of benefit out of doing usability testing if they know what they are looking for. Mainly, we are looking at two things – the interface of the software as well as the **workflow**. Web page testing shows that the first few minutes of interaction are hugely important, since people will 'bounce' from a web page very quickly if they do not see what they want or like.

Slide 8: The main issue with creating an effective front-end is that users are willing to invest a lot less time than developers might expect. Usability studies show that users read about 20% of the text on a web page (e.g. Nielson, J. (2008). How Little Do Users Read? [Available Online] http://www.useit.com/alertbox/percent-text-read.html) and often ignore things 'below the fold' i.e. outside of a page's initially viewable area (e.g. Nielson, J. (2010). Scrolling and Attention. [Available Online] http://www.useit.com/alertbox/scrolling-attention.html). These may not be issues if you are making a website on request for an individual or an organisation, because they have a vested interest in using it. On the other hand, if you are creating the 'next big thing' and you want people you have never met to use it, you have to structure the page to ensure that in the first few minutes they can see what they need to see and also know what to do to make the next few interactions occur.

Slide 9: Ajax removes much of the need to worry about navigation in a web application (we do not need separate pages for everything; we can create a perfectly functional user interface in a single page). However, it does not remove the need entirely, especially for large and complex web applications. One of the key aspects of building user confidence is the ease with which they can navigate a site. Not all of the pages in a web application will be Ajax, and so you need to provide a mechanism by which users can know where they are and how to get back. Breadcrumbs can be a useful way of doing this. Tangentially related to this is the issue of spatial navigation – users have expectations of where certain common functionality options are going to be found (search boxes for example), and this is influenced by all the other websites out there. When incorporating something common, such as a navigation panel or site map, you should house it on the document where people will expect it to be.

Slide 10: Having managed to keep a user interested for the first few minutes, the more substantial task of assessing the functionality of the site can be undertaken. Robustness of the site relates to its ability to deal with technical failures beyond the scope of the code, such as invalid user input or network connectivity issues. The responsiveness of the site is a key factor in ensuring a good user experience, with sites that have high latency resulting in greater user frustration. The ease with which tasks can be accomplished (clicks to completion) is also a key indicator of the usability of a site, especially when high latency is involved. Much of what characterises web applications can be an exercise in frustration if not handled properly – failing to auto-complete standard input forms (names, address, email address), or doing external validation and then asking for new information while simultaneously clearing existing text fields. In the latter case, the cost of making a mistake is high, and we want our websites to be forgiving for users.

Slide 11: Web pages that remember their users in various ways (recommendation based on previous interaction is one common method) are often especially highly valued by users, because of the reduced cognitive burden required in order to explore new and relevant content. However, one must be wary of the need for a user to register before this can be done – registration that is required in order to unlock basic functionality is a barrier to entry for many users who will simply look elsewhere for something that meets their current needs. Additionally, many users (older users in particular) have concerns about trusting personal details to websites. Registration can be the most effective way of handing user persistence, but it comes at a cost for casual web interaction.

Slide 12: While formal user testing is often beyond the resources of individuals to perform, it can be done profitably with relatively small numbers of participants, and equipment no more sophisticated than a laptop.

Slide 13: There is a relatively straightforward process for conducting usability studies on a web application. First, we identify representative users from our intended audience (if we do not know who our intended audience is, it is time to go back to the requirements specification for the web application and rethink the system). Having identified representative users, you need to recruit. Jacob Nielsen (2000) suggests that five users are adequate for most testing purposes (http://www.useit.com/alertbox/20000319.html), but in order to gain the appropriate rigor required to make firm conclusions, more users are better. Having recruited the users, choose where the testing is to occur (in your office, in their home, somewhere neutral), and then conduct the testing.

Slide 14:	Testing usually focuses on a number of separate measures which are then brought together and analysed. The simplest kind of metric is the performance measure in which you set some tasks for the user and time how long it takes them to perform. You may also do things, such as counting clicks, summing the time spent reading, or even recording the behaviour of the eyes if you have eye-tracking equipment available. Users are usually encouraged to talk through what they are doing and why, as this information gives a 'stream of consciousness' insight into why people are doing what they are doing, and this can highlight problems or issues in the design. Coaching involves you as the developer instructing someone in using the interface, while questionnaires allow you to gather quantifiable data, albeit on self-reported measures. Accessibility experts have other methods they use, such as heuristic walkthroughs and cognitive assessment. These are beyond the scope of this module, and usually come only at a significant consultancy cost.
Slide 15:	Once the data has been gathered, it must be interpreted. In some cases, it will be straightforward, such as identifying an obvious deficiency encountered by all users. In other cases, it may involve a more substantial reworking of the design of the application, and in such cases, you may introduce more errors in the process of fixing the first ones. If you have a testing budget that allows for 20 people to be used for evaluating an application, the approach that gives most value is four separate testing sessions using five users each time. In this way, you capture the majority of user issues whilst also making the best use of the iterative approach of user centred development.
Slide 16:	Accessibility is often an afterthought when developing web applications, but it is becoming more and more important as the age demographics change across the world. It is important that students do not think of accessibility as a chore that must be performed to comply with legislation, but instead as an active and positive thing that increases the reach of the software they develop. Users with extraordinary accessibility needs are no different from normal users with temporary impairments (such as RSI), and an accessible interface is good for everyone.
Slide 17:	Once the website has been created it is essential that in addition to testing user needs it is essential that the system meets its original purpose. Thinking about the scenario in the previous lecture, students could think about these questions and answer them based on the website they have created. They could also peer assess each other's websites using these questions as a checklist/commentary to get independent feedback on their website.
Slide 18:	Laws such as the Equality Act in the United Kingdom state that if people with impairments cannot access data on a website, it can be construed as discrimination and thus is illegal. Such legislation too is **anticipatory** – you cannot just wait until someone has a problem. While the legislation is not international and different countries have their own view on the issue, anti-discrimination legislation, such as the Equality Act is becoming more and more popular around the world.
Slides 19-20:	Most of the things we need to do in order to ensure accessibility involve simply providing extra information for the accessibility support equipment that individuals may be using. In the case of blind users, we supply a little extra mark-up in our web

pages to assist screen readers. In the case of users with colour blindness, we avoid having colour critical information as far as is possible (it could be argued that the standard style of having followed links in red and non-followed links in blue is a violation of this rule).

Slide 21:	For dealing with mobility issues, it is a case of optimising our applications so that the workflow required to accomplish actions is as short as possible. This in itself is a bonus to all users, as the fewer clicks and mouse motions required to accomplish a task, the more efficient everyone becomes.
Slide 22:	Hearing impairments are resolved through avoiding the use of critical information that is provided only through sounds, and by ensuring any multimedia content that we provide is suitably captioned. YouTube for example permits for videos uploaded to the service to be close captioned so as to allow users to include a textual description of the audio information.
Slide 23:	Web technologies and testing is changing all the time. Students need to consider testing their website on big screens, tablets and mobile devices and they should do this for the website they created in the previous lecture. Notes should be made of any difficulties experienced and a plan written to show these could be overcome. Other considerations are mapping the user journey and customer profiling. Mapping the user experience and journey is becoming very popular in various industries. These demonstrate the ways that users currently interact with the website and demonstrate different ways they could interact with the website. Using a user journey helps understand user behaviour – how long a user spends on the time, what clicks they make, the different pages they visit and if they leave without making a purchase. Other considerations are testing frameworks such as Karma and Mocha. Karma is an example of a test runner which watches files and provides instant feedback from their tests. It would be useful for students to research testing frameworks as a number of jobs exist linked to User Experience and Testing Frameworks and this is something they may wish to peruse.
Slide 24:	Conclusion
Slide 25-26:	Terminology and References.

12.5 Laboratory Sessions

The laboratory time allocation for this topic is 2 hours.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

This is an individual exercise, but students should be given ten minutes at the beginning of the class to prepare the names of three websites for evaluation. The websites should not be 'big name' websites, because these will have had accessibility experts assist in their design. Instead, they should be smaller websites with which students are familiar. For example, Yahoo would be inappropriate, but a single author blog would be fine. They should write the URL only onto a small scrap of paper, and each of these scraps should be placed in a hat or a box and then mixed up. Students should then pick two scraps each out of the box or hat. Students are then responsible for evaluating the site from the three perspectives discussed in the lecture.

Exercise 1:

Making use of the websites with which you have been provided, (Amazon.com and uglytub.com), evaluate each according to the following criteria:

1. Technical correctness
2. Standards compliance
3. User satisfaction (you are the user)
4. Accessibility
5. User Experience and Testing Frameworks

You do not need to write a formal report of your testing as part of this exercise, but you should include the results and your conclusions in your research journal.

Suggested Answer:

The nature of the exercise means that a suggested answer is not possible to give. However, an overview is provided below of the kind of approach which students should take towards arriving at their conclusions.

Technical Correctness

For this part of the evaluation, students will obviously have no access to the underlying server or the code, but they can assess how robust the system is by trying to break it. If the web page uses the GET method for parameters, have them manually insert the data via the navigation bar. Have them make sure each link works the way it should. Have them insert invalid data whenever they are asked for it. Have them try to access pages that are not explicitly linked. Students should also try to identify what kind of technical architecture is used. However, students should not attempt to do any damage to the site, and this would include SQL injection attacks if they suspect an underlying database.

Standards Compliance

To assess standards compliance, students should make use of the freely available validators and run these over the web pages. If they can get access to a raw XML feed as a result of using a GET URL, they should assess the provided XML to see if it is well formed or valid.

User Satisfaction

Students should assess the website in all the categories indicated in the lecture, including how easy it is to navigate, how instantly striking it is, how well it communicates its intention and how users interact with the system. Students should record their thoughts on workflow, clicks to completion, and general site design.

Accessibility

To assess the accessibility of the site, students should make use of the impairment simulators shown in the lecture (and any others they can locate) to identify usability issues. An assessment of the underlying code too would allow students to comment on things, such as alt tags for images.

12.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Research Journal

As part of your ongoing journal exercise, you should research the following topics:

- Accessibility legislation
- Browser incompatibilities
- The Web Accessibility Initiative
- Usability testing
- User Experience
- User Journey
- Customer Profiling
- Testing Frameworks

Exercise 2: Checklist

Making use of what you have researched for this topic, prepare a ten-twelve element check list of things people should (or should not) do when creating web applications. Present this check-list as a web application where further details justifying you inclusions are available when the user moves their mouse over the list entry. Be prepared to demonstrate this application and discuss your choices during the tutorial.

Suggested Answer:

Highly dependent on student research – no suggested answer.

Exercise 3: Revision

Review the lecture material and ensure that you are comfortable with everything discussed thus far.

12.7 Tutorial Notes

The time allowance for tutorials in this topic is 1 hour.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

This tutorial should be treated as a demonstration opportunity – students should load up their web checklists on a computer, and then students should be permitted to move around the room examining the guidelines others have put forward. You should do this also, making sure to ask questions of each student.

You should also allow time during the tutorial to check that students are working on their assignments and answer any general questions on the expected scope of the work. You may also wish to remind them of the submission deadline and documentation requirements.

Exercise 1: Checklist Showcase

As a result of your private study you will have a web application that presents a check-list of 'dos and don'ts' for web developers. Start this application up and be prepared to answer questions on it.