SCHOLAR Study Guide

Advanced Higher Computing Science
Web design and development

Authored by:

Charlie Love (CompEdNet)

Andy McSwan (Knox Academy)

Heriot-Watt University

Edinburgh EH14 4AS, United Kingdom.

First published 2020 by Heriot-Watt University.

This edition published in 2020 by Heriot-Watt University SCHOLAR.

Distributed by the SCHOLAR Forum.

SCHOLAR Study Guide Advanced Higher Computing Science: Web design and development

Advanced Higher Computing Science Course Code: C816 77

Print Production and Fulfilment in UK by Print Trail www.printtrail.com

Acknowledgements

Thanks are due to the members of Heriot-Watt University's SCHOLAR team who planned and created these materials, and to the many colleagues who reviewed the content.

We would like to acknowledge the assistance of the education authorities, colleges, teachers and students who contributed to the SCHOLAR programme and who evaluated these materials.

Grateful acknowledgement is made for permission to use the following material in the SCHOLAR programme:

The Scottish Qualifications Authority for permission to use Past Papers assessments.

The Scottish Government for financial support.

The content of this Study Guide is aligned to the Scottish Qualifications Authority (SQA) curriculum.

Contents

Topic 1

Analysis

Contents

Prerequisites

From your studies at Higher you should already know how to:

- Identify the end-user requirements of a program, database or website as relates to the design and implementation at Higher level.
- Identify the functional requirements of a program, database or website as relates to the design and implementation at Higher level.

Learning objective

By the end of this topic you should be able to:

- Identify the purpose of a software solution that relates to the design and implementation at this level
- Describe, exemplify, and implement research for:
  - Feasibility studies:
    - Economic
    - Time
    - Legal
    - Technical
  - User surveys
- Describe, exemplify, and implement planning in terms of:
  - Scheduling
  - Resources
  - Gantt charts
- Produce requirement specifications for end-users and develop:
  - End-user requirements
  - Scope, boundaries and constraints
  - Functional requirements in terms of
    - inputs
    - processes
    - processes
- Describe, exemplify, and implement Unified Modelling Language (UML):
  - Use case diagrams:
    - Actors
    - Use cases
    - Relationships

You may have already covered the material in this topic as it applies to Software, Database and Web design and development.

1.1 Introduction

Analysis is the process of understanding what it is that the client, users and stakeholders want from the system that is being developed. There needs to be a detailed list of the requirements for the product. This will form the basis of a contract between the developer and the client and will be legally binding.

We will create the following at the analysis stage:

- A Requirements Specification
- End User Requirements
- The Scope and Boundaries of the project
- A list of Constraints on the project
- The Functional Requirements for the Project

1.1.1 Requirement Specifications

The requirements specification details the scope, boundaries and constraints of the intended software, details the basis of payment for the work to be completed and sets out how the software will be designed, tested, documented and evaluated before hand over to the customer. It may also detail any longer term maintenance agreement between the customer and the developer.

1.1.2 End user requirements

In the Higher Computing Science course we looked at how the end user requirements would be generated by using **Personas**, **User Stories**, **User Scenarios**, **Use Case**, User Devices and Software.

At Advanced Higher level can use these techniques to help generate a list of what the users would expect from the software being created.

1.1.3 Scope and boundaries

The Scope of the project is held in the **product backlog** - it is the definition of the features that the product must have - also known as the requirements of the software. If a feature is no longer required then it is dropped from the product backlog because it is out of scope.

Constraints are limitations that affect the development of the product. A project may be constrained by time (so only a certain number of development sprints can be completed), or may be limited by budget or may be limited by legal position regarding the data that it processes.

It is very important at the outset to establish clearly the scope, boundaries and constraints of the project. Scope and boundaries are opposite sides of the same "coin". Between them, they give a precise description of the extent of the project.

Here is a typical statement about scope and boundaries. You will find similar statements by

searching the web. "The project scope states what will and will not be included as part of the project. Scope provides a common understanding of the project for all stakeholders by defining the project's overall boundaries."

One way of thinking about it is:

- the scope clarifies what the project must cover;
- the boundaries clarify what the project will not cover.

For example, suppose your project was to develop an expert system giving students guidance on job opportunities which they should consider after graduating from University.

The scope of the project would be to create an expert system. Then it would be necessary to describe the range of jobs and degrees that would be included in the system, the level of information that would be output by the system (does it suggest contact addresses as well as simply job types), the types of questions that the user will be asked. Does it cover all degrees, or is it only for students with Computing Science degrees, and so on . . . All these things will define the boundaries of the system.

Sometimes it is also helpful to spell out exactly what will NOT be covered. So, for example, a clear statement could be made which states that the system will NOT cover advice on jobs for those with medical and veterinary degrees, or jobs overseas.

The scope and boundaries could also refer to technical issues. For example, they might state that the resultant system will run on any computer capable of running any version of Windows after Windows 7, but not on any other operating system.

Defining scopes and boundaries

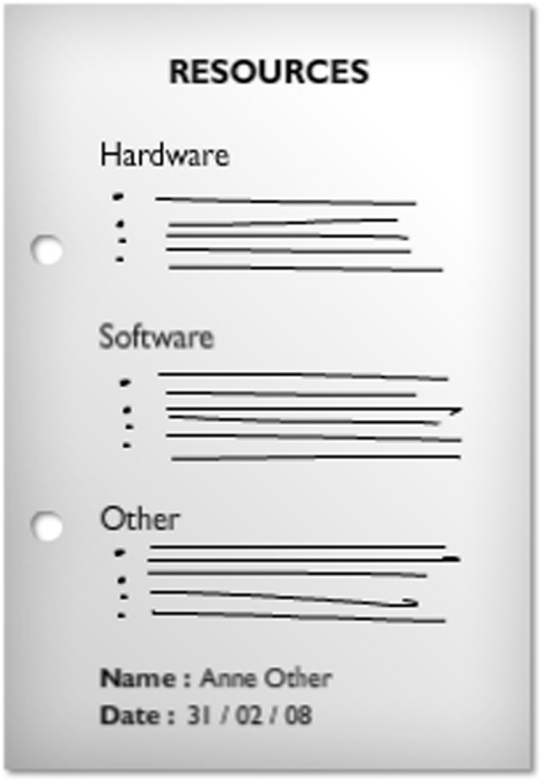
Now, starting from your project proposal, create clear statement of scope and boundaries for your project. Hint: write this as a bulleted list, rather than as a paragraph. This has two benefits - it helps you to clarify your ideas, and it gives you a clear list to use at the end of your project when you are evaluating what you have produced.

Discuss this with your tutor.

Add the scope and boundaries list to your Record of Work.

Don't forget to put your name/initials and the date on the page.

Your Record of Work should now include:



Scope

Examples of scopes for a modular program and a relational database

Scope of a programming task

This development involves creating a modular program.
The deliverables include:

- a detailed design of the program structure
- a test plan with a completed test data table
- a working program
- the results of testing
- an evaluation report

Scope of a relational database task

This development involves creating a relational database.
The deliverables include:

- a detailed design of the database structure
- a test plan with a completed test data table

- a working database
- the results of testing
- an evaluation report

Scope of a website task

This development involves creating a multi-level website.
The deliverables include:

- detailed wireframe designs of each page of the website
- a test plan with a completed test data table
- a working website
- a working website
- an evaluation report

1.1.4 Constraints

Constraints are limitations that affect the development of a product.
A project may be constrained by:

- Time (so only a certain number of development sprints can be completed)
- The scope of the project i.e. what the end result of the project will be and what should the developer deliver (project deliverables) to the client?
- Limited by budget (Cost)
- Legal position regarding the data that it processes i.e. What country will the stored data be held in and does it comply with GDPR

At this stage we should detail what operating systems the project will work on (Windows, iOS, Android, Linux, etc. . .), what environment will be used for development

1.1.5 Functional Requirements

Functional requirements are defined as what the product should do and are split into three sections — Input, Process and Output.

Consider this example of a forum website which allows users to create a user account.

Functional requirements

The new user page should present the user with a form enabling them to create a new account by entering username and email address. On submitting the form, the user receives a welcome email containing a link to confirm the account.

This link should take them to their account details where they can perform the following tasks:

- Inputs
  - enter /change personal information;
  - upload an avatar image;
  - change forum preferences;
  - change password;
  - delete account;
  - confirm their account details by pressing a submit button.
- Process
  - The welcome email will warn the user that their account will expire if they do not go to the link and confirm their details within a certain time period.
  - The site should reject usernames or email addresses which belong to existing users.
  - The site should reject obscene or racist usernames.
  - The site should be secure.
  - If the user details are not submitted within 48 hours then user is deleted from the user database.
  - The page contains a form with text entry boxes with maximum and minimum size restrictions.
  - Client side validation is used for email address and date of birth.
  - The username text box entry is a required one, with a minimum size of 8 characters and a maximum of 16.
  - The Email text box is also required and uses client side validation to check for a valid email address.
  - The page allows an avatar upload restricted to 1Mb. The default avatar is allocated if no upload occurs.
  - Forum options are check boxes.
  - On submit, the form data is passed to a PHP script which sanitizes the data then the username and email address are checked against existing entries in the user database. Usernames are also checked against the database table containing unacceptable values and an appropriate rejection advice response is sent if a match is found in either case. If the username and email are unique, a new database entry is created for that user and a welcome email is sent including a link to the user configuration page using a further PHP script.
  - On submit, the form data is passed to a PHP script where it is sanitized and the user database is updated.
- Outputs
  - Confirmation of successful
    - account creation
    - change of password
    - deletion of account
  - Database reports will be available to administrators.

1.2 Research: feasibility study, user surveys

Learning objective

By the end of this section you will be able to:

- explain the different types of feasibility;
- explain the use of users surveys to gather information about a project.

Research is a vital part of the initial analysis of what the project sponsor is requesting.

1.2.1 Feasibility study

A key part of the project initialisation is undertaking research to ensure that the project is feasible. Software development projects, depending on their size, can be in development for many years and can have budgets worth millions of pounds. It would be foolish to proceed with any project without assessing if completing it is possible.

The basic purpose of a feasibility study is to work out if the proposed expenditure of time and money is likely to be worthwhile and whether the objectives of the project can be achieved: what some projects set out to do might be totally unrealistic.

The results of the feasibility study will determine which, if any, of a number of possible solutions can be further developed at the design phase.

One result of undertaking a feasibility study may indicate that only a simple solution is required to solve the problem:

- a software fault may be identified that is easily fixed;
- more staff training might be required if particular software is to be used.

The feasibility study is most often carried out by the **project leader** - an experienced member of staff who is able to understand the needs of the project and the various aspects of feasibility that apply to it.

1.2.2 Types of feasibility

A feasibility study will look at four main areas of feasibility: Technical (is it technically possible to create a solution with the available technology), Economic (is it possible to complete the project with the budget available or is the cost of creating the project justified by the financial reward of doing so), Legal (can the solution be created and adhere to existing laws) and Schedule (is there enough time to complete the project, are the right people and resources available when required to deliver the project on time).

1.2.2.1 Technical feasibility

The feasibility study must ascertain what technologies are necessary for the proposed system to work as it should. It may be the case that suitably advanced technology does not yet exist. Unless it is the object of the project to design a system to use such advanced technology, this would rule the project out as being a non-starter. It would be a foolish move for a feasibility study to evaluate

technologies which are either under development or undergoing testing.

Given that suitable technology does exist, the study must establish if the organisation already has the necessary resources. If not, the study must make clear what new resources the organisation would have to acquire. This will also involve determining whether the hardware and software recommended will operate effectively under the proposed workload and in the proposed environmental conditions. The development of a new system involves risks of one kind or another. Every understanding that might be reached could carry the risk of some misunderstanding:

- software companies and their clients often have different vocabularies and consequently they appear to be in perfect agreement until the finished product is supplied;
- management may have unrealistic expectations of computer systems. The feasibility study is where idealism meets reality.

Further issues might include the training of personnel to use the new system, consideration of service contracts, warranty conditions and the establishing of help desk facilities for inexperienced users.

1.2.2.2 Economic feasibility

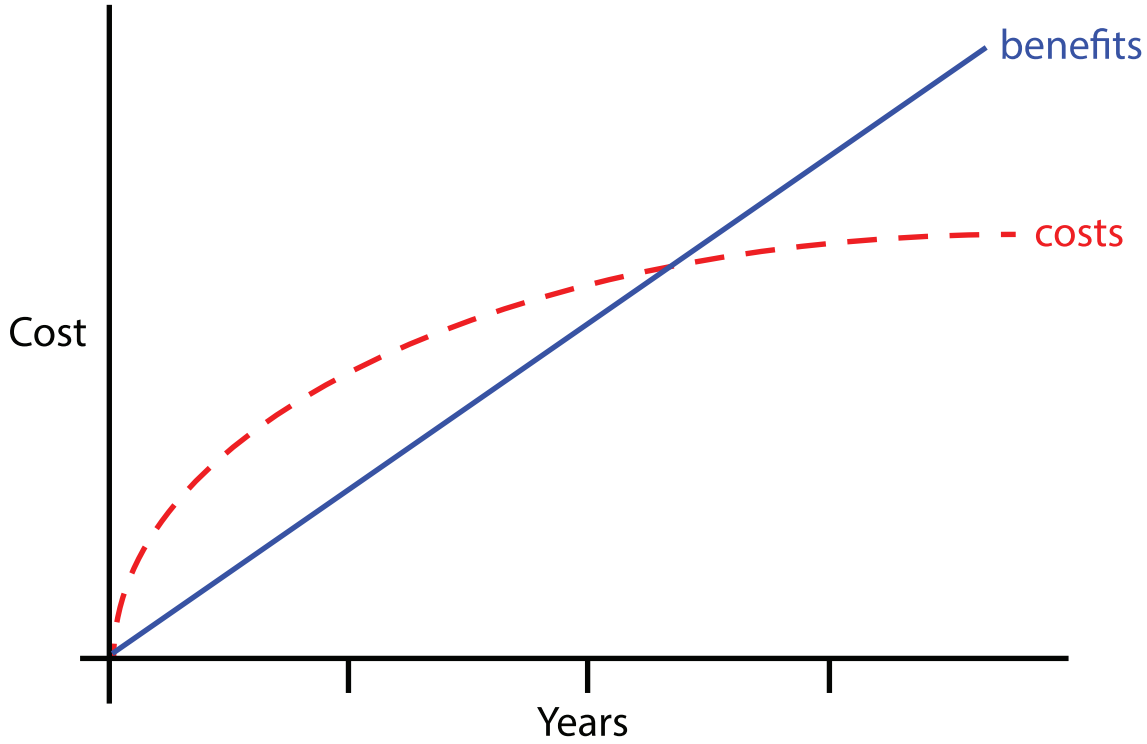
This deals with the cost implications involved. Management will want to know how much each option will cost, what is affordable within the company's budget and what they get for their money. A cost-benefit-analysis is part of the budgetary feasibility study. If the project is not cost-effective then there is no point proceeding.

Setting up a new computer system is an investment and involves capital outlay. The costs of a new system include the costs of acquiring it in the first place (consultancy fees, program development, etc.); the costs of installing it (disruption of current operations, cost of new equipment, alteration of workplace, etc.); and the costs of maintaining it which also includes training.

In the long term management will also want to know the 'break-even point' when the new system stops costing money and starts to make money. This is extremely difficult to quantify. However an accurate estimate of a system's operational life span is a valid option and will rely solely on the knowledge and experience of the systems analyst involved.

Figure 1.1 depicts such a case where the break-even point is at the intersection of the graphs:

Figure 1.1: Break-even point



Tangible benefits that management would certainly be looking for in the new system would be:

- reduced running costs;
- increased operational speed;
- increased throughput of work;
- better reporting facilities.

Note that not all the costs and benefits lend themselves to direct measurement. For example, new systems generally affect the morale of the staff involved, for good or ill. This can only be resolved by competent personnel management practices.

1.2.2.3 Legal feasibility

This has to do with any conflicts that might arise between the proposed system and legal requirements: how would the new system affect contracts and liability, are health and safety issues in place and would the system be legal under such local laws as the UK Data Protection Act? What are the software licensing implications for the new system?

Software licensing can be quite a thorny problem. Licences can be purchased as: client licence (per seat), server licence, network licence or site licence and the period of operation may be annual or perpetual. Software vendors vary in their licensing regulations so this has to be fully investigated.

1.2.2.4 Schedule feasibility

Schedule feasibility may be assessed as part of technical feasibility. Most organisations have an annual schedule of events such as the AGM, end of financial year, main holiday period and so on. Obviously time is a main factor in the development of a new system.

Questions to be asked at this stage might include:

- how long will the proposed system take to develop?
- will it be ready within the specified time-frame?
- when is the best time to install?

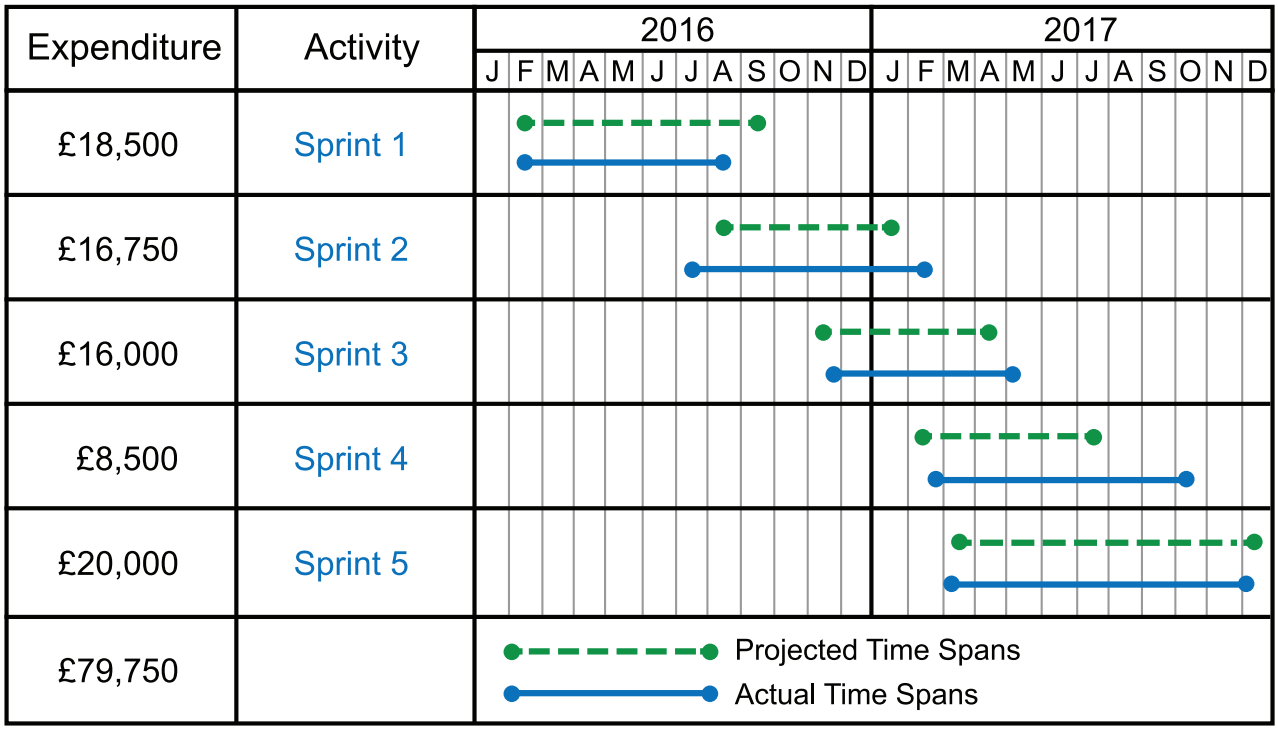
For example, a project might have to start within six months; assuming it would take three months to purchase and install the required hardware and software and a further six months to train the end users. Such a project is not technically feasible because of shortage of time so it would not go ahead unless some of the time constraints were reviewed and changed.

In many cases of project management the scheduling component can be aided by means of a Gantt chart.

A Gantt chart is like a horizontal bar graph used to plan and schedule projects involving several concurrent tasks. The horizontal axis represents the time scale and start and finish times of component parts are graphically represented.

The advantage of a Gantt chart is that it shows, at a glance, the progress of a project as shown in Figure 1.2

Figure 1.2: Example of a Gantt chart



You will look more closely at the creation of a Gantt chart and the concepts of Project management in the Project management Topic of this course.

Activity: Types of feasibility Go online

Q1:

Match the parts together to generate true statements:

1. A project that cannot be delivered in the time available because of the complexity of the task
2. A project which has insufficient finance to hire the necessary staff to complete the project when required
3. A project that processes banking transactions to indicate spending habits in contravention of data protection laws
4. A project which cannot proceed because the software to link two data sources into the project is not available

a) has an issue with technical feasibility.
b) has an issue with legal feasibility.
c) has an economic feasibility issue.
d) has an issue with schedule feasibility.

1.2.3 User surveys

Often with the development of software, the experience of existing users is vital in shaping what the new software should do better than the old software. The users of a particular program are the best people to identify the existing problems with the software and how it could be improved.

Equally, if the software being developed is entirely new, the target user group should, generally, understand the process that the programme will carry out. For example, a new library system, which is to replace a manual card based system, would be developed following some time spent capturing the experience of existing librarians and how they operate.

When the opinion of a number of users is required, it is often easier to create a survey or questionnaire which will capture the information that the project team require.

1.3 Planning

Project management is the process of planning and controlling the activities of a project to ensure that it is delivered on time, with the required features and within the budget available to the required quality.

The Association Of Project Management define project management as:

"Project management is the application of processes, methods, knowledge, skills and experience to achieve the project objectives."

Projects are different from the day-to-day business of an organisation - a project is a specific piece of work which is defined by what it attempts to achieve. A project is usually judged to be successful if it achieves the **objectives** (normally according to some **acceptance criteria**) within an agreed

timescale and **budget**.

A project plan is more than just a list of tasks to be completed. It is a plan that details individual responsibilities, resources available to complete work, the order of work and the key **milestones**.

1.3.1 Relationship between time, cost, scope and quality

Learning objective

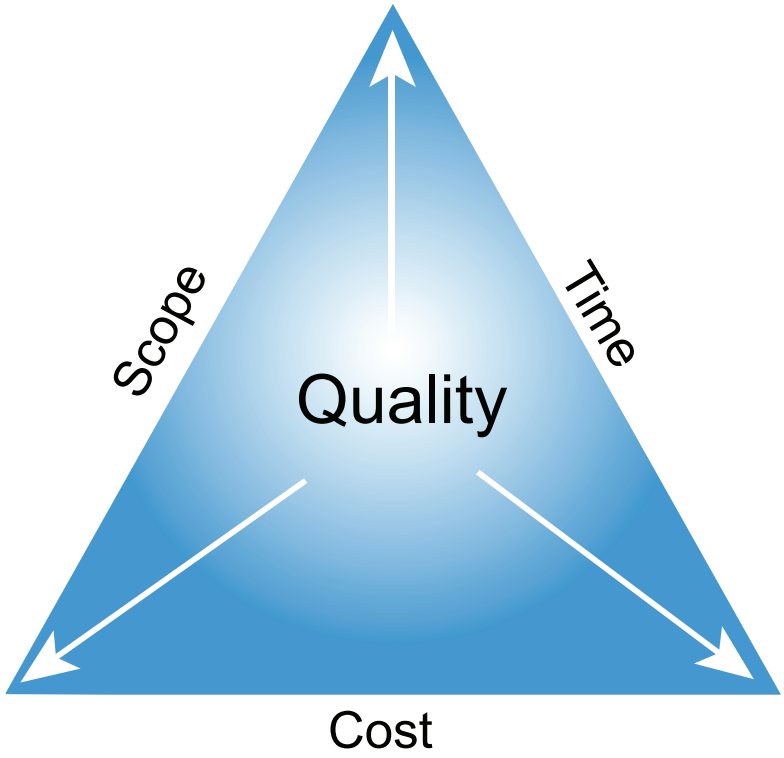
By the end of this section you will be able to:

- describe the relationship between time, cost, scope and quality in the project "triangle".

Before looking more closely at the process of creating a simple project plan, it is important to understand the relationship between **time**, **cost**, **scope** and **quality** when planning a project.

These are the **constraints** of the project. Changes to any one of these constraints has an immediate impact on the others.

Figure 1.3: The Project management triangle



The **Scope** of a project details what needs to be produced and delivered. In the case of a software development project, the system requirements will give us this information. With the **waterfall model** of software development, the requirements of the project are completely known after the analysis phase. Any subsequent changes to the scope will add **time** and/or **cost** to the project because more time and/or more resources (people who have to be paid or over time hours for existing staff) will be required to deliver the project to the same required level of **quality**.

Time is the amount of time it takes to complete the project, to the required **scope** with the available **budget** (cost) to the required quality.

Cost is the budget for the project. If the budget changes then less or more can done. An increase in budget may increase the quality of the work to be done, allow more to be achieved (increased scope) or for the work to be completed in less time.

In a project that uses an **agile Scrum** approach, the time and cost constraints are generally fixed, because the agreement between the developer and the client is for a specific fee for work completed. The reality of large and medium scale projects is that it is almost impossible to understand all of the requirements for the software at the start of the project - using an agile approach reduces the risk to the client as they can change the scope of the project by removing requirements which are no longer needed, adding new ones and by changing the priority of the existing requirements as the project proceeds. This evolving list of requirements is held in the product backlog and is updated prior to each **sprint** in the agile process.

Typically, **Scrum** projects do not have a formal **Project manager** as the team is self-organising and supported by the ScrumMaster, however, in large scale Scrum projects, where there are multiple teams working to produce elements of the project, the role of project manager to act as a buffer between outside organisations and the teams can be useful. This project manager may construct a high-level project management plan based on the **product backlog** and maintain this to estimate completion of the project.

1.4 Unified Modelling Language (UML)

Learning objective

By the end of this section you should be able to use the following design notation associated with programming paradigms:

- Unified Modelling Language (object-oriented programming).

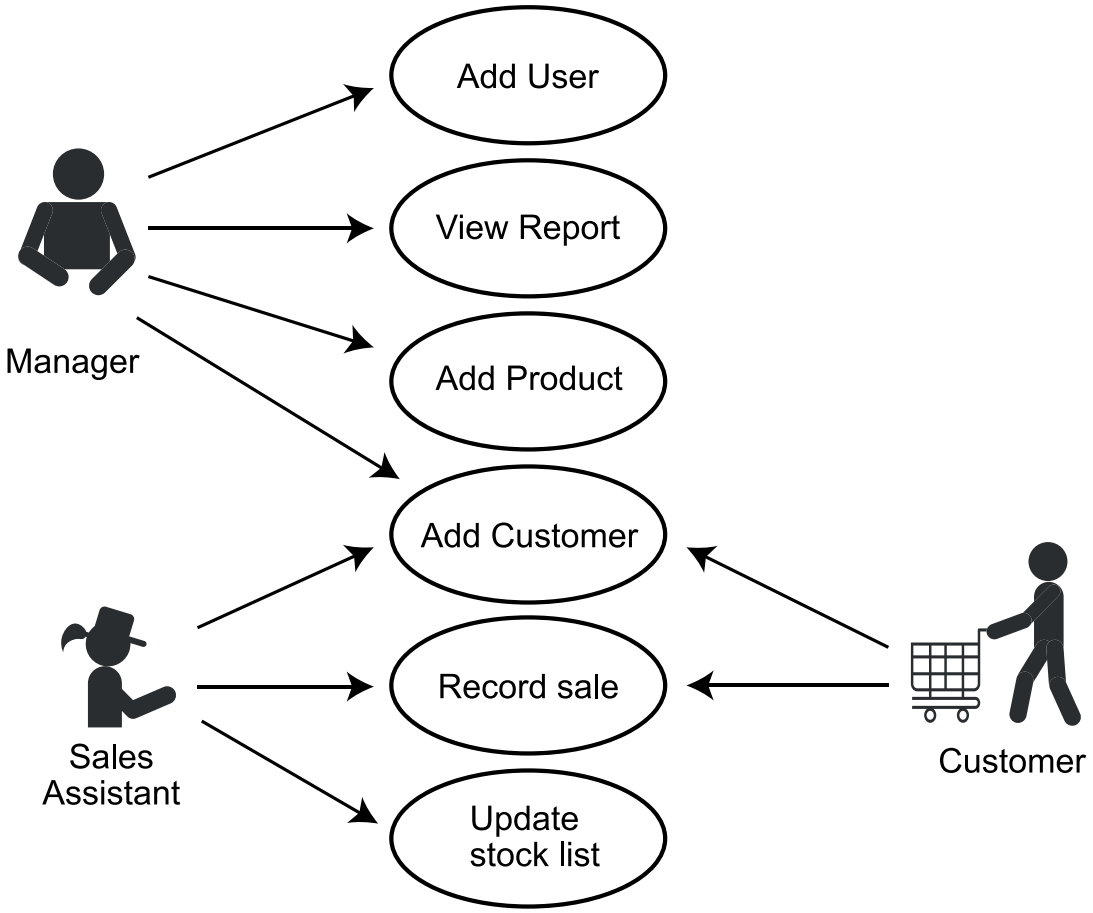
UML is a design notation created specifically to help programmers design systems using the object oriented programming paradigm. As its name suggests it is a formal system for creating models for software design. UML provides a specification for a number of different types of diagram which can be used to represent a software system. Just like pseudocode, UML is programming language independent - it does not dictate which language will be used to translate its diagrams into source code. UML tells you what model elements and diagrams are available and the rules associated with them. It does not tell you what diagrams to create.

There are a number of different types of diagrams specified by UML however for the analysis topic we will only look at **Use Case diagrams**. UML Class Diagrams will be looked at in the design stage of Software Development.

1.4.1 Use Case diagrams

Use case diagrams contain **Actors** (the people or entities who interact with the system) and **Use Cases** which are the procedures which they interact with.

The **Use Cases** in the diagram are identified by the circled text and the **Actor's** interactions with them are identified by the arrows.



A large system will have several different actors. This example might be part of a sales recording system in a shop.

The first stage in creating a Use Case diagram is identifying the actors, by classifying the people who interact with the system, in this case the managers, sales assistants and customers.

The next stage is to identify the data which the actors will interact with and how they change it. The scenario above assumes that the system stores the following data:

- A stock table which stores details of items in stock and their quantities.
- A customer table which stores customer details including previous purchases.
- A staff table which stores staff details and their access privileges to the system.

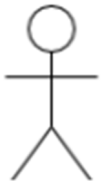
The Customer is involved when a sale is made. If they are a new customer then their details are added to the customer table. When they make a purchase the sale is added to their details.

The Sales assistant also interacts with the customer table but also needs to update the stock list when a sale is made.

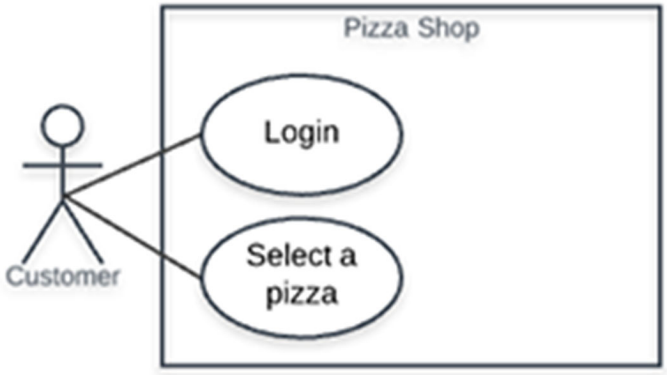
The manager needs to be able to update the staff table if a new sales assistant joins the company, add new products to the stock table and generate reports of sales.

Use case diagrams contain Actors (the people or entities who interact with the system), Use Cases which are the procedures/actions which the user will interact with and the relationships that exist between actors and use cases.

Actors:



Actors should be general categories of users for a system, i.e. customer, staff or another system that it may need to interact with (i.e. a database that the website needs to connect to). We depict an actor in a Use Case Diagram by using a stick figure and putting the type of actor underneath. Each actor must have at least one use case associated with it.



There are two types of actors in use case diagrams, primary and secondary. The primary actors are the ones who start the system. These actors are placed on the left side of the system boundary.

The secondary actors are those users who react to something happening in the system and are displayed on the right side of the system boundary.

Use Cases:

The Use Cases in the diagram are identified by an ellipse with text in the middle explain a task the system is to do. Each individual action that the user can perform in the system will get it's own use case.

Relationships:

- Association
- Include
- Extend
- Generalisation of an Actor
- Generalisation of a Use Case

Association— Solid line shows a basic communication between the actors and the system

Include:

- This is a use case that is not directly accessed by the actors, it shows a dependency between

a **Base Use Case** and a **Dependent Use Case**.

- It is indicated by a dashed line with an arrow that points to the **Dependent Use Case** with the word include.
- The relationship should also be identified in the diagram using double angle brackets and the word <<**include**>>.

Extend:

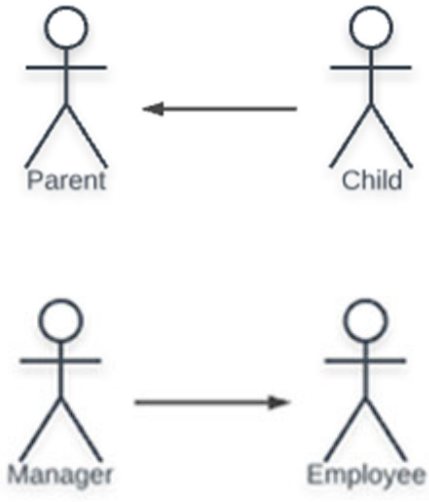
- Similar to an include the **Extend Use Case** will not be directly accessed by an actor and will have a **Base Use Case** but with this relationship the **Extend Use Case** may not always be actioned, certain criteria needs to be met before it can be ran.
- This is also indicated by a dashed line and an arrowhead but this time it will point to the **Base Use Case**.
- The relationship should also be identified in the diagram using double angle brackets and the word <<**extend**>>.

Generalisation (inheritance)

There are two types of **Generalisation** we need to look at, generalisation of actors and generalisation of use cases.

Generalisation of Actors

For actors generalisation uses the idea of inheritance to allow new actor (child) who need to make use of the same basic use cases as the parent actor and can then be given access to any additional use cases they need.

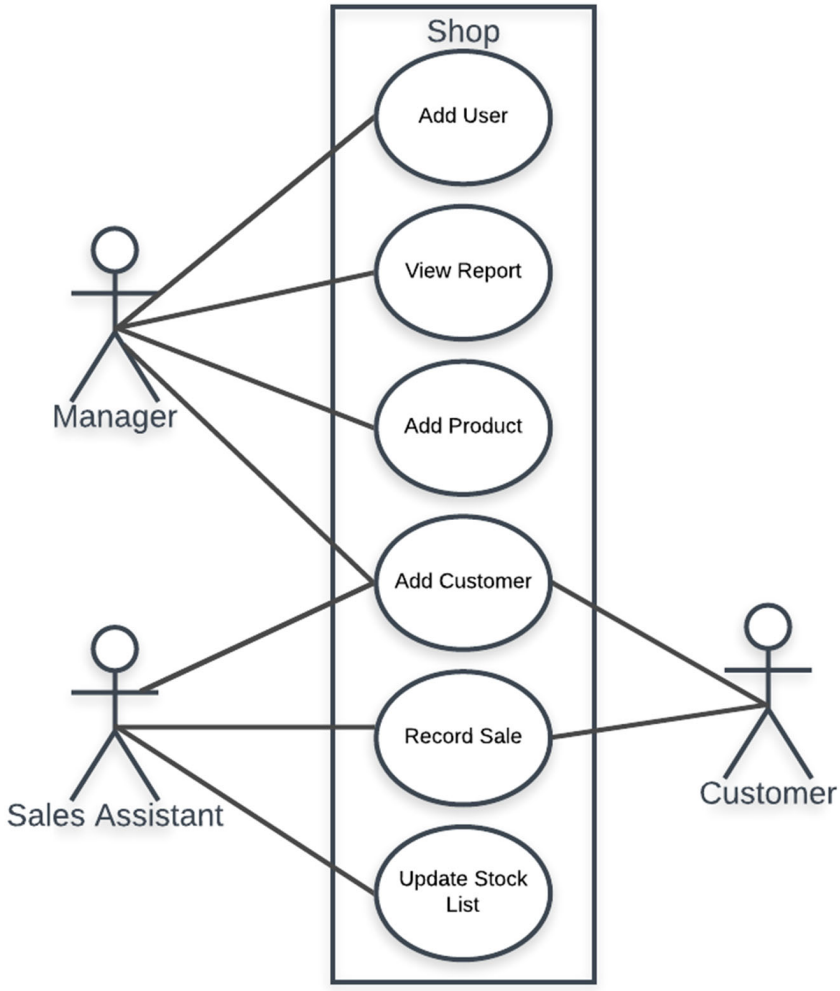


Generalisation of Use Cases

This is Indicated by a solid line and an arrowhead pointing at the **General Use Case**.

Diagrams (Basic example)

The Actor's interactions (associations) with them are identified by solid lines. This is a type of **relationship**.



A large system will have several different actors. This example might be part of a sales recording system in a shop.

The first stage in creating a Use Case diagram is identifying the actors, by classifying the people who interact with the system into categories, in this case the managers, sales assistants and customers.

The next stage is to identify the data which the actors will interact with and how they change it. The scenario above assumes that the system stores the following data:

- A stock table which stores details of items in stock and their quantities.
- A customer table which stores customer details including previous purchases.
- A staff table which stores staff details and their access privileges to the system.

The Customer is involved when a sale is made. If they are a new customer then their details are added to the customer table. When they make a purchase the sale is added to their details.

The Sales assistant also interacts with the customer table but also needs to update the stock list when a sale is made.

The manager needs to be able to update the staff table if a new sales assistant joins the company, add new products to the stock table and generate reports of sales.

Review of Use Case Diagrams Go online

Q2:

Fill in the gaps with the word from the word list.

Word list: Generalisation, Actors, system, ellipse, actioned, Actor, Extend, Use Case, boundary, Include

_____ diagrams are made using _____ that will interact with the system. The system is indicated by a _____ _____ where all the use cases will be placed. A Use Case in the diagram will be represented by an _____.

There are 5 relationships we need to use in a Use Case Diagram:

Association

_____ — this is a use case that is always _____ but not directly by the Actor.

_____ — This Use Case is not directly accessed by the Actor and doesn't always run.

Generalisation of an _____

_____ of a Use Case

1.5 End of topic test

End of topic test: Analysis Go online

Try the end of topic test: Analysis.

Q3: What is an actor in a use case diagram?

a) A person or entity that interacts with the system.
b) An object derived from a class.
c) A relationship between a person and a system.
d) A method.

. . . . . .

Q4: A use case diagram illustrates:

a) interactions between objects.
b) how an object is derived from a class.
c) how actors interact with the system.
d) relationships between classes.

. . . . . .

Q5: There are four types of feasibility: legal, schedule,. . . .

a) sequential and iterative
b) social and economic
c) economic and technical
d) economic and logistical

. . . . . .

Q6: Which four constraints are applied to any project?

- Management
- Sequence
- Milestones
- Rentals
- Product backlog
- Time
- Developer resource
- Scope
- Legal
- Compiler time
- Cost

. . . . . .

Q7: A requirements specification will consist of:

- End User Requirements
- System Specifications
- Use Case Diagram
- Scope
- Boundaries
- Constraints

Topic 2

Design

Contents

Prerequisites

From you previous studies at Higher you should be able to:

- Describe and develop the structure of a multi-level website with a home page and two additional levels.
- Create a website taking into account end-user requirements and device type, an effective user-interface design (visual layout and readability) using wire-framing.
- Create a prototype (low fidelity) web page from a wireframe design.
- Design, describe and apply the key aspects of user-centred design including using analysis tools to capture user requirements.
- Develop user profiles, user personas, user stories, user scenarios and use cases.
- Describe, explain and apply the development of user interfaces using prototypes (from low to high fidelity including wireframes).
- Create a wireframe design that can:
  - Show a horizontal navigational bar.
  - Plan the horizontal and vertical positioning of the media.
  - Indicate form inputs.
  - Indicate file formats of the media in a web page (text, graphics, video, and audio).

Learning objective

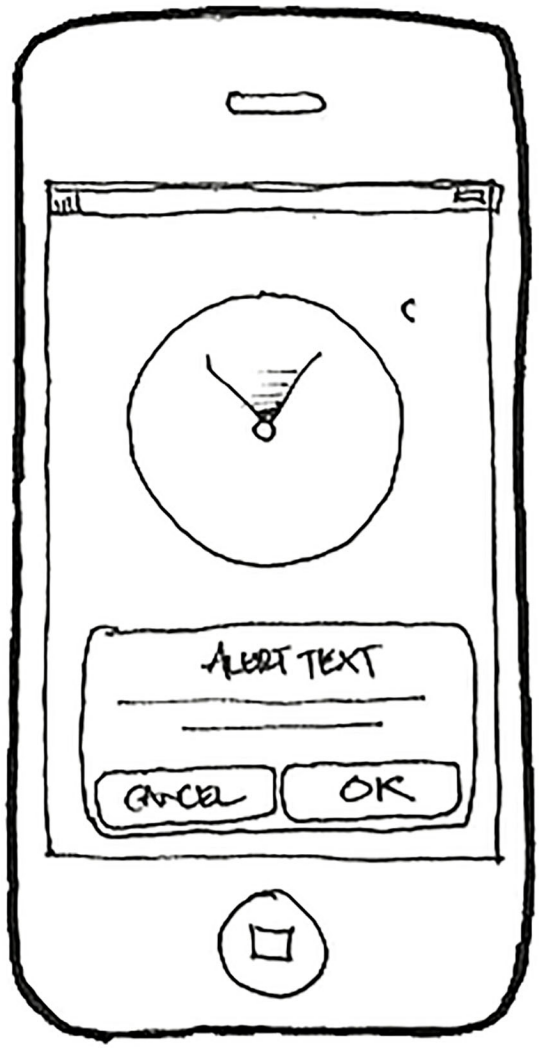
By the end of this topic you should be able to:

- describe, exemplify, and implement wireframe designs showing:
  - visual layout;
  - navigation;
  - consistency;
  - underlying processes;
- use a wireframe design to create a low-fidelity prototype;
- read and understand designs of server-side processes at this level, using the following design techniques:
  - structure diagrams;
  - pseudocode;
- exemplify and implement the design of server-side processes using pseudocode.

2.1 Wireframe designs

Revision

From your previous work at Higher you should be aware that a wireframe design can start out as a simple pencil sketch of a user interface. The wireframe can be developed from a low level of detail and functionality up to a high level of detail and functionality.



A simple wireframe sketch

High quality version of the wireframe

Your wireframe designs from previous levels would show the:

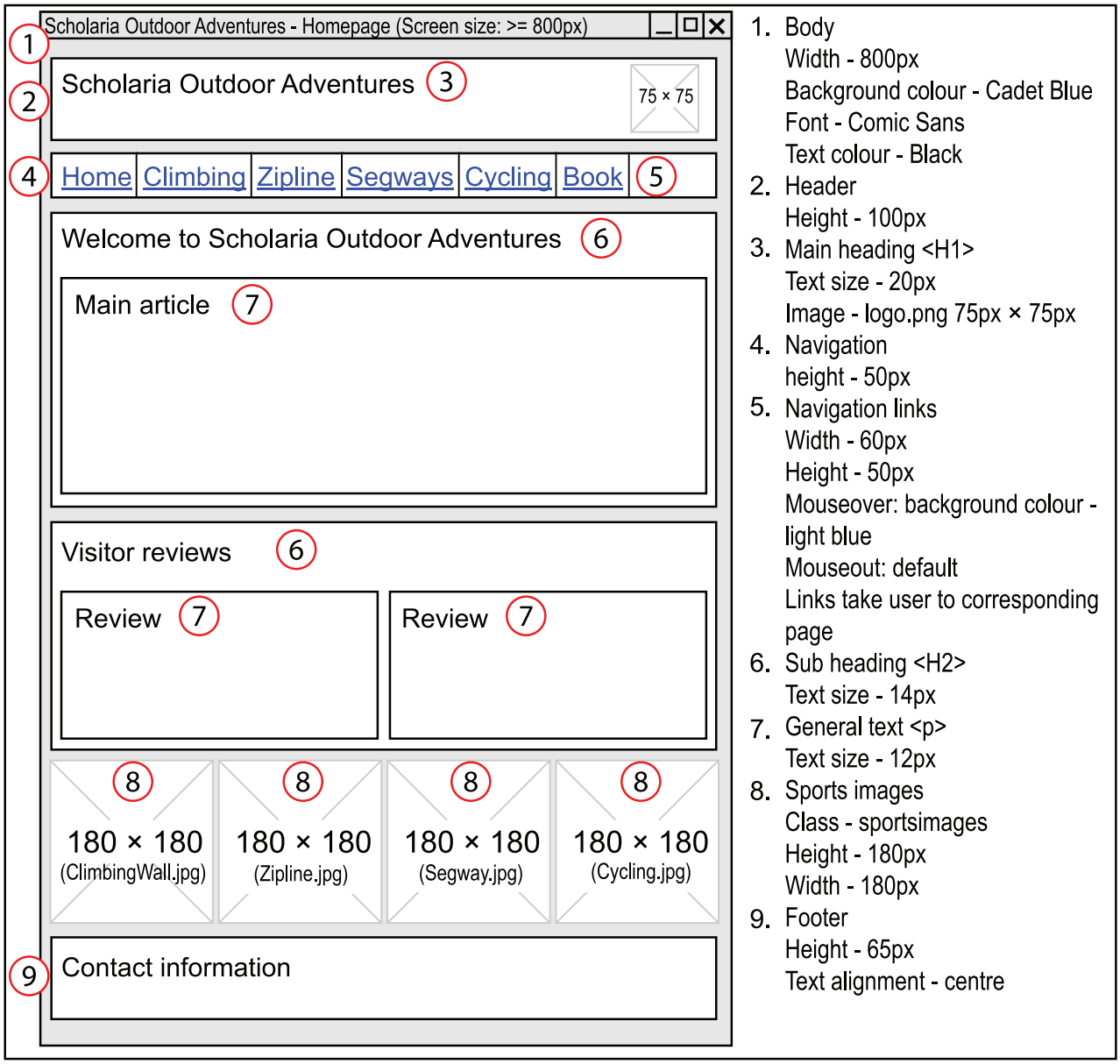
- Positioning (horizontal and vertical) of media on a page (text, graphics, audio and video clips).
- Elements that the user can interact with.
- Horizontal navigation bars and hyperlinks in a page.
- Location of form and types of inputs.
- File formats of media on the page.

2.1.1 Detailed wireframe designs

We now need to add extra detail to the wireframe.

- Visual layout
    - Our wireframe designs need to have detailed and clear information to help the web developers match the design. We should include information such as the dimensions for each page element being used in each page and any media to be displayed.
- Navigation
    - In a simple one page wireframe design where there are links to pages, we can show these links in the annotations.
    - We can also show navigation in wireframes by showing the flow that the user would see when they click on a link or tap on a button and what new pages would open for them.
- Consistency
    - Pages from one website should have a consistent user interface / style to help the user navigate around the site. Our wireframe designs should demonstrate this in as much clarity and detail as possible.
- Underlying processes
    - Elements in a page can have processes applied to them either from CSS rules or JavaScript events being triggered.

Consider this wireframe example.



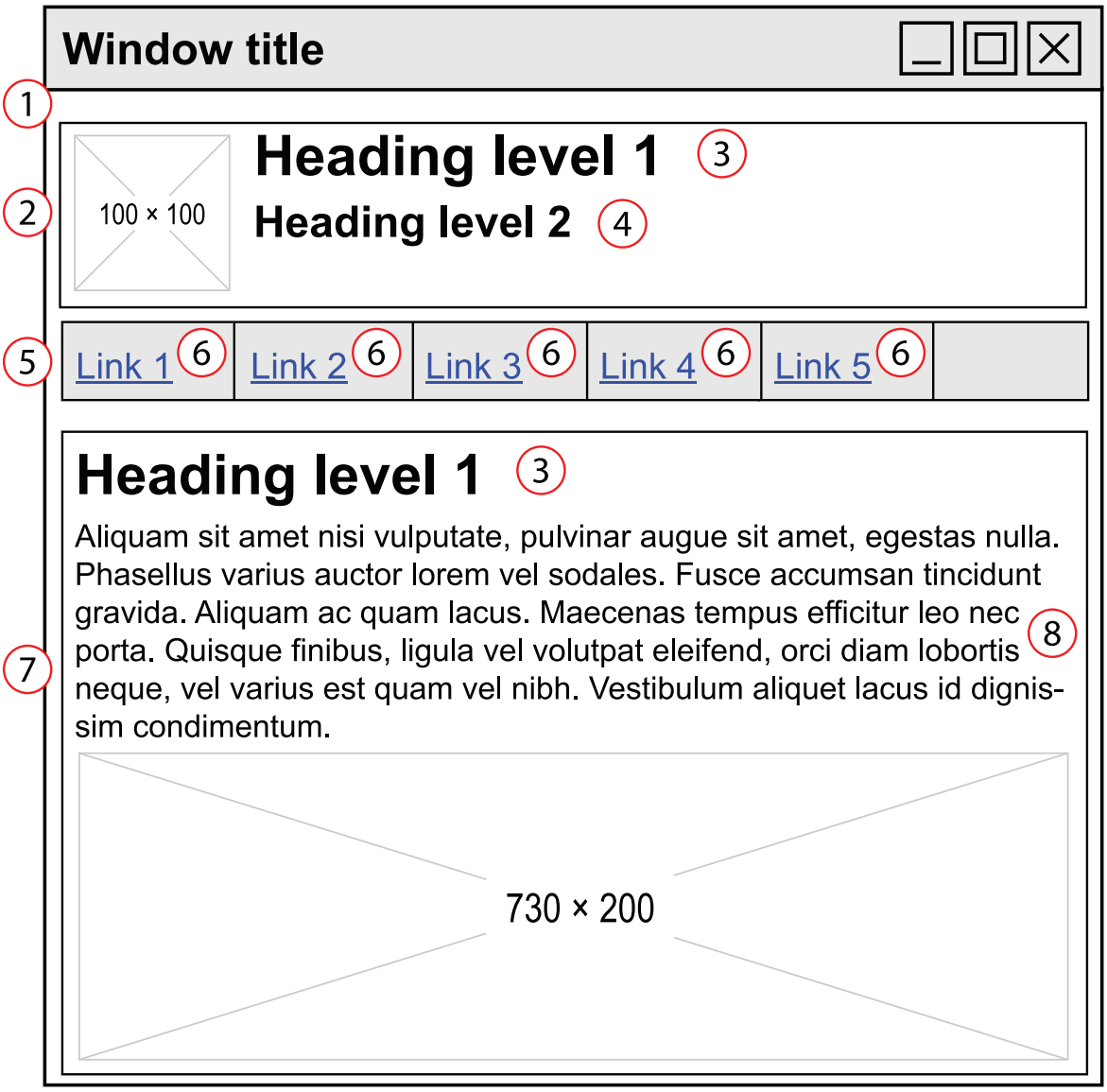
Each element to be placed in the page has been annotated with a number and a list of these annotations with their instructions placed to the right hand side of the design.

The list clearly indicates the dimensions, details of background and text colour and any processes being applied. There is some indication of navigation links and the pages they link to.

Consistency can be shown when multiple elements have the same number assigned (e.g. all sub headings are number six) meaning a consistent style is being applied.

Wireframe design Go online

We now want to add extra detail to this wireframe design.



1. Body
2. Header
3. Main heading
4. Sub heading
5. Navigation
6. Navigation link
7. Content
8. Text

Complete the wireframe design for the page by adding a list of all the annotations and their details. Be sure to include all the requirements at this level.

High fidelity prototype

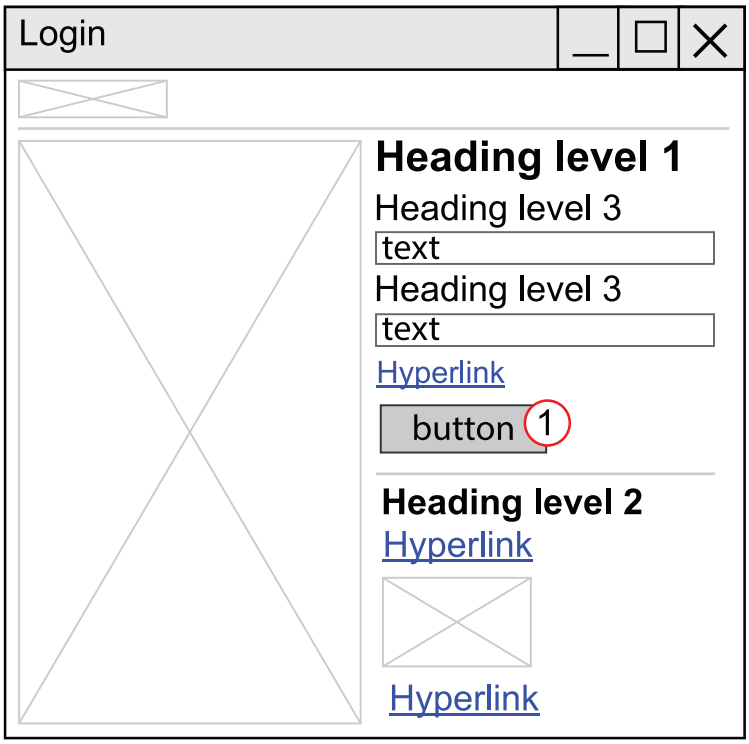
Using your chosen HTML development environment, take the wireframe design you created in the previous activity and then create a high fidelity prototype of the page that matches the design.

Please go online to download an example answer.

2.1.2 Underlying processes

We've looked at how to annotate a wireframe design, however we also need to create designs showing what needs to happen when certain actions are performed by the user. An example of this could be what happens when you log in to SCHOLAR.

A basic wireframe for the SCHOLAR login page would look like this. For this example we are only detailing the attributes for the ***Login*** button.



1. Background color - blue
   Text colour - white
   Text alignment - centred
   Processes: use action attribute of form to locate where login data to be sent
   Data: (Username and Password) will be sent using POST method

Underlying processes such as PHP scripts could be annotated on a wireframe (e.g attached to a search button as in the previous image) and then further designed using pseudocode.

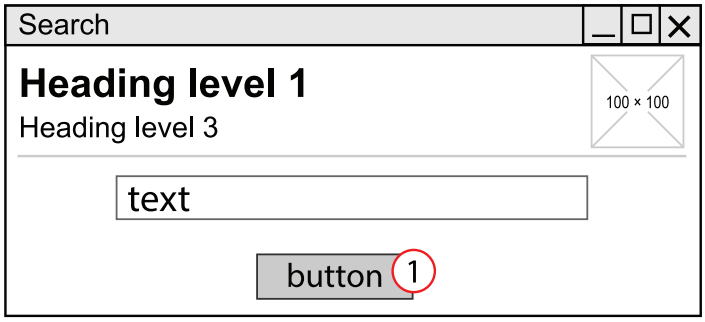
Key point

Any processes at this point don't need to be fully designed, this can be completed later using a more structured design notation like pseudocode or structure diagrams where it is easier to show more detail.

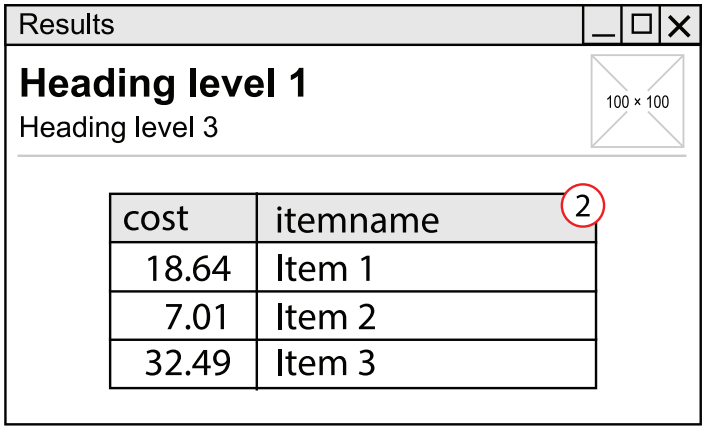
A better option would be to incorporate the submit button into a PHP/SQL search and show a formatted results (table) being returned on your wireframe.

Example : Search results

A simple search page has been designed along with a results page that loads once the user has entered their search terms.



1. When the search button is pressed the text entered by the user is passed to the results page using the GET method.



2. A table element is created in HTML and an SQL query is performed to find the data to populate the table from the database using the text entered by the user on the search page received via the GET method.

2.2 Prototypes

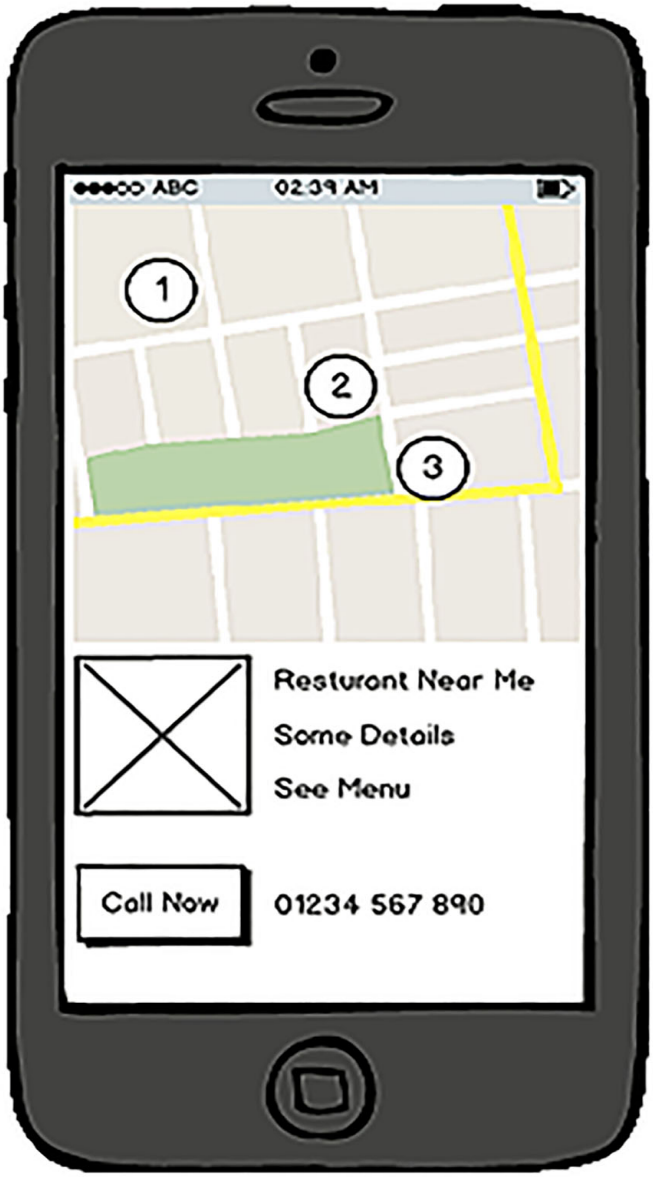
After you have decided on the final design for your page with your wireframe we will then need to create a prototype for the page. A low fidelity prototype has a lot in common with a wireframe design however it is at this point that we start to add more detail to the design.

To do this we may add real details such as the text we want to appear in the page, images we'd want to use and some basic colour schemes.

Low fidelity prototype

Low fidelity prototypes may be paper-based or computer based with little or no interaction. They range from a series of hand drawn mock-ups to printouts to simple models of the user interface with limited interaction.

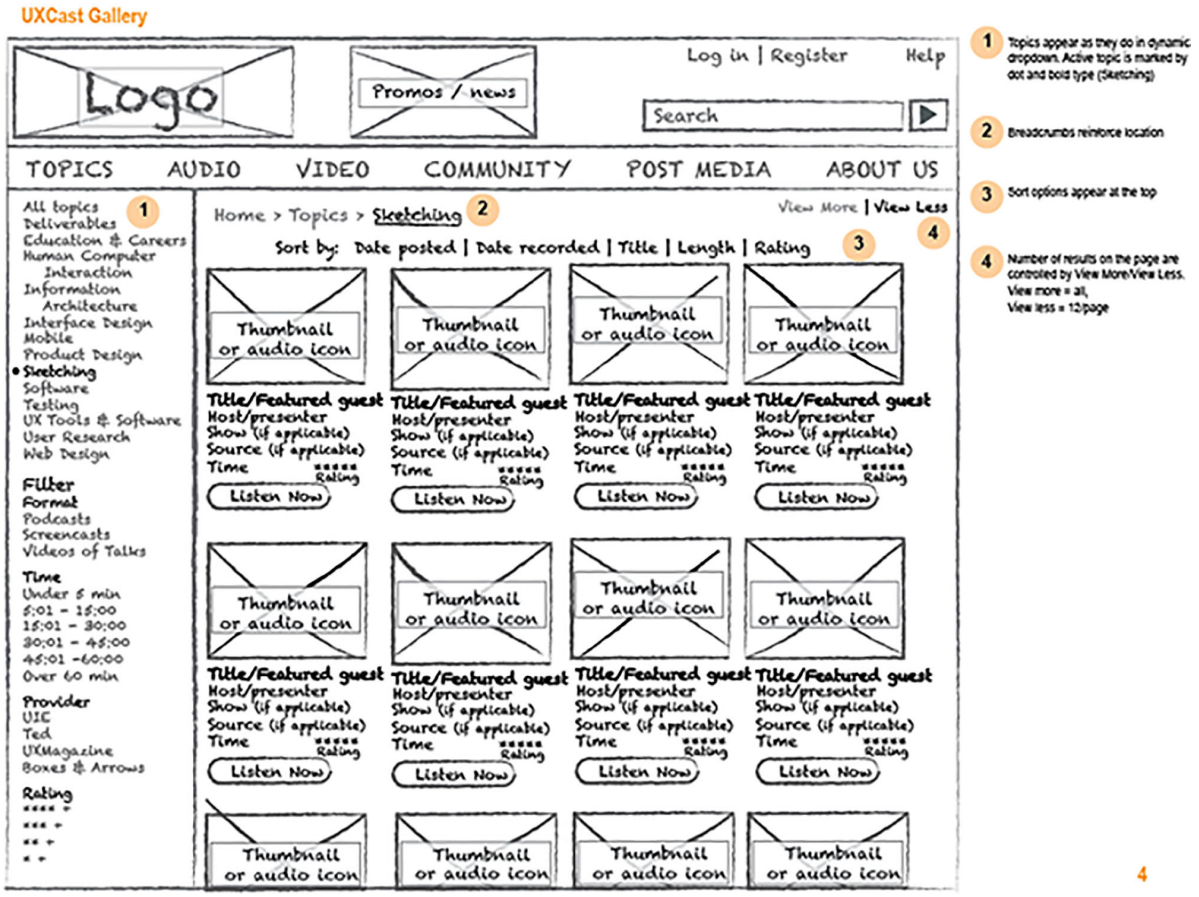
In theory, low fidelity sketches are quick to create. Low fidelity prototypes are helpful in enabling early development of alternative designs, which helps provoke innovation and improvement. An additional advantage to this approach is that when using rough sketches or very simple models, users may feel more comfortable suggesting changes.



In the following examples you will see a similar structure to the wireframes in *Detailed wireframe designs* however you will notice there is now much more detail (e.g. the link titles are present as well as the "Listen Now" buttons). You should also notice that again the annotations appear alongside the prototype with the numbers being used to identify what is being demonstrated.

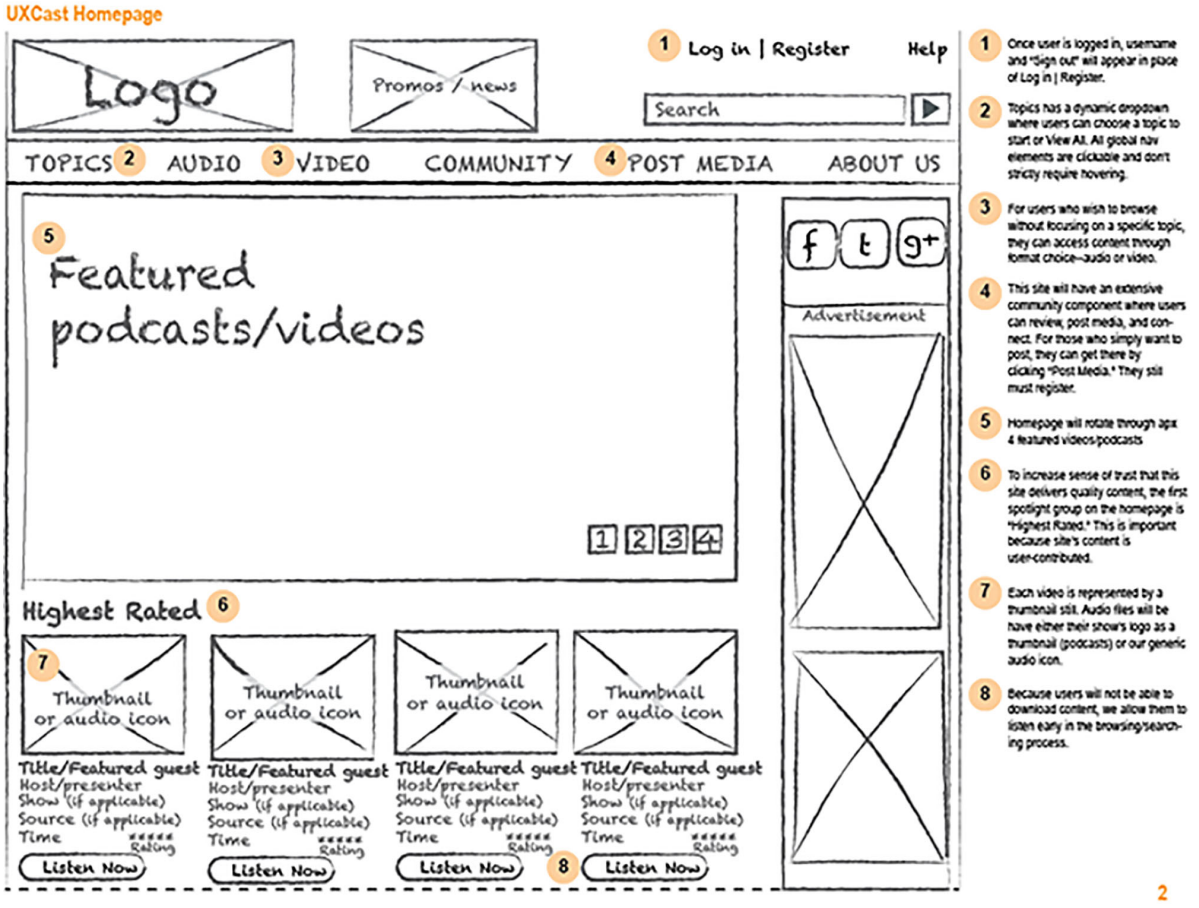
Examples

1. Detailed wireframe 1



. . . . . .

2. Detailed wireframe 2



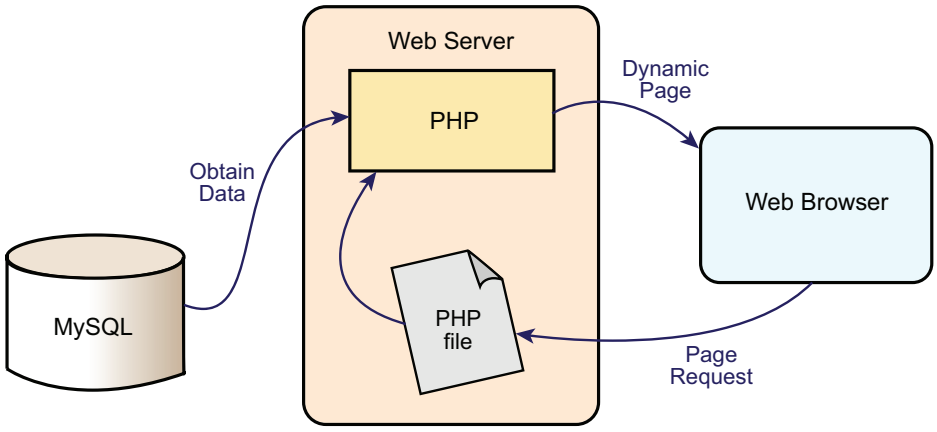
2.3 Server side processes

The web services we use on the internet are based on the **client-server model** - the **client**: your web browser or application requests data from the **server** or **web server**. The server retrieves that data and sends it to the client.

The client has some power to process and display the data it receives and it can send data with the requests it makes to the server. The server can process these requests and the data it receives.

Don't think of a server as just a very big computer. A server can refer to either the hardware / computer that is running server software or the server software itself. A single computer is capable of running several servers at the same time, but may be no larger than one you might have at home.

A typical set up for web development has a web server, one or more server-side scripting languages and a database server for data storage. The database server could be running on the same computer as the web server or could run on a computer in another location.



Later in the course you will need to create our own similar system. We will use PHP as our programming language however, you should be aware that there are a number of other popular server-side scripting languages such as Ruby, Java and Python.

2.4 Design notations

As you would with a program in software development or a query in database design, we need to plan what our system is going to do. To do this we will be using two design notations you should already be familiar with.

- Structure diagrams.
- Pseudocode.

Key point

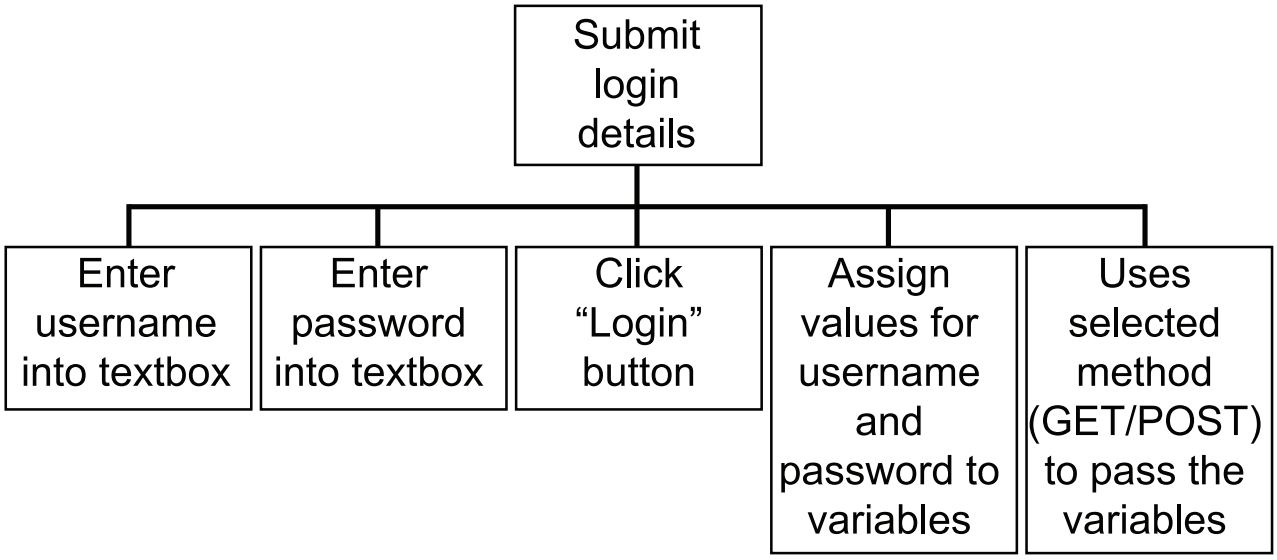
While you will need to be able to read a server-side program designed in both of these notations you will only ever need to create a design in pseudocode at this level.

Both of these notations can be used together, for example the top level design for an application can be in a structure diagram while refinements for a procedure are given in pseudocode.

Consider the steps that are taken when you log in to SCHOLAR.

1. A PHP session is started.
2. The login page is displayed.
3. Submit log in details.
4. The server validates (authenticates) your details.
5. A page is displayed depending on whether the details are valid or not.

Structure diagram



Step 3 (Submit login details)

Pseudocode

5.1 Check the username exists in the user database with an SQL query.
5.2 If the user is found check that the correct password for the user has been entered via an SQL query.
5.3 If the username and passwords match user details from database then set status to "Logged in".
5.4 If the username and passwords don't match user details from database then reload login page with an error message.

Step 5 (A page is displayed depending on whether the details are valid or not)

2.5 Summary

Summary

You should now be able to:

- describe, exemplify, and implement wireframe designs showing:
  - visual layout;
  - navigation;
  - consistency;
  - underlying processes;
- use a wireframe design to create a low-fidelity prototype;
- read and understand designs of server-side processes at this level, using the following design techniques:
  - structure diagrams;
  - pseudocode;
- exemplify and implement the design of server-side processes using pseudocode.

2.6 End of topic test

End of topic test: Design Go online

Q1: A good wireframe design should show detailed information that will be displayed on the final page.

a) True
b) False

. . . . . .

Q2: Designers use which technique to generate an outline design of the user interface of an application/website?

a) Persona.
b) Wireframe.
c) Pseudocode.
d) Use case.

. . . . . .

Q3: A paper prototype is a:

a) version of the application drawn on paper with which users can interact.
b) sketch of the user interface which users can comment on.
c) pseudocode design.
d) template of buttons, menus, and text boxes that users can discuss.

. . . . . .

Q4: A typical web server model consists of:

a) a web server and a web browser making requests.
b) a database server responding to web server requests.
c) storage and a PHP interpreter.
d) a web browser and a MySQL database.

Topic 3

Implementation

Contents

Prerequisites

From your studies at Higher you should be able to:

- describe and exemplify the structure of a web page;
- use semantic elements (HTML5) to structure a basic web page;
- create an internal stylesheet to control the layout of a page;
- create a basic form structure to collect information;
- apply validation to form elements that check:
  - the user has entered data (presence check);
  - data is within a set range (range check);
  - the data entered is within a set maximum / minimum length;
- inline styles, internal stylesheets and external Cascading Style Sheets (CSS);
- understand the display order priority when stying a web page (external CSS, internal stylesheet, inline styles);
- understand the order of priority given to each method of styling a web page;
- grouping selectors to increase efficiency of CSS and web pages;
- descendant selectors to increase efficiency of CSS and web pages;
- using the following properties and rules to control appearance and positioning of elements in a web page:
  - display (block, inline, none);
  - float (left, right);
  - clear (both);
  - margins / padding;
  - sizes (height, width);
- grouping and descendant selectors to create horizontal navigation bars;
- use the following properties and rules when creating a horizontal navigation bar:
  - list-style-type:none;
  - hover;
- read and explain code that makes use of the above CSS;
- use a client side scripting language (JavaScript) to add interactivity to a web page by using the following events:
  - onmouseover;
  - onmouseout;
  - onclick.

Learning objective

By the end of this topic you should be able to:

- **CSS**
  describe, exemplify, and implement responsive pages using the following queries:
  - media type:
    - print;
    - screen;
  - media feature:
    - max-width;
- **HTML**
  describe, exemplify, and implement form elements including:
  - FORM element:
    - action;
    - method (get and post);
  - INPUT, SELECT and TEXTAREA elements:
    - name;
    - value;
  - TABLE element:
    - th;
    - tr;
    - td;
- **PHP**
  describe, exemplify, and implement coding of server-side processing to:
  - assign form data to server-side variables:
    - $_get();
    - $_post();
  - open and close connection to database server:
    - die();
    - mysqli_connect();
    - mysqli_close();
  - execute SQL query:
    - mysqli_query();
  - format query results:
    - echo;
    - mysqli_fetch_array();
    - mysqli_num_row();

  and:

Learning objective continued

- assignment, repetition and selection using server-side local and global variables;
- sessions:
  - session_start();
  - session_destroy();

- read and explain code that uses constructs appropriate to this level.

3.1 Responsive web design

One area of **CSS** that is worth considering is the ability to write rules that will apply to particular types of media: screen, print, screen reader software, etc. This has become an increasingly important area of design as web designers strive to create sites that are not only accessible, but can re-adjust themselves to best suit the device and the **viewport** of the user. This is called **responsive** web design.

In order for our responsive design to work on all devices we need to add the following **meta** tag into our HTML pages <head> section.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Once this is in place we will then use CSS **media queries**. These allow you to create different CSS rules that enable your site to display well on whatever device the user chooses to access it from.

Media queries

Media queries work like a selection (if) statement in software development. If the conditions are true then use the CSS rules inside the media query. The structure of a media query will declare what type of media it is being set for and what condition the web page is looking for.

The syntax for a media query is:

```
@media not | only mediatype and (expressions) {
  ...
}
```

not or **only**

not: can be used to apply a set of styles if a certain condition is False. For example, you could set the background colour to be light blue for all conditions except when the screen size is 300 pixels wide where it is white instead. If you plan to use this operator when designing a site you should keep in mind that this is an HTML5 command and this was not supported in older browsers.

only: prevents older browsers which don't support media queries from applying the style, it has no effect on modern browsers.

mediatype

This specifies where to apply the media query - screen or print.

screen: allows media rules to be set for when the site / page is being viewed on a device.

print: allows you to specify media rules for how your document will look when it is printed on paper. This can be useful for the user to print a basic version of the page and not print anything unnecessary.

In a web page you will have navigation buttons which are displayed on-screen, but you may not want these to appear when printed. Using the *screen* mediatype you can make then visible and in the *print* mediatype you can make them hidden.

expressions

A list of one or more conditions that have to be True before the media query will be used, for example

the maximum width of the display or its orientation.

Example : Smartphone media query

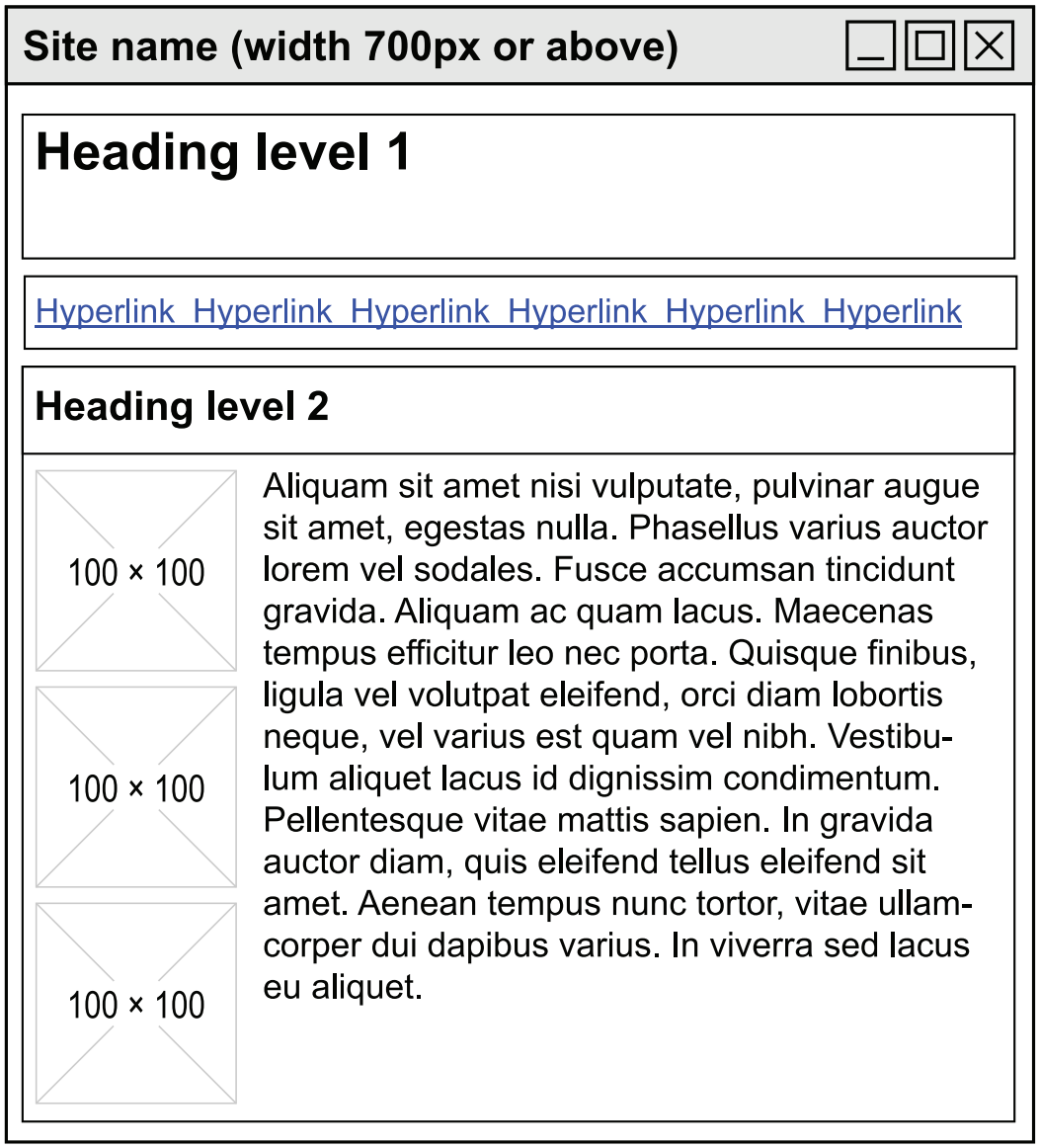
To display a web page on a 480 pixel wide smartphone.

```
1 @media only screen and (max-width: 480px)
2 {
3 ...
4 }
```

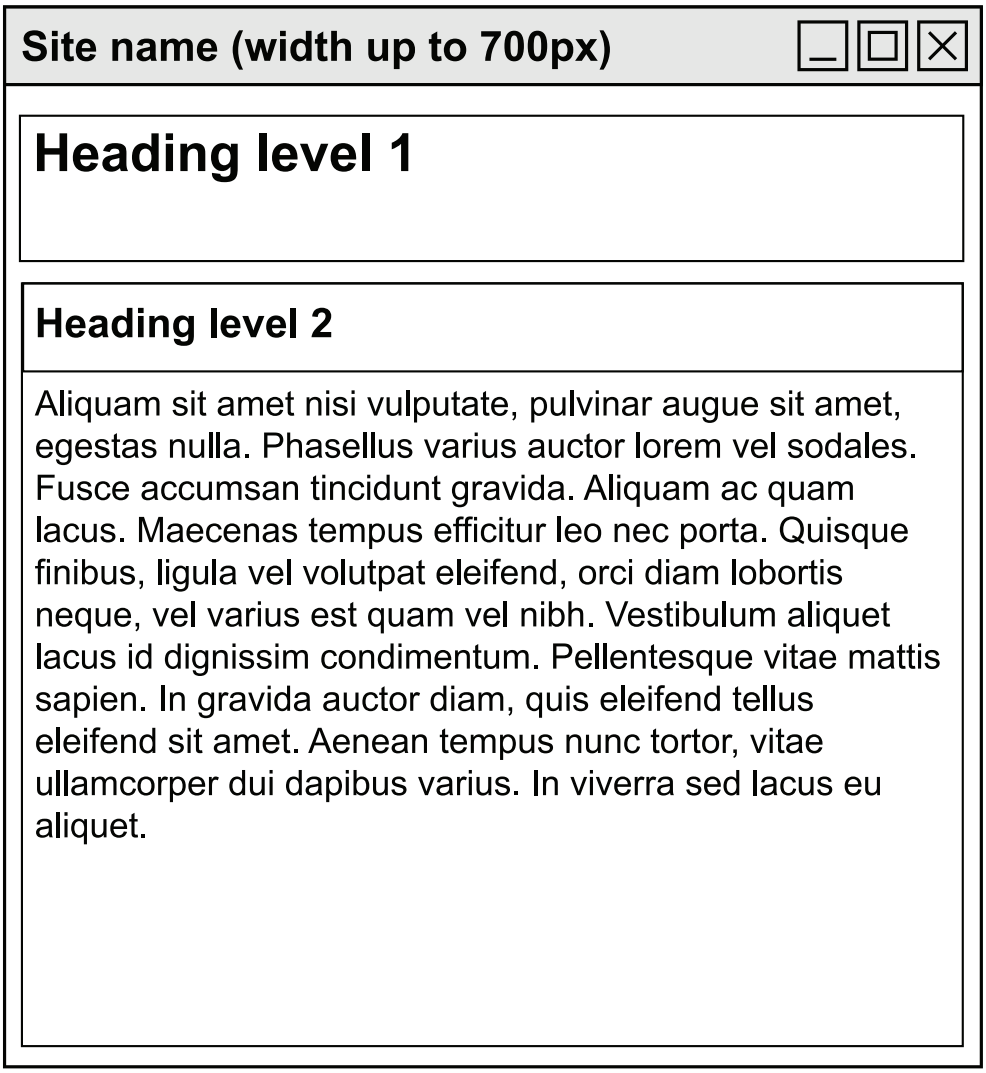
Key point

Your media queries should be placed at the end of your CSS document to ensure that they can properly override the styles already set for the site.

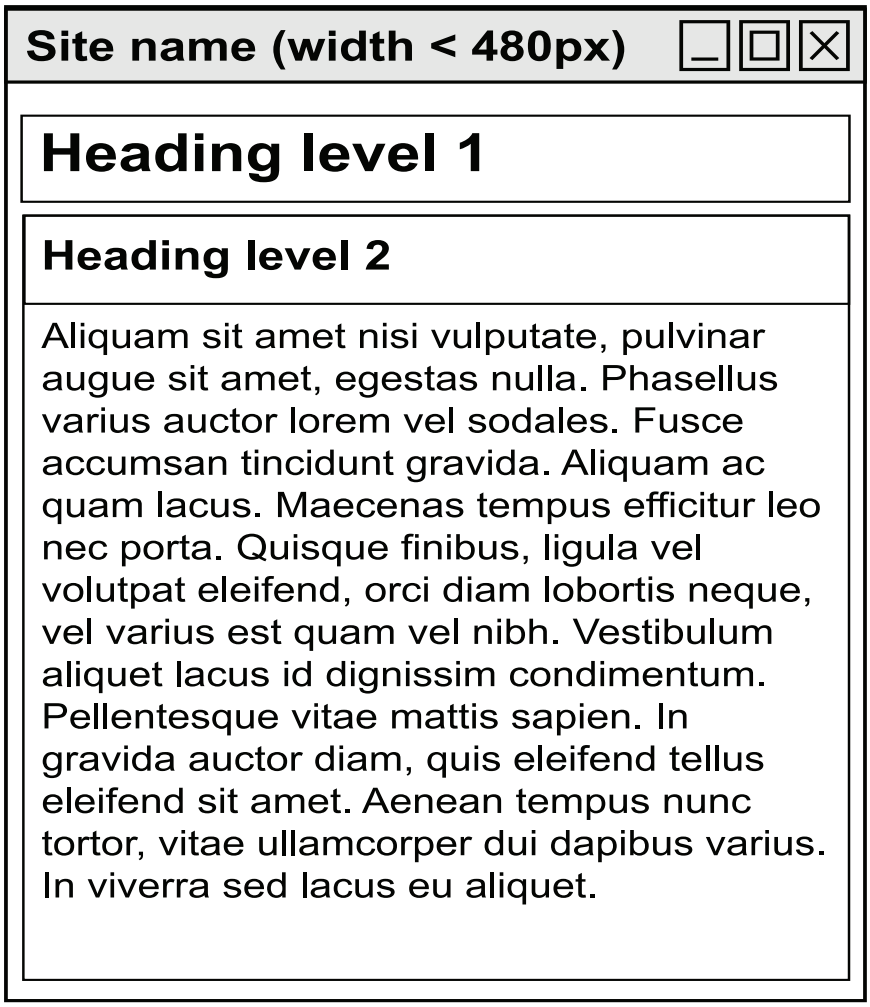
As an example look at the following wireframe designs for the same web page but catering for three different sizes of screens / devices.



Personal computer (desktop or laptop) - 1280 × 720 landscape mode
The background colour will be #e6e6fa



7" Tablet - 600 $\times$ 1024 portrait mode
The background colour will be #c6e0fb



5" Smartphone - 480 × 854 portrait mode
The background colour will be #c6e0fb

To see these media queries being applied download the *Visit Scholaria* website and view it in your web browser. Press the F12 key on your keyboard (in most browsers) to open the developer tools and try resizing the pages of the site to see the changes being made.

Note: this is not a full website, but you'll complete it in the next activity.

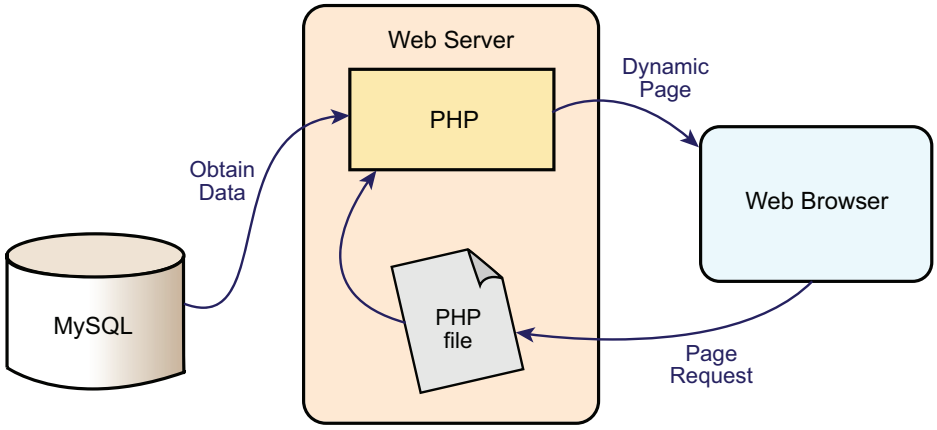
Download the website file(s) here: *Please go online to download the file(s).*

Complete the Visit Scholaria website

- Create the missing pages for this site and apply the stylesheet to these pages.
- Add some:
  - extra rules to the existing media queries such as changing the font families.
  - new rules in order to display the navigation section that has been hidden to work on a smaller screen.

3.2 Client and server-side processes

Server-side processes are carried out as you would expect on the **web server** where the website is stored. This server performs a service for the user.



The web services we use on the internet are based on the **client-server model** - the client (your web browser or application) requests data from the **server**. The server then retrieves that data and sends it to the client.

The client has some power to process and display the data it receives and it can also send data with the request it makes to the server. The server can process these requests and the data it receives.

Don't think of a server as just being a very large computer, it may be the same size as a desktop computer. A server can refer to either the hardware/computer that is running the server software or just the server software itself. A single computer is capable of running several servers at the same time.

A typical set up for web development has:

- A web server.
- One or more server-side scripting languages.
- A database server for data storage.

The database server could be running on the same computer as the web server or could be run on a different computer in another location.

Later in this course you will need to create a web server system and we will use PHP as our programming language. You should be aware that there are a number of other popular server-side scripting languages such as Ruby, Java and Python.

3.3 Working with PHP

You should be familiar with programming and scripting languages (e.g. JavaScript), but now we will look at PHP. There are similarities between it and other languages, some of which should appear familiar to you (e.g. IF statement), but there may be others that are new or are different.

You will have used loops like these in *Software design and development* programs in the past and

so will be aware of how they work, but not the syntax to use them in PHP.

PHP is a server-side scripting language. This concept is not obvious, especially if you're just used to designing pages with **HTML** and **JavaScript**. A server-side scripting language is similar to JavaScript in many ways, as they both allow you to embed little programs (scripts) into the HTML of a web page. In executing, such scripts allow you to control what will actually appear in the browser window more flexibly than is possible using HTML.

The key difference between JavaScript and PHP is that, while the web browser interprets JavaScript once the web page containing the script has been downloaded, server-side scripting languages like PHP are interpreted by the web server before the page is even sent to the browser. Once interpreted, the PHP code is replaced in the web page by the results of the script, so all the browser sees is an HTML file. The script is processed entirely by the server hence why it is called a server-side scripting language.

Activity 1

Create the following code in your HTML editor and save it with the name *today.php* in the root folder of your web server installation. Now run the script by entering the URL (e.g. http://localhost/today.php).

```
1  <html>
2    <head>
3      <title>Today's date</title>
4    </head>
5    <body>
6      <p>Today's date (according to this web server) is
7      <?php
8        echo (date("l, dS F Y h:i:s a."));
9      ?></p>
10   </body>
11 </html>
```

Most of this is plain HTML, but the line between <?php and ?> however is written in PHP. The web server is asked to interpret everything between these two tags and convert it to regular HTML code, before sending the web page to a browser that requested it.

```
1  <? php     // begin PHP code
2  ...
3  ?>     // end PHP code
```

The browser is presented with something like this:

```
1  <html>
2
3  <head>
4      <title>Today's date</title>
5  </head>
6
7  <body>
8      <p>Today's date (according to this web server) is
```

```
 9         Tuesday 11th of August 2015 02:10:29 PM</p>
10 </body>
11
12 </html>
```

All signs of the PHP code have disappeared. In their place, the output of the script has appeared and looks just like standard HTML. This example demonstrates several advantages of server-side scripting:

- No browser compatibility issues. PHP scripts are interpreted by the web server and nothing else, so you don't have to worry about whether the language you're using will be supported by your visitors' browsers.
- Access to server-side resources. In the above example, we place the date according to the web server into the web page. If we had inserted the date using JavaScript, we would only be able to display the date according to the computer on which the web browser was running. Now while this isn't an especially impressive example of exploiting server-side resources, we could have just as easily inserted some other information that would only be available to a script running on the web server, for example, information stored in a MySQL database running on the web server computer.
- Reduced load on the client. JavaScript can significantly slow down the display of a web page on slower computers, as the browser must run the script before it can display the web page. With server-side scripting, this load is passed to the web server.

3.3.1 Syntax and PHP

PHP syntax will be very familiar to anyone with an understanding of C, C++, Java, JavaScript, Perl, or any other C-derived language. A PHP script consists of a series of commands, or "statements", each of which is an instruction that the web server must follow before proceeding to the next. PHP statements, like those in the above-mentioned languages, are always terminated by a semicolon (;).

The following is a typical PHP statement:

```
1 echo ("This is a <strong>test</strong>!");
```

This statement invokes a built-in function called ***echo*** and passes it a string of text: `This is a <strong>test</strong>!`. Built-in functions can be thought of "things that PHP knows how to do without us having to spell out the details". PHP has a lot of built-in functions that let us do everything from sending e-mail to working with information stored in various types of databases. The echo function, however, simply takes the text that it is passed and places it into the HTML code of the page at the current location.

Activity 2

Create the following script:

```
1  <html>
2
3  <head>
4      <title>Simple PHP Example </title>
5  </head>
6
7  <body>
8      <p>
9          <?php echo( "This is a <strong>test</strong>!");?>
10     </p>
11 </body>
12
13 </html>
```

Copy and paste this code into a file called *test.php* and place it the root folder of your web server. View the file by entering the URL http://localhost/test.php and then view the HTML source.

The browser viewing the page sees the following:

```
1  <html>
2
3  <head>
4      <title>Simple PHP Example </title>
5  </head>
6
7  <body>
8      <p>This is a <strong>test</strong>!</p>
9  </body>
10
11 </html>
```

Notice the string of text contained HTML tags (and), which is perfectly acceptable.

You may wonder why we needed to surround the string of text with both parentheses (brackets) and quotes. Quotes are used to mark the beginning and end of strings of text in PHP, so their presence is required.

The parentheses serve a dual purpose. First, they indicate that *echo* is a function that is to be called. Second, they mark the beginning and end of the list of "parameters" that you wish to provide to tell the function what to do. In the case of the echo function, you only need to give the string of text to appear on the page, but we'll be looking at functions that take more than one parameter (for which we'll list the parameters separated by commas), as well as functions that take no parameters at all (for which we will still need the parentheses, but won't type anything between them).

3.3.2 Variables and Operations

Variables in PHP are identical to variables in most other programming languages. As you may already know, a variable is a name given to a location in memory into which a value may be placed.

The following statement creates a variable called `$testvariable` (all variable names in PHP begin with a dollar sign) and assigns it a value of 3:

```
1 $testvariable = 3;
```

PHP is a "loosely typed" language, which means that a single variable may contain any type of data (be it a number, a string of text, or some other kind of value), and may change types over its lifetime. So the following statement, if written after the statement above, assigns a new value to our existing `$testvariable`. In the process, the variable changes from containing a number to containing a string of text:

```
1 $testvariable = "Three";
```

The equals sign used in the last two statements is called the "assignment operator", as it is used to assign values to variables. Other operators may be used to perform various mathematical operations on values:

```
1 $testvariable = 1 + 1;  // Assigns a value of 2.
2 $testvariable = 1 - 1;  // Assigns a value of 0.
3 $testvariable = 2 * 2;  // Assigns a value of 4.
4 $testvariable = 2 / 2;  // Assigns a value of 1.
```

The lines above each end with a comment. Comments are a way to describe what your code is doing by inserting explanatory text into your code and telling the PHP interpreter to ignore it. Comments begin with `//` and end at the end of the same line.

The C language standard comments which use `/* */` at the start and end of the comment also work in PHP. Comments are used by programmers to note what the program code does and are inserted into the program as it is written and amended. All the examples, from this point on, will include comments.

The `+`, `-`, `*` and `/` operators provide the standard arithmetic operations of add, subtract, multiply and divide.

One other commonly used operator is used for string concatenation. The `.` operator is used to join strings of text together. For example:

```
1 /* Assigns a value of "Hi there!". */
2 $testvariable = "Hi " . "there!";
```

Variables may be used anywhere an actual value can be. Look at the following example PHP code:

```
<?php
$var1 = "SCHOLAR";   // Assigns a value of "SCHOLAR" to $var1
$var2 = 5;   // Assigns a value of 5 to $var2
$var3 = $var2 + 1;   // Assigns a value of 6 to $var3
$var2 = $var1;   // Assigns a value of "SCHOLAR" to $var2
                 // the . "<br />" ensures a new line for each output
echo($var1) . "<br />";   // Outputs "SCHOLAR"
echo($var2) . "<br />";   // Outputs "SCHOLAR"
echo($var3) . "<br />";   // Outputs 6
echo($var1 . " rocks!" . "<br />" );   // Outputs "SCHOLAR rocks!"
echo("$var1 rocks!" . "<br />");   // Outputs "SCHOLAR rocks!"
echo('$var1 rocks!' . "<br />");   // Outputs '$var1 rocks!'
?>
```

Look at the last two lines. Note that you can include the name of a variable inside a text string and have the value inserted in its place if you surround the string with double quotes, however, a string surrounded with single quotes will not convert variable names to their values.

Key point

A variable will display its:

- **Value** if surrounded by **double** quotes.
- **Name** if surrounded by **single** quotes.

3.3.3 Function parameters

Consider a function that adds two numbers together (e.g. addNumbers()) and then displays the result. Until now, we have declared the variables within the function along with their values. If we then needed another function to do a similar calculation but with different values, we could copy the function and change the variables.

For a simple function such as this, it is not difficult to change the variables, but it is not an efficient way of coding. When functions get very long or complex, errors are more likely to occur if you change one variable without making a corresponding update to another variable.

A better way is to create a common function into which we pass the values we want the function to use within it. The values (or parameters) are listed within parentheses after the function name.

Function with no parameters

```
function nameOfFunction() {

  // Code to be executed goes here

}
```

Example : A function with no parameters

```
<?php
  $first = 15;
  $second = 20;
  $answer = 0;
  addNumbers();
?>
...
<?php
  function addNumbers() {
  global $first, $second;
  $answer = $first + $second;

    echo "The result of " . $first . " + " . $second . " is " .
        $answer;
  }
?>
```

Function with parameters

```
function nameOfFunction($param1, $param2, ...) {

  // Code to be executed goes here

}
```

Example : A function with parameters

```
<?php
  $num1 = 15;
  $num2 = 20;
  addNumbers($num1, $num2);
?>
...
<?php
  function addNumbers($first, $second) {
    $answer = 0;
    $answer = $first + $second;

    echo "The result of ". $first . " + " . $second . " is " .
        $answer;
  }
?>
```

You should notice that:

- The variables passed to the function have been changed (i.e. `$num1` and `$num2`), but within the function they are still `$first` and `$second` respectively.
- `$answer` has been moved into the function. Each time the function is called it will set the initial answer to be zero, but it cannot be set or changed to another value outside of the function.

Key point

The names given to the parameters as part of the function declaration (e.g. function addNumbers ($first, $second)) will be the names of the variables used within the function (i.e. $first and $second), you don't need to declare them again inside the function.

The first value passed to the function will be assigned to the variable $first, the next to $second, etc.

3.3.4 Arrays

An array is a special kind of variable that contains multiple values. If you think of a variable as a box that contains a value, then an array can be thought of as a box with compartments, where each compartment is able to store an individual value.

The simplest way to create an array in PHP is with the built-in array function:

```
1 $myarray = array('best', 62, 'kelly');
```

This code creates an array called `$myarray` that contains three values: 'best', '62', and 'kelly'. Just like an ordinary variable, each space in an array can contain any type of value. In this case, the first and third spaces contain strings, while the second contains a number.

best	62	kelly
$myarray		

To get at a value stored in an array, you need to know its index. Typically, arrays use numbers, starting with zero, as indices to point to the values they contain.

That is, the first value (or element) of an array has index 0, the second has index 1, the third has index 2, and so on. In general, therefore, the index of the nth element of an array is n-1. Once you know the index of the value you're interested in, you can get that value by placing the index in square brackets following the array variable name:

```
1 echo($myarray[0]);  // Outputs "best"
2 echo($myarray[1]);  // Outputs "62"
3 echo($myarray[2]);  // Outputs "kelly"
```

You can also use the index in square brackets to create new elements, or assign new values to existing array elements:

```
1 $myarray[1] = 'which';  // Assign a new value
2 $myarray[3] = 'cat';  // Create a new element
```

You can add elements to the end of an array by using the assignment operator as usual, except with empty square brackets following the variable name:

```
1 $myarray[] = 'new element';
2 echo($myarray[4]);  // Outputs "new element"
```

3.3.5 Associative Array

Array indices don't always have to be numbers but numbers are the most common choice of index. You can also use strings as indices to create what is called an associative array. This type of array is called associative because it associates values with meaningful indices.

In this example, we associate a date with each of three names:

```
1 $birthdays['Charlie'] = '1969-01-05';
2 $birthdays['Tom'] = '1959-05-16';
3 $birthdays['David'] = '1962-09-09';
```

Now if we want to know Charlie's birthday, we just look it up using the name as the index:

```
1 echo('My birthday is: ' . $birthdays['Charlie']);
```

This type of array is especially important when it comes to user interaction in PHP.

3.3.6 Control statements

All the examples of PHP code that we've seen so far have been either simple, one statement scripts that output a string of text to the web page, or have been a series of statements that were to be executed one after the other in order. If you've ever written programs in any other languages (be they JavaScript, C, or BASIC) you already know that practical programs are rarely so simple.

PHP, just like any other programming language, provides facilities that allow us to affect the flow of control in a script. That is, the language contains special statements that permit you to deviate from the one-after-another execution order that has dominated our examples so far. Such statements are called control structures. Don't get it? Don't worry! A few examples will illustrate perfectly.

An IF statement in PHP has the following formats.

Syntax for a simple IF statement

```
1 if ( condition ){
2
3   // Code to be run if first condition is true
4
5 } else {
6
7   // Code to be run if the first condition is not true
8
9 }
```

Example : Simple IF statement

```
1 if ($age < 18) {
2   echo "You are a child.";
3 } else {
4   echo "You are an adult.";
5 }
```

Key point

If you only want to run the code when the first condition is true (e.g. `$age < 18`), the `else` part of the statement can be removed.

Syntax for a complex IF statement

```
1  if ( condition ){
2
3    // Code to be run if first condition is true
4
5  } elseif ( condition ) {
6
7    // Code to be run if the second condition is true
8
9  } else {
10
11   // Code to be run if both first and second conditions are not true
12
13 }
```

Example : Complex IF statement

```
1 if ($age < 13) {
2   echo "You are a child.";
3 } elseif ($age < 18) {
4   echo "You are a teenager.";
5 } else {
6   echo "You are an adult.";
7 }
```

Using IF statements

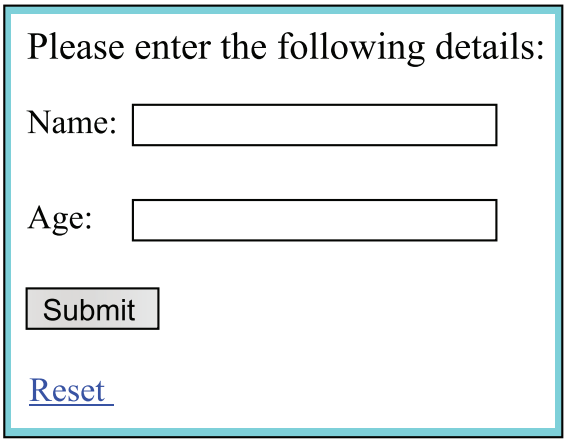
Use the following code to create a site that will ask the user to enter their name and age, and then tell them if they are old enough to have a driver's license.

DriverCheck.php

```
<html>
  <body>
    <form method="get" action="Answer.php">
      <h2>Please enter the following details:</h2>
      Name: <input type=text name="name">
      <br />
      Age: <input type=number name="age">
      <br />
      <input type="submit" value="Submit">
    </form>

    <a href="DriverCheck.php">Reset</a>
  </body>
</html>
```

Expected output



Answer.php

```
<html>
  <head>
    <title>Result</title>
  </head>

  <body>
    <?php
      function ageCheck ($age, $name){
        if ($age >= 17) {
          return "<p>Congratulations $name, you are old enough to
             hold a driver's license</p>";
```

```
11          }else{
12            return "<p>Sorry $name, you are not old enough to hold a
                 driver's license</p>";
13          }
14        }
15        echo (ageCheck($_GET['age'], $_GET['name']));
16      ?>
17    </body>
18  </html>
```

Expected output

```
1  Sorry Carson, you are not old enough to hold a driver's license
```

In *Answer.php* we have created a function called **ageCheck** that will pass in the age and name of the person and then return the result in a new web page to the user. There is also a selection made depending on the age that the user entered on the first page *DriverCheck.php*.

3.3.7 Loops in PHP

There are three loop statements we use when working in PHP:

- While
- Do. . . While
- For

While

The `while` loop will repeat a section of code as long as the condition that is checked at the start of the loop is true. Using of the while loop is useful if you don't know how many times the code will be executed or you don't always want a block of code to be executed unless it's needed.

Syntax of a While loop

```
1  while ( condition ){
2
3    // code to be executed
4
5  }
```

Example : While loop

```
1 $count = 20;
2
3 while ($count > 0) {
4   echo $count;
5   echo "<br>";
6   $count--;
7 }
```

Expected output

```
1 20
2 19
3 18
4 ...
5 3
6 2
7 1
```

Ten green bottles

Create a new page that will use a loop to print the song "10 green bottles" on the page.

Do...While

Similar to the `while` loop this will repeat a block of code until a condition is met, however with this loop the condition is only checked at the end of the code block. This means whatever code is contained in the loop will be executed at least once.

Syntax of a Do..While loop

```
1 do {
2
3   // code to be executed
4
5 } while ( condition );
```

Example : Do..While loop

```
$heating = "on";

do {
  echo "The heating is " . $heating . "<br />";
  if ($heating === "on") {
    $heating = "off";
  } else {
    $heating = "on";
  }
} while ($heating === "off");
```

Expected output

```
The heating is on
The heating is off
```

When the loop begins, the heating is "on" and this is output to the screen. Since the heating is "on", the if statement changes it to "off". As the heating is "off", the while condition is true and the loop begins again. "The heating is off" message is displayed and the if statement sets the heating to be "on". Now the while condition is false and so the loop is not executed a further time.

For

The `for` loop allows a block of code to be executed a specific number of times. An initial (stepper) value is set, the condition to check is then set and an increment command is given.

Syntax of a For loop

```
for( $counter = 0; $counter < 10; $counter++ ){

  // code to be executed

}
```

In this case, the program will loop from the values 0 - 9 stepping up by 1 each time.

Example : For loop

This code uses an array of names and prints each one on a separate line.

```
$names = ["Archer", "Lenny", "Rachel"];

for ($index = 0; $index < count($names); $index++){
```

```
4     echo $names[$index] . "<br />";
5 }
```

Expected output

```
1 Archer
2 Lenny
3 Rachel
```

3.3.8 Server-side variables

Just as in *Software design and development* we needed to know and make appropriate use of **Local** and **Global** variables, we now need to be aware of their use in server-side applications.

Variables in PHP always begin with a dollar sign ($), such as:

```
1 $name = "Lachlan Andrews";
2 $course = "Advanced Higher Computing Science";
3 $school = "Scholaria Academy";
4 $year = 2019;
5 $active = True;
```

Key point

Variable types don't need to be declared in PHP.

Global variable

Any variable we declare outside of a function in our PHP scripts is considered to be a global variable and is therefore accessible throughout our script. In order for them to be accessed within any functions in our script without being passed as a parameter, we need to re-declare them by using the **global** keyword.

Local variable

Local variables in PHP are created inside functions and are discarded when the function terminates.

To see the difference consider the following two scripts.

Using global variables

Script to call the function

```
1 <div id="outFunction">
2   <?php
3     addNumbers();
4   ?>
5 </div>
6
7 <?php
8   echo "the following line is outside the function:";
```

```
9   echo "<br />";
10  echo "The result of ". $first . " + " . $second . " is " . $answer;
11 ?>
```

Script to run the function

```
1  <? php
2    // Note the variables are declared outside of the function
3    $first = 15;
4    $second = 20;
5    $answer = 0;
6
7    function addNumbers () {
8      global $first, $second, $answer;
9      $answer = $first + $second;
10
11     echo "The following line is printed from within the function:";
12     echo "<br />\n"; //This takes a new line
13     echo "The result of " . $first . " + " . $second . " is " . $answer;
14     echo "<br />";
15   }
16 ?>
```

Output

```
1 The following line is printed from within the function:
2 The result of 15 + 20 is 35
3 The following line is printed from outside the function:
4 The result of 15 + 20 is 35
```

Using local variables

Script to call the function

```
1  <div id="inFunction">
2    <?php
3      addNumbers();
4    ?>
5  </div>
6
7  <?php
8    echo "the following line is outside the function:";
9    echo "<br />";
10   echo "The result of ". $first . " + " . $second . " is " . $answer;
11 ?>
```

Script to run the function

```
1  <? php
2    function addNumbers () {
3    // Note the variables are declared within the function
4      $first = 15;
5      $second = 20;
6      $answer = 0;
7
8      $answer = $first + $second;
```

```
 9
10      echo "The following line is printed from within the function:";
11      echo "<br />\n"; //This takes a new line
12      echo "The result of " . $first . " + " . $second . " is " . $answer;
13      echo "<br />";
14    }
15  ?>
```

Output

```
1  The following line is printed from within the function:
2  The result of 15 + 20 is 35
3  The following line is printed from outside the function:
4  Notice: Undefined variable: first in localExample.php on line 10
5  Notice: Undefined variable: second in localExample.php on line 10
6  Notice: Undefined variable: answer in localExample.php on line 10
7  The result of + is
```

You should notice:

- There are three errors shown because the variables we're using in our script are local to the function. The `echo` statements are not included within the function, but after it, so the code doesn't know about the variables and therefore raises an error.
- The line showing the calculation result is echoed to the screen but has no values to display.

Using parameters

In each of these previous examples, no parameters have been passed into the functions. We pass parameters in by listing them in the parentheses at the end of the function name. Taking the addNumbers() function which we have seen earlier, gives us this.

Script to call the function

```
1  <?php
2    $first = 15;
3    $second = 20;
4    $answer = 0;
5
6    echo "The following line is printed from outside the function:";
7    echo "<br />\n"; //This takes a new line
8    echo "The result of " . $num1 . " + " . $num2 . " is " . $answer;
9    echo "<br />";
10
11   addNumbers($first $second);
12
13   echo "The following line is printed from outside the function:";
14   echo "<br />;\n"; //This takes a new line
15   echo "The result of " . $first . " + " . $second . " is " . $answer;
16   echo "<br />";
17 ?>
```

Script to run the function

```
<? php
  function addNumbers ($first, $second) {
    $answer = $first + $second;

    echo "The following line is printed from within the function:";
    echo "<br />\n"; //This takes a new line
    echo "The result of " . $first . " + " . $second . " is " . $answer;
    echo "<br />";
  }
?>
```

Output

```
The following line is printed from outside the function:
The result of 15 + 20 is 0
The following line is printed from within the function:
The result of 15 + 20 is 35
The following line is printed from outside the function:
The result of 15 + 20 is 0
```

You should notice:

- In the calling function, `$answer` is still zero after the `addNumbers` function has been executed.
- As we have not declared an extra parameter to the `addNumbers` function to accept the answer variable (e.g. `addNumbers($first, $second, $answer)`) and `$answer` has not been passed to it, the variable cannot be changed by the function and so keeps its initial value.
- Within the running function, `$answer` is treated as a local variable.
- If the global keyword had been added to `$answer` in the running function (e.g. `global $answer`), rather than passing it as a parameter, its value would have been updated by the function.

3.4 Web and database integration

3.4.1 Revision

Quiz: Revision Go online

Q1: Server side script is used to:

a) process data within the browser.
b) execute commands using HTML.
c) process data at the server side.
d) securely pass values within an HTML file.

. . . . . .

Q2: Server side validation of data is more secure because:

a) nearly all client-side validation can be circumvented.
b) the data is always sent to the server using an encrypted data connection.
c) form data can only be processed by the server.
d) cookies can be used to authenticate users.

. . . . . .

Q3: Client-side validation of form data is useful because:

a) it allows values to be sent to the browser.
b) it prevents users navigating away from a form while it is partially complete.
c) it validates data before it leaves the browser reducing the number of resubmissions for errors.
d) allows bookmarks to be created which contain data values.

. . . . . .

Q4: A database driven web site:

a) presents a static web page to all users.
b) loads some of the web site data from a sequential text file.
c) is build using data statements and JavaScript.
d) pulls site data from a database to create site pages.

. . . . . .

Q5: To process a server site script your web server will require a:

a) database server.
b) HTML parser.
c) server-side language module.
d) cache.

3.4.2 Server side scripting requirements

This section requires that you have access to a web server that makes use of the *AMP stack of technologies.

*AMP is a term used to design a combination of operating system, Apache Web Server, MySQL database server and PHP. The popularity of *AMP across the World Wide Web is driven by the low-cost of deployment; the components of *AMP (other than the operating system) are Open-Source applications, free software that can be used without purchasing a license.

Typical implementations of this are WAMP (Windows based), LAMP (Linux based) and MAMP (Apple Mac based). As of 2015, approximately 54% of the World Wide Web operated on Apache based servers (with Nginx, Microsoft-IIS and Google Servers making up for the rest).

High traffic (sites with lots of visitors) such as apple.com, paypal.com, adobe.com, twitter.com and many others make use of *AMP (and particularly LAMP) for the operation of their web sites.

MySQL is the key database component of *AMP and dominates online databases used to provide

services. There are over 10 million active installations of MySQL and it is the database technology that drives popular web tools such as Wordpress, Tumblr and Twitter.

3.4.3 Connecting to a database

As you know from previous topics, you have to connect to the database server and select the database to be used before you can execute MySQL statements. PHP has a number of commands which are used for this purpose.

The following PHP function call establishes the connection and selects the database required (or returns an error):

```
1 $mysqli = new mysqli($server, $username, $password, $database);
```

For example:

```
1 $mysqli = new mysqli("localhost", "root", "root","esports");
```

The items required to connect are:

- **server** the IP address or name of the server on which the database server software is running;
- **username** the username required to connect to the database server;
- **password** the password required to connect to the database server;
- **database** the name of the database, stored on the database server, that this connection will connect to.

The `mysqli` function shown above, for example, returns an object that includes details of the connection that has been established. We need to store this object so that we can make use of it later.

Since the MySQL server is a completely separate piece of software, we must consider the possibility that the server is unavailable, or inaccessible due to a network failure, or because the username/password combination you provided is not accepted by the server. In such cases, the `mysqli` function returns an object containing details of the error.

```
1 if ($mysqli->connect_errno) {
2       //if the connect_errno value is set then show the error.
3  echo "Failed to connect to MySQL: (" .
4  $mysqli->connect_errno . ") "
5  . $mysqli->connect_error;
6  die;
7 } else {
8  echo "Connected to database";  //is okay, we connected
9 }
```

There are two points to note about this code. `$mysqli` is an object created from the constructor method of the class `mysqli` and `connect_errno` is an instance variable set in this object by the class constructor method when a connection fails. The `->` is used in PHP to indicate we are referring to a value or function within the object.

The last point to note is the `die` function, which is your first example of a function that does not use parameters. All this function does is cause PHP to stop reading the source file at this point. This is a good response to a failed database connection, because in most cases the page will be unable to display any useful information without the database connection.

Key point

The username and password for the MySQL server will depend on how the software was installed. You may need to create a new user instead of the default root user.

Activity 3

You are going to create a *server side include*. This is a piece of code, a module or small library of code, that you can import into your main program. In this case the *include* will establish the database connection for us.

Start a new document in your editor. Delete any lines of HTML which might appear pre-written in your editor (such as <HTML> </HTML> etc.)

Enter the following text - **only** this text should appear in the file.

```
1  <?php
2  //connect to the database using mysqli API
3  $mysqli = new mysqli($host, $username, $pword, $database);
4  if ($mysqli->connect_errno) {
5    echo "Failed to connect to MySQL: (" .
6    $mysqli->connect_errno . ") "
7    . $mysqli->connect_error;
8    die;
9  }
10 ?>
```

Save this file in the root of your web server as *db_connection.php*.

Start a new HTML document and enter the following code:

```
1  <?php
2    //set up the connection variables for the db_connection.php
         include
3    $host="localhost";
4    $username="root";
5    $pword="root";
6    $database="esports";
7
8    require("db_connection.php"); //require the include code
9  ?>
10
```

```
<html>
  <head>
    <title>Database Connect</title>
  </head>
  <body>
    <?php
      //Just to check, let's get the name of the current database
      /* return name of current default database */

      //run the query method to get the database result object
      if ($result = $mysqli->query("SELECT DATABASE()")) {
        //read this from the query results object as an array of
           data
        $row = $result->fetch_row();
        //set the value to the first element [0] in the array
        $default_database = $row[0];
        //free the $result set (clear it)
        $result->close();
      }
    ?>

    <p>You are connected to the MySQL server and the
    <i><?=$default_database?></i> database.</p>
  </body>
</html>
```

Save this file as *database1.php*. View the PHP file using your browser.

You can insert the content of a file into a PHP file before the server executes it, with the `include()` or `require()` function. The two functions are identical in every way, except how they handle errors. The `include()` function generates a warning (but the script will continue execution) while the `require()` function generates a fatal error (and the script execution will stop after the error).

This script sets the connection values for the database (the host/server, username, password and required database) as variables which are then used in the include file. For future database connections you now can reuse the *db_connection.php* include file and set the database connection values as required.

The `include()` and `require()` functions are used to create functions, headers, footers, or elements that can be reused on multiple pages. This can save the developer a considerable amount of time. This means that you can create a standard header or menu file that you want all your web pages to include. When the header needs to be updated, you only have to update the include file, or when you add a new page to your site, you can change the menu file (instead of updating the links on all web pages).

3.4.4 Executing a query

All SQL queries are executed using the `mysqli->query()` method of the `mysqli` class. This executes the query and returns the result set as an object for SELECT queries. For other queries (e.g. creating a new table) it returns a true or false result depending on the success of the query.

Example

Return the number of rows that match a query:

```
1 /* Select queries return a resultset */
2 if ($result = $mysqli->query("SELECT game FROM game")) {
3     printf("Select returned %d rows.\n", $result->num_rows);
4     /* free result set */
5     $result->close();
6 }
```

The above code will display the number of rows from the game table.

Activity 4

Create a table and receive some feedback:

```
1 /* Create table doesn't return a resultset */
2 if ($mysqli->query("CREATE TABLE tmplogs (
3         id INT PRIMARY KEY,
4         username varchar(30),
5         time timestamp ) ") === TRUE)
6 {
7     printf("Table tmplogs successfully created.\n");
8 }
```

This code will create a table called *tmplogs*. If this is successful it will display the message "Table tmplogs successfully created."

MySQL also keeps track of the number of rows affected by `INSERT`, `DELETE` and `UPDATE` queries and the number of rows in the results of a `SELECT` query. This number can be accessed using the `mysqli->affected_rows` method.

So a script to run a query and trap any errors would be:

```
1  <?php
2
3  if ($mysqli->query($query_string)) {
4   echo ("Query successful with" . $mysqli->affected_rows .
5           "rows changed/accessed.");
6  } else {
7   echo ("Error performing query: " . $mysqli->error());
8  }
9
10 ?>
```

To see this working, replace $query_string with an SQL query such as "SELECT * FROM game"

3.4.5 Retrieving results

The result set from a database query is provided as an object, each row from the result set is also fetched as an object. The instance variables within the object can then be individually accessed.

So if the `game` table consists of the following data:

game	platform
Massive RPG	X-station
SuperJoe	S-box-360
Terra 1999	PC

The query `"SELECT * FROM game"` will return this as the result set from the code:

```
1 $query_string = "SELECT * FROM game";
2
3 if ($result = $mysqli->query($query_string)) {      //run the query
4 ...
5 }
```

The query is executed within the `if` condition, so if the query fails we can recover from it without crashing. If the query is successful, the contents of the `game` table will be held in the object called `$result`.

To read the rows and display the data from each row, we need to read a row at a time and display its values.

The code:

```
1 while ($row = $result->fetch_object()) {
2 ...
3 }
```

retrieves each row from `$result` (an object that holds the result set) and repeats the process until there are no rows left in the result set object.

In order to read the instance variables from each row object (these correspond to the column names) we use the code:

```
1 printf ("%s (%s)", $row->game, $row->platform);
```

The code `$row->game` returns the game column value for the current row and the `$row->platform` code returns the platform column value.

So by printing these values to the screen, the resulting web page output should be:

Massive RPG (X-station)

SuperJoe (S-Box-360)

Terra 1999 (PC)

Create the full script using the code below.

Activity 5

Create a copy of the file *database1.php* called *database2.php*. Amend the code as follows:

```
<?php
  $host="localhost";
  $username="root";
  $pword="root";
  $database="esports";
  require("db_connection.php");
?>

<html>
  <head>
    <title>Database Query Test</title>
  </head>
  <body>
    <?php
      //set up a query to use
      $query_string = "SELECT * FROM game";

      if ($result = $mysqli->query($query_string)) {  //run the
          query
        //fetch each row as an object
        while ($row = $result->fetch_object()) {
          //display the game and platform fields from the row
              object
          printf ("%s (%s)", $row->game, $row->platform);
          echo "<br />";  //start a new line
        }
        $result->close();  //free the $result set (clear it)
      } else {
        echo ("Error performing query: " . $mysqli->error() );
      }
      $mysqli->close(); //close the database connection
    ?>
  </body>
</html>
```

3.4.6 Adding, updating and deleting data

From your previous knowledge of SQL queries you will be able to create INSERT, UPDATE and DELETE statements in order to maintain your databases.

These queries can also be implemented through the use of PHP in our webpages. The next few examples will show you how apply these to the eSports database.

Download the SQL script and create the eSports database.

Download the `eSports` database here: *Please go online to download the file(s).*

3.4.6.1 Editing and amending data

To edit or change data in the database we need to first read the current records and then display the values to be edited in a suitable format.

A simple way to do this is to display each field from the table and then have an edit button which will display a form to update that specific record.

Activity 6

Create a copy of the file *database2.php* and call it *database3.php*. Amend the code as follows:

```
<?php
  $host="localhost";
  $username="root";
  $pword="root";
  $database="esports";

  require("db_connection.php");
?>

<html>
  <head>
    <title>Database Display</title>
    <style>
      table, td, th {border: 1px solid black;}
    </style>
  </head>
  <body>
    <?php
      //set up a query to use
      $query_string = "SELECT * FROM game";
    ?>

    <table>
      <thead>
        <th class="outline">game</th><th>platform</th><th></th>
      </thead>
      <tbody>
```

```
28          <?php
29
30            //run the query
31            if ($result = $mysqli->query($query_string)) {
32              while ($row = $result->fetch_object()) {
33                echo "<tr>";
34                echo "<td>" . $row->game . "</td>";
35                echo "<td>" .  $row->platform . "</td>";
36                echo "<td><a href=edit-data.php?game=" .
37                  urlencode($row->game).
38                  ">Edit</a></td></tr>";
39              }
40
41              //free the $result set (clear it)
42              $result->close();
43            }
44            $mysqli->close();
45          ?>
46        </tbody>
47      </table>
48    </body>
49  </html>
```

This code reads every row from the game table and displays each row with an **Edit** hyperlink.

game	platform	
Massive RPG	X-station	Edit
SuperJoe	S-box-360	Edit
Terra 1999	PC	Edit

Notice that the PHP function `urlencode` is used to encode the game values so that they are ready for passing using the `GET` method.

Clicking on the hyperlink will activate a script to display the row values in a form for editing. The script required for this is:

```
1   <?php
2     $host="localhost";
3     $username="root";
4     $pword="";
5     $database="esports";
6
7     require("db_connection.php");
8     $game = filter_var($_GET['game']);
9   ?>
10
11  <html>
12
13  <head>
```

```
14      <title>Database Update</title>
15  </head>
16
17  <body>
18      <table>
19          <tbody>
20              <?php
21                  //set up a query to use
22                  $query_string = "SELECT * FROM game WHERE
                        game.game = '". $game . "'" ;
23                  //run the query
24                  if ($result = $mysqli->query($query_string)) {
25                      while ($row = $result->fetch_object()) {
26                          ?>
27
28                       <form action="update-row-game.php"
                            method="get" name="updateform">
29
30                          <!-- keep track of the original primary
                                key value to update the row -->
31                          <input type="hidden" name="gameid"
                                value="<?php echo
                                urlencode($row->game); ?>">
32
33                          <!-- values which can be edited -->
34                          Game: <input type="text" value="<?php echo
                                $row->game; ?>" name="game"
                                size="40"><br />
35                          Platform:<input type="text" value="<?php
                                echo $row->platform; ?>"
                                name="platform" size="40"> <br />
36
37                          <!-- button to submit form -->
38                          <button class="submit"
                                type="submit">Update Row</button>
39                      </form>
40
41              <?php
42
43                      }  // end while
44                      $result->close();   //free the $result set
                            (clear it)
45                  } else {
46                      echo "Error on Game Table Query";
47                  }  // end if
48              $mysqli->close();
49              ?>
50      </tbody>
51      </table>
52  </body>
53  <html>
```

This code (save it as *edit-data.php*) produces a very simple web form containing the database values. Notice that a hidden input is used to hold the original value of the *primary key* `game`.

This is because if the primary key value is changed we need a record of the previous key value in order to find the row and update it. The form displays the data as follows:

These values can be edited in the form. e.g.

To process the result of this change we need to have a script to read the form data and update the database:

```
1  <?php
2    $host="localhost";
3    $username="root";
4    $pword="";
5    $database="esports";
6
7    require("db_connection.php");
8    $gameid = filter_var($_GET['gameid']);   //read the previous
       primary key
9    $gameid = urldecode($gameid);          //decode the previous key
       value
10   $game = filter_var($_GET['game']);       //read the value to be
       written for game
11   $platform = filter_var($_GET['platform']); //read the value to
       be written for platform
12 ?>
13
14 <html>
15   <head>
16     <title>Database Update</title>
17   </head>
18   <body>
19
20     <?php
21       //set up a query to use
22       $query_string =
23         "UPDATE game SET game = '". $game . "', platform = '" .
             $platform . "'
24         WHERE game.game = '". $gameid . "'" ;
25
26           //run the query
27           if ($result = $mysqli->query($query_string)) {
28             //show successful output
```

```
29              echo "Completed: Updated Row in Table ";
30                          echo "<br>";
31              //navigation back to records
32              echo "<a href=database3.php />Back</a>";
33            } else {
34              //show error because it's failed
35              echo "Error: Unable to Update Row in Table";
36            }
37            $mysqli->close();
38      ?>
39
40    </body>
41  </html>
```

Save this script as *update-row-game.php*.

Notice that the script reads all three values from the form using the `GET` method. The hidden primary key value, which was encoded in the previous script, is not decoded using the function `urldecode` and `filter_var` is used again to sanitize the values passed to the script.

An `UPDATE` query is used to set the new values for each of the columns and the old primary key value is used in the `WHERE` clause to locate the row so that it can be updated.

3.4.6.2 Insertion

Inserting values can be done in a similar manner to updating values. To do this we will add a link to the record display script to insert a new record, present a blank version of the update form and have a new script that will insert this new record.

Activity 7

Open *database3.php* and add the following line of code between the `</table>` and `</body>` tags (around line 47).

```
1   </table>
2
3     <a href="add-game-row.php">Insert a new row</a>
4
5   </body>
```

Now create a new script called *add-game-row.php* which will include this form code.

```
1   <html>
2     <head>
3       <title>Database Insert</title>
4     </head>
```

```
  <body>
    <form action="insert-row-game.php" method="get"
       name="insertform">
      Game: <input type="text" value="" name="game" size="40"><br
         />
      Platform:<input type="text" value="" name="platform"
         size="40"> <br />

      <!-- button to submit form -->
      <button class="submit" type="submit">Insert Row</button>
    </form>
  </body>
</html>
```

This is how the script will display the form:

To process the data from this form, create the following script called *insert-row-game.php* which will be called from the *add-game-row.php* script.

```
<?php
  $host="localhost";
  $username="root";
  $pword="";
  $database="esports";

  require("db_connection.php");
  //read the value to be written for game
  $game = filter_var($_GET['game']);
  //read the value to be written for platform
  $platform = filter_var($_GET['platform']);
?>

<html>

<head>
    <title>Database Insert</title>
</head>

<body>
    <table>
        <tbody>
            <?php
      //set up a query to use
      $query_string = "INSERT INTO game (game, platform)
        VALUES ('". $game . "','" . $platform . "')";
          //run the query
```

```
28            if ($result = $mysqli->query($query_string)) {
29              //show successful output
30              echo "Completed: Inserted Row in Table ";
31                          //navigation back to records
32                          echo "<a href=database3.php />Back</a>";
33            } else {
34              //show error because it's failed
35              echo "Error: Unable to Insert Row in Table<br />";
36              //display a list of the MySQL errors that occurred
37              print_r($mysqli->error_list);
38              //navigation back to records
39              echo "<br /><a href=database3.php />Back</a>";
40            }
41
42
43        $mysqli->close();
44      ?>
45          </tbody>
46      </table>
47  </body>
48
49  </html>
```

This script has some additions. The error checking is improved because now the script will report that an error has occurred (line 36) and also report the type of error or errors that have occurred (line 38). There is then an option to go back to the list of database rows.

3.4.6.3 Deletion

Deleting a row is can be done with from the table display script. To delete a row all that is required is the primary key value. If we add a `DELETE` link next to the `EDIT` link, this can be used to delete that specific row.

Activity 8

Update the script *database3.php* as follows.

Around line 25 edit the code to add an additional open and close `th` tag.

```
1  <th class="outline">Game</th><th>Platform</th><th></th><th></th>
```

Around line 39
(after

```
1  echo"<td><a href=edit-data.php?game=" . urlencode($row->game) .
2   ">Edit</a></td>";
```

) add the line:

```
echo "<td><a href=delete-data.php?game=" . urlencode($row->game) .
">Delete</a></td></tr>";
}
```

MAKE SURE to remove the </tr> from the end of the previous line of code, otherwise your table will display incorrectly.

The completed script will be:

```
<?php
  $host="localhost";
  $username="root";
  $pword="root";
  $database="esports";

  require("db_connection.php");
?>

<html>
  <head>
    <title>Database Query Player</title>
    <style>
      table, td, th {border: 1px solid black;}
    </style>
  </head>
  <body>
    <?php
      //set up a query to use
      $query_string = "SELECT * FROM player";
    ?>

    <table>
      <thead>
        <th class="outline">Username</th> <th>Realname</th>
           <th>Password</th>
        <th>Email</th> <th>Message</th> <th>Terms and
           Conditions</th>
      </thead>
      <tbody>

        <?php

          //run the query
          if ($result = $mysqli->query($query_string)) {
            while ($row = $result->fetch_object()) {
              echo "<tr>";
              echo "<td>" . $row->username . "</td>";
              echo "<td>" .  $row->realname . "</td>";
              echo "<td>" .  $row->pword . "</td>";
              echo "<td>" .  $row->email . "</td>";
              echo "<td>" .  mb_convert_encoding($row->message,
```

```
                'utf-8', 'iso-8859-1') . "</td>";
              echo "<td>" .  $row->terms_and_conditions . "</td>";
              echo "<td><a href=edit-data-player.php?username=" .
                urlencode($row->username) . ">Edit</a></td>";
              echo "<td><a href=delete-data-player.php?username=" .
                urlencode($row->username) .
                   ">Delete</a></td></tr>";
            }

            $result->close();  //free the $result set (clear it)
          }
          $mysqli->close();
        ?>
      </tbody>
    </table>

    <a href="add-player-row.php">Insert a new row</a>

  </body>
</html>
```

The table now displays as:

game	platform		
Massive RPG	X-station	Edit	Delete
SuperJoe	S-box-360	Edit	Delete
Terra 1999	PC	Edit	Delete

Insert a new row

This code will call the delete script with the encode primary key value for the row to be deleted.

Create a new script called *delete-data.php*.

```
<?php
  $host="localhost";
  $username="root";
  $pword="root";
  $database="esports";

  require("db_connection.php");
  $game = filter_var($_GET['game']);
?>

<html>
  <head>
    <title>Database Delete</title>
  </head>
  <body>
    <?php
```

```
17        //set up a query to use
18        $query_string = "DELETE FROM game WHERE game.game = '".
              $game . "'" ;
19
20        //run the query
21        if ($result = $mysqli->query($query_string)) {
22          echo "Success, the row for game=" . $game . " has been
                deleted.<br />";
23          //navigation back to records
24          echo "<br /><a href=database3.php />Back</a>";
25        } else {
26          //show error because it's failed
27          echo "Error: Unable to Delete Row in Table<br />";
28          //display a list of the MySQL errors that occurred
29          print_r($mysqli->error_list);
30          //navigation back to records
31          echo "<br /><a href=database3.php />Back</a>";
32        }
33
34        $result->close();  //free the $result set (clear it)
35        $mysqli->close();
36      ?>
37    </body>
38  </html>
```

When run, the following confirmation will appear:

Success, the row for game=Test123 has been deleted.

Back

3.4.6.4 Exercise

Player table

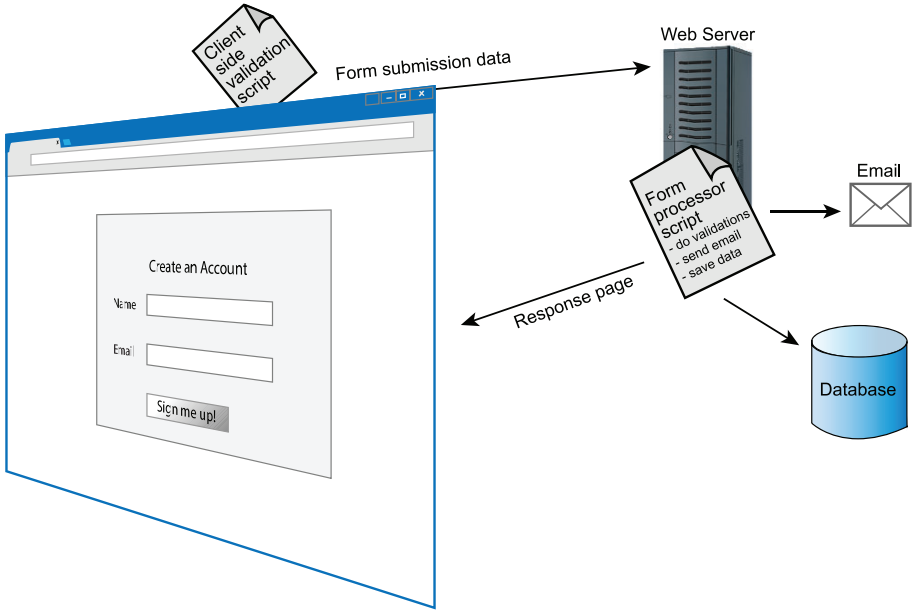
Using what you have learned from the previous exercises, complete scripts for the `player` table in the `esports` database which will allow records to be amended, inserted and deleted.

3.5 HTML forms

Forms in web development are typically used to submit data for processing. A web form has two parts:

1. An HTML ***front end***.
2. A ***back end*** form processor.

The front end operation occurs in the client (the browser) and the back end processing occurs on the server side.



The interactions with a web form are typically:

1. A user visits a web page that contains a form.
2. The web browser displays the HTML form.
3. The user completes the form and submits it.
4. The browser sends the submitted form data to the web server.
5. A form processor script running on the web server processes the form data.
6. A response page is sent back to the browser.

In the following activities you will create a simple HTML form, process the form data, explore issues of form security and how hackers exploit poorly designed web forms, introduce client side and server side validation of the form using PHP.

3.5.1 Form structure

Forms are created using the <form> HTML element. This element is used to open a collection of form associated elements (the things that form is made up of) and the closing tag </form> is used to end the collection.

Activity: Signup form - Stage 1

Create the following HTML code and save it in a file called *signup_form.php* in the root folder of your testing web server (e.g. htdocs or httpdocs).

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
   <title>Sign-up form</title>
    <link rel="stylesheet" media="screen" href="styles.css">
 </head>
 <body>    <!--start form for signup -->
    <form class="signup_form" action="" method=""
       name="signup_form">

      <!-- collection of form associated elements goes here -->

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

This simple HTML document contains a <meta> element to force it to use Unicode encoding, a title for the document and a link to a cascading style sheet that does not yet exist (we will create the HTML elements first then add the **CSS** and **JavaScript** later). The <form> element has a number of attributes:

- class (the CSS class applied);
- action (which will be the **URL** of the backend processing script);
- method (which will be how the data will be transmitted to the processing script);
- and a unique name to identify the form (i.e. "signup_form").

Form input elements

Forms may use a number of input elements. These can be text boxes, check boxes, radio buttons, password fields, buttons and many more.

Single line text box	A single line text box can be used to collect name, email and other details that normally occupy a single line.

The <input> element is used to create the text box using the *text* type.

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

As is the convention, because the *input* element doesn't have a closing tag, we use /> to close it.

Common attributes used with the **input** element:

- **type="text"** specifies that the browser should display a single line text input box;
- **name="firstname"** means that when the form is submitted the contents of this input will be referred to as firstname;
- **value="Please enter"** specifies a value to place in the text box when the form is created;
- **maxlength="60"** gives the text box a maximum number of characters that it can hold;
- **size="40"** the width of the text box in characters as it appears in the web page;
- **placeholder="Jenny Smith"** specifies a short hint that describes the expected value of the input and displays this hint in the input area before the user enters a value.

A complete list can be found at w3schools.com - http://www.w3schools.com/tags/tag_input.asp

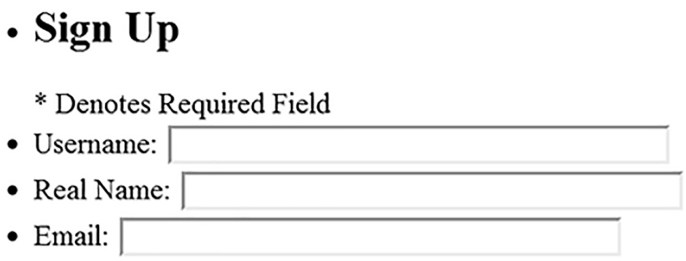
Activity: Signup form - Stage 2

Add the following code to the *signup_form.php* under the comment line `<!- collection of form associated elements goes here ->`

```
<ul>
    <li>
         <h2>Sign Up</h2>
         <span class="required_notification">* Denotes Required
            Field</span>
    </li>
    <li>
        <label for="username">Username:</label>
        <input id="username" type="text" name="username" value=""
           size="40"
         maxlength="40" placeholder="jensmith72" />
    </li>
    <li>
        <label for="realname">Real Name:</label>
        <input id="realname" type="text" name="realname" value=""
           size="40"
         maxlength="40" placeholder="Jenny Smith" />
    </li>
    <li>
        <label for="email">Email:</label>
        <input id="email" type="email" name="email" value=""
           size="40"
         maxlength="60" placeholder="jenny.smith@example.com"/>
    </li>

    <!-- end of user ID sign up fields -->
</ul>
```

This code produces a form which looks like this:



Hints can be added to the form to assist users when completing it, for example:

```
<li>
 <label for="email">Email:</label>
 <input type="text" name="email" value="" size="40" maxlength="60"/>
 <span class="form_hint">Use the format "name@domain.com"</span>
</li>
```

Activity: Signup form - Stage 3

Add suitable hints for the three single line text inputs in the form using the code `<span class="form_hint">Use the format "name@domain.com"</span>` above.

Change the contents of the hint to be appropriate to each input.

- A username should contain only alphabetic characters and numbers.
- A real name should be at least a firstname and lastname.
- An email address must be in a valid format.

It should look something like this.

To complete the form, we will add a small text area for multi-line text input, a check box to agree to some terms and conditions and a submit button.

Text area	A multi line text area can be used to collect an extended amount of text that would occupy multiple lines.

The <textarea> element is used to create the text area.

```
1 <textarea name="message" cols="40" rows="6">
```

Common attributes used with the **textarea** element:

- **name="message"** means that when the form is submitted the contents of this textarea will be referred to as 'message';
- **cols="40"** sets how wide the textarea will be in terms of number of characters;
- **rows="6"** sets the number of visible lines of text in the text area.

A complete list can be found at w3schools.com - http://www.w3schools.com/tags/tag_textarea.asp

Button	A clickable button used to perform an action of some sort.

The <button> element is used to create a button. Unlike the button which can be created with the <input> element, the <button> element can hold content, like text or images.

```
1
2
3 <button class="submit" type="submit">Sign Up</button>
```

Common attributes used with the **button** element:

- **type="**<**value**>**"** details the type of button with a value of:
  - "**button**" creates a button without a form action which can be used with JavaScript and an OnClick function;
  - "**reset**" creates a reset button that clears the values entered in the form;
  - "**submit**" creates a button that sends the forms contents for processing.

A complete list can be found at w3schools.com - http://www.w3schools.com/tags/tag_button.asp

Activity: Signup form - Stage 4

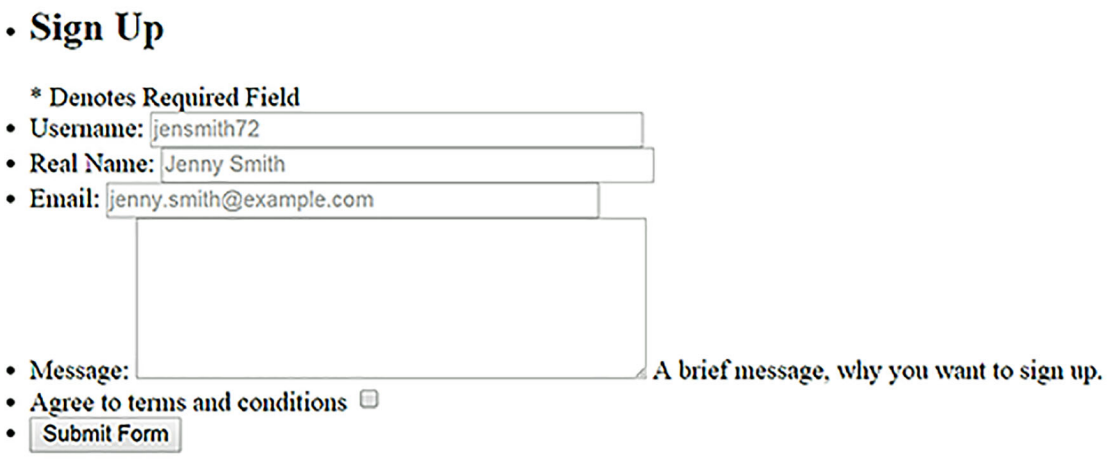
Amend the *signup_form.php* file to include the following lines of code after the HTML comment `<!- end of user ID sign up fields ->` and before the closing tag.

```
<li>
 <label for="message">Message:</label>
 <textarea id="message" name="message" cols="40" rows="6"
    ></textarea>
 <span class="form_hint">A brief message, why you want to sign
    up.</span>
</li>
<li>
 <label for="terms_and_conditions">Agree to terms and
    conditions</label>
 <input id=" terms_and_conditions" type="checkbox"
    name="terms_and_conditions">
</li>
<li>
 <button class="submit" type="submit">Submit Form</button>
</li>

<!-- end of additional fields -->
```

Preview the form in your web browser using the server url (e.g. *http://localhost/signup_form.php*).

The form will now appear as:



Applying CSS rules to the form

Create a stylesheet file called *styles.css* and save this in the same location as your *signup_form.php* file. Enter the following rules into the file.

```
@charset "UTF-8";
/* CSS Document */

/* set default fonts and styles for type */
body {
  font: 14px/21px "Lucida Sans", "Lucida Grande", "Lucida Sans Unicode",
 sans-serif;
}

.signup_form h2, .signup_form label {
  font-family:Georgia, Times, "Times New Roman", serif;
}
.required_notification {
  font-size: 11px;
}

.form_hint {
  font-size: 11px;  vertical-align:top;
}

/* remove the focus style which looks odd */
*:focus {
  outline: none;
}

/*make the form fields more attractive by redefining the list style*/
.signup_form ul {
    width:950px;
    list-style-type:none;
    list-style-position:outside;
    margin:0px;
    padding:0px;
}
.signup_form li{
    padding:12px;
    border-bottom:1px solid #eee;
    position:relative;
}

/*add some visual style to the top and bottom of the form*/
.signup_form li:first-child, .signup_form li:last-child {
    border-bottom:1px solid #777;
}

/*add a better header style and put the "required" field on the right*/
.signup_form h2 {
    margin:0;
    display: inline;
}
.required_notification {
```

```
51      color:#d45252;
52      margin:5px 0 0 0;
53      display:inline;
54      float:right;
55  }
56
57  /*space out the form input elements to make them more attractive*/
58  .signup_form label {
59      width:200px;
60      margin-top: 3px;
61      display:inline-block;
62      float:left;
63      padding:3px;
64  }
65  . signup_form input {
66      height:20px;
67      width:220px;
68      padding:5px 8px;
69  }
70
71  .signup_form textarea {
72    padding:8px; width:300px;
73  }
74
75  .signup_form button {
76    margin-left:156px;
77  }
78
79  /* add some enhanced visual styles */
80  .signup_form input, .signup_form textarea {
81      border:1px solid #aaa;
82      box-shadow: 0px 0px 3px #ccc, 0 10px 15px #eee inset;
83      border-radius:2px;
84  }
85  .signup_form input:focus, .signup_form textarea:focus {
86      background: #fff;
87      border:1px solid #555;
88      box-shadow: 0 0 3px #aaa;
89  }
90  /* Button Style */
91  button.submit {
92      background-color: #68b12f;
93      background: -webkit-gradient(linear, left top, left bottom,
94      from(#68b12f), to(#50911e));
95      background: -webkit-linear-gradient(top, #68b12f, #50911e);
96      background: -moz-linear-gradient(top, #68b12f, #50911e);
97      background: -ms-linear-gradient(top, #68b12f, #50911e);
98      background: -o-linear-gradient(top, #68b12f, #50911e);
99      background: linear-gradient(top, #68b12f, #50911e);
100     border: 1px solid #509111;
101     border-bottom: 1px solid #5b992b;
102     border-radius: 3px;
103     -webkit-border-radius: 3px;
104     -moz-border-radius: 3px;
105     -ms-border-radius: 3px;
106     -o-border-radius: 3px;
```

```
107     box-shadow: inset 0 1px 0 0 #9fd574;
108     -webkit-box-shadow: 0 1px 0 0 #9fd574 inset ;
109     -moz-box-shadow: 0 1px 0 0 #9fd574 inset;
110     -ms-box-shadow: 0 1px 0 0 #9fd574 inset;
111     -o-box-shadow: 0 1px 0 0 #9fd574 inset;
112     color: white;
113     font-weight: bold;
114     padding: 6px 20px;
115     text-align: center;
116     text-shadow: 0 -1px 0 #396715;
117 }
118 button.submit:hover {
119     opacity:.85;
120     cursor: pointer;
121 }
122 button.submit:active {
123     border: 1px solid #20911e;
124     box-shadow: 0 0 10px 5px #356b0b inset;
125     -webkit-box-shadow:0 0 10px 5px #356b0b inset ;
126     -moz-box-shadow: 0 0 10px 5px #356b0b inset;
127     -ms-box-shadow: 0 0 10px 5px #356b0b inset;
128     -o-box-shadow: 0 0 10px 5px #356b0b inset;
129 }
130
131 /* interactive CSS3 */
132 .signup_form input:focus, .signup_form textarea:focus {
133 /* add this to the already existing style */
134     padding-right:30px;
135 }
136
137 .signup_form input, .signup_form textarea {
138 /* add this to the already existing style */
139     -moz-transition: padding .25s;
140     -webkit-transition: padding .25s;
141     -o-transition: padding .25s;
142     transition: padding .25s;
143 }
```

This CSS code significantly enhances the visual appearance and behavior of the form. Finally, before we deal with the submission and processing of the form, we can use the HTML5 required attribute to tell the browser that an input/textarea element must have a value before the form can be submitted e.g.

```
<input type="text" name="username" value="" size="40" maxlength="40"
placeholder="jensmith72" required />
```

Activity: Signup form - Stage 5

Add the **required** attribute to the inputs for username, realname and email. Leave the message textarea as an optional item.

Some additional styling can be added to these fields. These rules will add a red asterisk to the background of each required field.

Activity: Signup form - Stage 6

Add the following rules to your *styles.css* file.

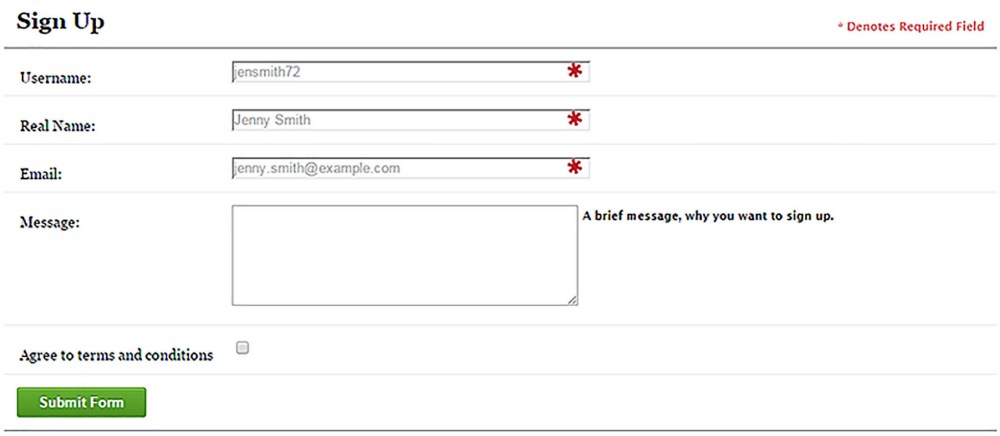
```
/*add red asterisk for the required elements*/
.signup_form input, .signup_form textarea {
    padding-right:30px;
}

input:required, textarea:required {
    background: #fff url("images/red-asterisk.png") no-repeat 98%
        top;
}
```

Download this image of a red asterisk and place it in a subfolder called "images" inside your web folder. Name the downloaded file "red-asterisk.png" to match the CSS rule.

red-asterisk.png *Please go online to download the file(s).*

The page should now look something like this:



3.5.2 Form submission

Forms are submitted when a submit event occurs, either triggered via JavaScript or when a Submit button is activated. A form is submitted to a specific script on the server. This script is identified in the action attribute of the form element. The action specifies the URL of the processing script.

Activity: Signup form - Stage 7

Amend the **action** attribute of the <**form**> element so that it points to a script called **process_signup.php** in the same location as the *signup_form.php* file.

```
1 action="process_signup.php"
```

3.5.3 Methods

The **method** attribute of the <**form**> element specifies how the form-data - the data entered in the form collection and sent to the processing script specified by the **action** attribute - is sent. The form-data is sent either as a URL containing the values (with the method="get") or as an HTTP post transaction (with the method="post").

The GET method

The GET method adds the list of values from the form to the URL of the processing script. For example, if we were processing a form with two fields, the GET method would send the data as:

```
http://myserver.com/processscript.php?username=jensmith72&realname=Jenny%20Smith
```

The values from the form are visible in the URL so are insecure. The string on values starts with ? (question mark) and each name/value pair is joined using a & (ampersand). Notice the %20 in the string where the space would have been. Non-text characters are converted to unicode Hexadecimal values. You can review a list of Unicode UTF-8 values here (http://www.utf8-chartable.de/)

Key points about the GET method:

- Appends form-data into the URL in name/value pairs (delimitated by &).
- The length of a URL is limited (to roughly 3000 characters).
- Not suitable for sensitive information because the data is visible in the URL.
- Useful for instances where the data will be part of a bookmark/link to a page.
- Useful when debugging because the values passed to the processing script are visible.

The POST method

The POST method adds the form data inside the body of the HTTP request to the processing script so the data is not visible in the URL. This is generally more secure than the GET method.

Key points about the POST method:

- Adds the form data to the HTTP request to the processing script.
- Has no size limitations on the amount of data that can be submitted.
- form submissions with POST cannot be bookmarked.

Activity: Signup form - Stage 8

Amend the **method** attribute of the <**form**> element so that it uses the **get** method.

```
method="get"
```

Now that all the HTML for the form is complete, let's add a short PHP script to process it on the server site.

Activity: Signup form - Stage 9

Create a file called ***process_signup.php*** in the same location as the *signup_form.php*. Enter the following PHP code. You will have some experience of server-side development from Higher Computing Science.

```
1 <?php
2 /* script to process details from the submitted form */
3
4 //read in the values from the form
5 $username = $_GET['username'];
6 $realname = $_GET['realname'];
```

```
$email = $_GET['email'];
$message = $_GET['message'];
$terms_and_conditions = $_GET['terms_and_conditions'];
?>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title><?php echo $message;?></title>
    <link rel="stylesheet" media="screen" href="styles.css">
  </head>
  <body>
   <!-start form response -->
   <ul>
     <li>
       Username: <?php echo $username;?>
     </li>
     <li>
         Realname: <?php echo $realname;?>
     </li>
     <li>
       Email: <?php echo $email;?>
     </li>
     <li>
       Message: <?php echo $message;?>
     </li>
     <li>
       <?php
         if($terms_and_conditions == "on" ) {
          echo "User has agreed to the terms and conditions";
         } else {
          echo "User has not agreed to the terms and conditions";
         }
       ?>
     </li>
   </ul>
  </body>
</html>
```

At the moment this form just displays the values sent from the form. Test the script by loading the form, entering data and submitting it. The output should be as follows:

- **Username: ragdoll619**
- **Realname: Jenna Sauders**
- **Email: jensos18@hotmail.com**
- **Message: I wanted to join the gaming club.**
- **User has agreed to the terms and conditions**

This is a simple script and it isn't very secure. On some web browsers, it can be exploited to either inject code to carry out actions on a user's web page OR to display HTML that was never intended

to be displayed. You are going to create some harmless code to "attack" the server.

3.5.4 Form security

The processing script receives values from the form. Using a number of coding "tricks" it is possible to enter data that could be used to exploit the server.

Code injection attacks

A **code injection** attack is when a weakness in poorly written code is used by an attacker to inject code into a vulnerable script and change the execution of the script.

Modern web browsers are more sophisticated at preventing these kind of attacks. The browser checks the source code of the page with the code to be executed to ensure all the code to be executed is valid.

Try the following harmless example of code injection.

Activity: Signup form - Code injection

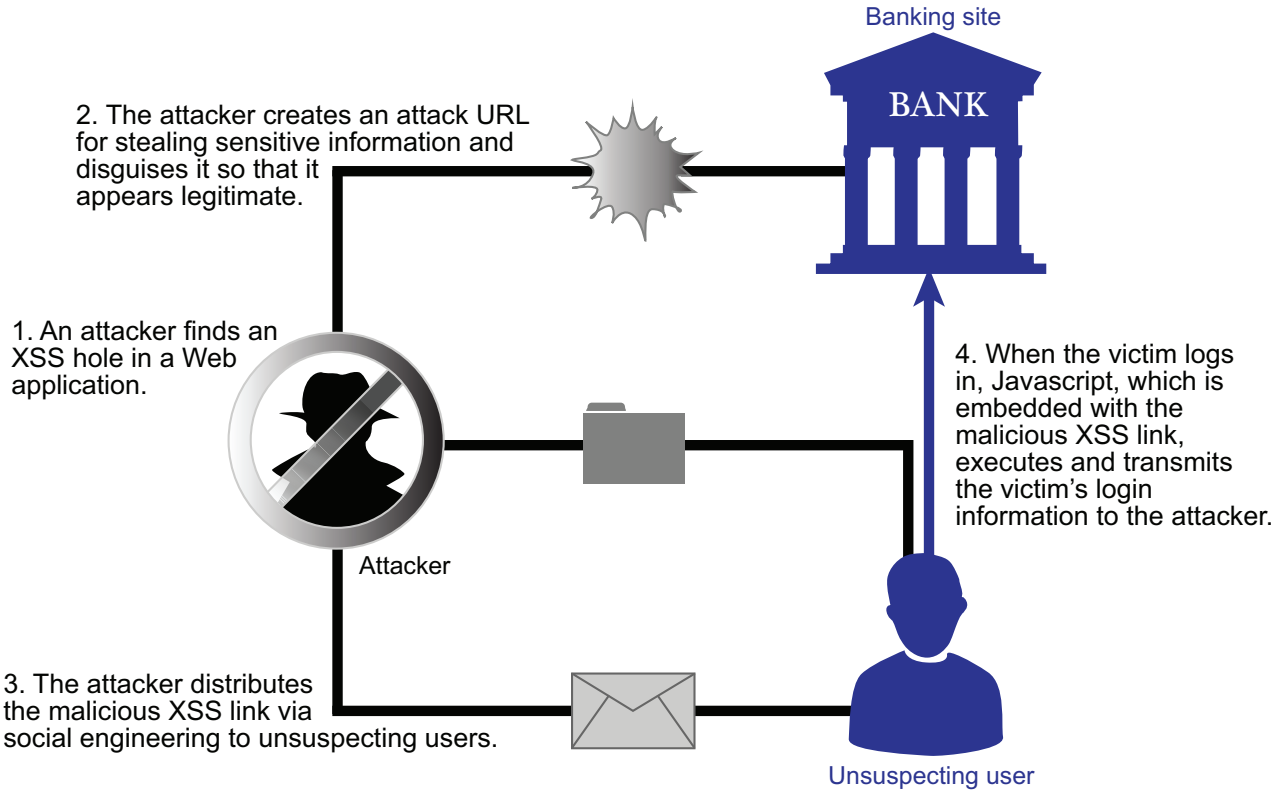
Load the form again and enter the following values.

Username:	ragdoll619
Real Name:	Jenna Sauders
Email:	jensos18@hotmail.com
Message:	example</title></head><body><img src=http://charlielove.org/htp.png></body></html>

Now submit the form. The fields are not checked for HTML so this code allows an image to be injected into the web page. You can try other HTML elements including links / anchors in the message textarea.

Cross-site scripting

Cross-site scripting (**XSS**) is a type of attack carried out on web applications. It allows hackers to inject client-side script into a web page that others can view. Cross-site scripting uses gaps in the security of web applications to allow malicious content to be delivered from a compromised site. When the user visits the compromised page, information can be harvested by the attacker.



3.5.5 Validation

Validation is the key to protecting against code injection and cross-site scripting attacks. By validating the form data, we can protect against these exploits.

Client side validation techniques

Any validation carried out on the client site, using either JavaScript or another client technology, can be subverted. There are tools, such as interception proxy servers, that intercept the data between the client and the server and allow the hacker to change the requests before they can be sent effectively rendering all client side validation for the purpose of security useless.

However, for the typical user, carrying out some kind of client side validation is useful as it can catch errors before data is sent to the server.

Activity: Signup form - Validation

HTML5 carries out validation based on the type attributes of form collection elements. There is no specific markup required in order to activate form validation - it is on by default.

The first field, username, is text and is required. The web browser will validate the field so that it must not be empty and contains characters. As long as the user entered al least one character the field it will validate.

We can use the **:valid** and **:invalid** CSS selectors to format valid and invalid fields providing visual information to the user.

Download each of these files and place them in your images folder.


valid.png


invalid.png

Please go online to download the file(s).

Add the following CSS rules to your styles.css file.

```
.signup_form input:focus:invalid, .signup_form
    textarea:focus:invalid {
     /* when a field is considered invalid by the browser */
     background: #fff url("images/invalid.png") no-repeat 98%
         center;
     box-shadow: 0 0 5px #d45252;
     border-color: #b03535
}

.signup_form input:required:valid, .signup_form
    textarea:required:valid {
/* when a field is considered valid by the browser */
     background: #fff url("images/valid.png") no-repeat 98% center;
     box-shadow: 0 0 5px #5cd053;
     border-color: #28921f;
}
```

Load the form and fill out the values to see how the use of the :valid and :invalid selectors affects the form.

Further validation can be performed using the HTML5 **pattern** attribute. The username must be a combination of only letters and numbers; no special characters or spaces are allowed.

Activity: Signup form - Validation 2

Add the **pattern** attribute and the regular expression **^[A-Za-z0-9]+$** to the input element for username. A **regular expression** is a method of declaring rules to match string contents.

```
<input id="username" type="text" name="username" value="" size="40"
maxlength="40" placeholder="jensmith72" pattern="^[A-Za-z0-9]+$"
    required />
```

Test the code and ensure that the username is only valid when it consists of letters and/or numbers.

Activity: Signup form - CSS formatting

To complete the formatting of the form add the following CSS rules to your *styles.css* file.

```
.form_hint {
  background: #d45252;
  border-radius: 3px 3px 3px 3px;
  color: white;
  margin-left: 8px;
  padding: 1px 6px;
  z-index: 999; /* hints stay above all other elements */
  position: absolute; /*allows proper formatting if hint is two
     lines*/
  display: none;
}

.form_hint::before {
  content: "\25C0"; /* left point triangle in escaped unicode */
  color: #d45252;
  position: absolute;
  top: 1px;
  left: -6px;
}

.signup_form input:focus + .form_hint {display: inline;}

/* change form hint color when valid */
.signup_form input:required:valid + .form_hint {background:
    #28921f;}


/* change form hint arrow color when valid */
.signup_form input:required:valid + .form_hint::before
    {color:#28921f;}
```

This CSS formats the hints so they are only shown when editing a field.

HTML5 can be used to carry out validation on the client side however every input must be validated on the server side and made suitable for use.

Server Side validation techniques

Variables passed from a form normally go through two processes on the server side before they can be reliably used. Data passed using the GET or POST methods must never be used without first being sanitized and validated (if required).

- Sanitizing data - removing any illegal characters from the data.
- Validating data - determining if the data is in proper form.

Sanitization of form data

Sanitization is the process of removing any HTML entities from values passed to the processing script. In PHP this is relatively easy to do using **filters**.
You can read more about PHP filters at http://www.w3schools.com/php/php_ref_filter.asp

Activity: Signup form - Sanitization

Change the code in the *process_signup.php* form so that it correctly sanitizes the fields. Replace lines 4 to 9 with the following code.

```
1 //read in the values from the form
2 $username = filter_var($_GET['username'], FILTER_SANITIZE_STRING);
3 $realname = filter_var($_GET['realname'], FILTER_SANITIZE_STRING);
4 $email = filter_var($_GET['email'],FILTER_SANITIZE_STRING);
5 $message = filter_var($_GET['message'], FILTER_SANITIZE_STRING);
6 $terms_and_conditions = filter_var($_GET['terms_and_conditions'],
7 FILTER_SANITIZE_STRING);
8
9 //end of reading in values and sanitizing them
```

This code removes all HTML, JavaScript and other code from the text.

Validation

The username and email address fields need to be validated. This can be done using a PHP function. If either of these fields is invalid, then we will force the user to return to the original form.

Activity: Signup form - Validation 3

Add the code to validate the email address. At the top of the *process_signup.php* file, on a new line after the <**?php** enter the following PHP function.

```
1  //function to validate username and email address
2  function validate_form($username, $email)
3  {
4    if(!ctype_alnum($username)||!filter_var($email,
         FILTER_VALIDATE_EMAIL))
5   {
6      //one of these is invalid so return false;
7      return false;
8    }
9    return true;
10 }
```

This function uses the PHP type `ctype_alnum` to validate the username because this requires that $username be alphanumeric only and the PHP filter, FILTER_VALIDATE_EMAIL, is used

for the email address. Should either of these be invalid then a value of false is returned from the function. If they are valid then true is returned.

Add the following code under `//end of reading in values and sanitizing them`

```
1 //check if $username or $email are invalid,
2 //if they are redirect back to the form.
3 if (!validate_form($username, $email)) {
4   //there is an error so go back to where we came from
5   Header('Location: signup_form2.php');   }
```

Now test the form again. Change the values in the URL to introduce errors for the script *process_signup.php* to manage. Remember to test to ensure your form values are sanitized to prevent code injection.

Activity: Signup form - POST

The POST method can also be used to pass values. Use the POST method and the PHP function $_POST to create an alternative version of the form and the processing script which makes use of the POST method.

3.6 PHP sessions

Each time you log in to a website that you visit a session is created. This will hold certain information about you while using the site. This can help keep track of things like what page you were last on, what your username is for the site as these can often be displayed on the pages for the user to see.

All session variables are stored in what's known as a **superglobal** array. Unlike variables that we use with `GET` and `POST`, a `SESSION` variable will retain it's values on every page the user visits on the site / application until they log out and the `SESSION` variable is cleared.

The code to start a session is placed at the start of the web page.

```
1 <?php
2   // start the session
3   session_start();
4 ?>
```

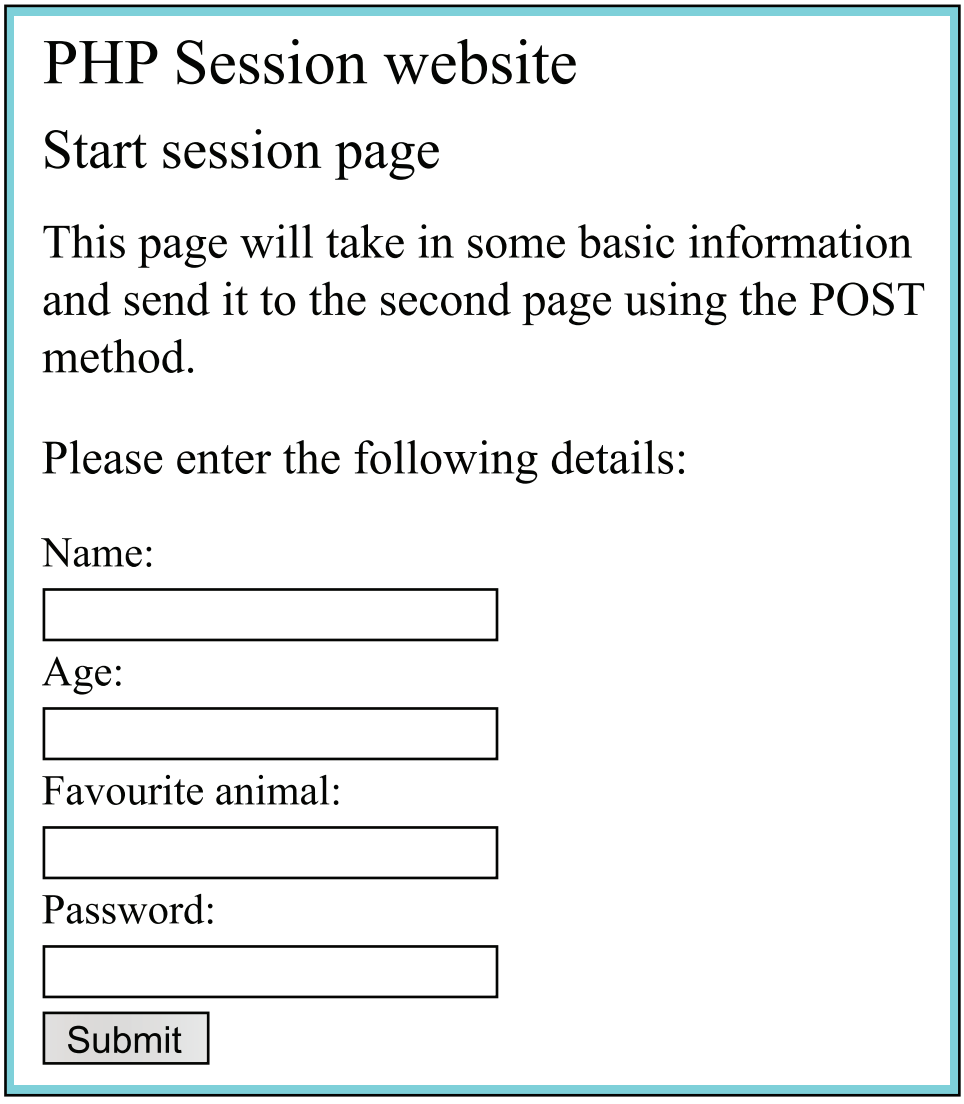
This code block should be placed in every page that will make use of the session variables.

To end a session you would use the destroy function. This will clear all session variables from the application. You may want to store some of the session variables to a file for the user on the server as this can help personalise the application for the user next time they log in. For example each time you log in to Scholar it tells you what you last worked on and helps you get straight back to that section.

To see these in action, download the *Session* website.

Download the database here: *Please go online to download the file(s).*

It is currently made up of five PHP pages and a stylesheet. It makes use of many of the skills we've looked at in the topic already and adds the use of session variables.



Session variables

Try adding more inputs to the form for the user to enter and then use these on different pages in the site until you are comfortable using session variables.

3.7 Summary

Summary

You should now be able to:

- **CSS**
  describe, exemplify, and implement responsive pages using the following queries:
  - media type:
    - print;
    - screen;
  - media feature:
    - max-width;
- **HTML**
  describe, exemplify, and implement form elements including:
  - FORM element:
    - action;
    - method (get and post);
  - INPUT, SELECT and TEXTAREA elements:
    - name;
    - value;
  - TABLE element:
    - th;
    - tr;
    - td;
- **PHP**
  describe, exemplify, and implement coding of server-side processing to:
  - assign form data to server-side variables:
    - $_get();
    - $_post();
  - open and close connection to database server:
    - die();
    - mysqli_connect();
    - mysqli_close();
  - execute SQL query:
    - mysqli_query();
  - format query results:
    - echo;
    - mysqli_fetch_array();
    - mysqli_num_row();

Summary continued

and:

- assignment, repetition and selection using server-side local and global variables;
- sessions:
  - session_start();
  - session_destroy();

- read and explain code that uses constructs appropriate to this level.

3.8 End of topic test

End of topic test: Implementation Go online

Q6: When using media queries to format the display of a website, what is the name given to the user's visible area of the screen?

a) Screensize
b) View
c) Viewport
d) Screen-width

. . . . . .

Q7: PHP code is downloaded to the user's computer and decoded on their device.

a) True
b) False

. . . . . .

Q8: The PHP *echo* command:

a) displays output to a text file.
b) displays output.
c) duplicates a variable.
d) writes a variable as HTML.

. . . . . .

Q9: When passing values between pages we can use the GET and POST methods. Which method is more secure for transmitting passwords or sensitive data from the client to the server.

. . . . . .

Q10: Which four parameters are required to connect to a database server using the MySQLi API?

. . . . . .

Q11: A PHP page has been written to create a connection to the *customers* database. It will connect to the database using the *root* username, but no password has been specified.

```
1 <?php
2   $host="localhost";
3   $username="root";
4   $pword="";
5   $database="customers";
6   // import db_connect.php
7 ?>
```

Line 6 has a comment to import the database connection code for "db_connect.php". Write the line to import the file and raise a fatal error if the file does not exist.

. . . . . .

Q12: A website wants to personalise the home page for each visitor by including their name (e.g. "Welcome to Scholaria Gladys") within the main heading <h1>.

Write the full PHP statement to concatenate the message "Welcome to Scholaria" and the visitor's name from a $User variable, before displaying it on the page. Remember to also include the heading style.

.....

Q13: Which of these is the correct PHP code required to execute a query stored in the variable `$query`?

a) `$mysqli->query($query);`
b) `$result->fetch_object($query);`
c) `$mysqli->prepare($query);`
d) `$result->mysqli->error($query);`

.....

Q14: Which function should be used to sanitize POST and GET values submitted by a form?

a) urlencode
b) htmlspecialchars
c) filter_var
d) sanitize

Topic 4

Testing and evaluation

Contents

Prerequisites

From your studies in Higher Computing Science you should already be able to:

- understand and perform usability testing making use of:
  - personas;
  - test cases;
  - scenarios based on low fidelity prototypes;
- describe and perform testing on:
  - input validation for data being entered into a form (Client Side Scripting);
  - the site's navigational bar both displays correctly and all links work;
  - all of the media content displays correctly;
- understand and perform compatibility testing to ensure that websites display properly on a variety of different devices:
  - tablet;
  - smart phone;
  - desktop;
- understand the importance and perform compatibility testing to ensure that websites display properly on a variety of different software:
  - operating systems;
  - web browsers.
- evaluate:
  - if your website is fit for purpose;
  - your website's usability.

Learning objective

By the end of this topic you should be able to:

- describe, exemplify, and implement the following:
  - integrative testing;
  - usability testing based on prototypes;
  - final testing;
  - end user testing;
- evaluate a software solution in terms of:
  - fitness for purpose;
  - maintainability:
    - perfective;
    - corrective;
    - adaptive;
  - robustness;
  - usability.

4.1 Testing

Testing any new or updated system, application, website or database is a very important step within the development cycle of a product. The aim is to remove any bugs and errors before it is delivered to the end user.

If bugs still exist within the system after it has been released, users will become frustrated when they cannot perform their tasks. It may not take long before comments are added to social media and the resulting costs, both in terms of the time required to correct the problems and the developer's reputation, can be very big.

When testing our web development projects we will focus on:

- Integrative (integration) testing.
- Usability testing.
- Final testing.
- End user testing.

Key point

Even if we've performed all of our testing there are sometimes unforeseen or unexpected errors with web development projects. Such as:

- web sites may fail to load due to Denial of Service (DOS or DDOS) attacks;
- updates to servers or server software can cause existing projects to stop working;
- the end user receives a 404 error (page not found) after clicking a link to a page because it has been moved or deleted;
- external files that the site requires to run become unavailable.

4.1.1 Integrative testing

The websites you have been designing are made up of many components such as HTML, CSS and JavaScript. From your studies at Higher you should be familiar with the need to perform ***functional testing*** on each of these elements individually. **Integrative testing** ensures they all work well together in order to produce a fit for purpose website. If an element needs to be tested but is not yet finished, developers may use **stubs** and **drivers** to check its operation.

As our projects and applications get bigger and more complex through the inclusion of web applications, software and databases, fully testing everything to ensure they all work together is even more important. You could spend a lot of time on the style of your web pages, but it won't help your visitors if the content is not displayed from a database.

An integrative test plan for web design will identify:

- The original web site specification.
- Any wireframe designs or prototypes (low and high fidelity) for the website.
- Any test documentation from the functional testing stage.

- If the website can successfully communicate/connect with the external software or database application.
- A list of conditions which must be in place before the testing starts.
- Details of what constitutes a successful test.

Integrative testing is done in-house by members of the development team, or by a testing group appointed by the developers.

4.1.2 Usability testing

Again as you might guess from the name, this type of testing is designed to ensure that the software or website is as easy to understand and use as possible, and that the interface is suitable for everyone who might be using it. The following criteria can be used to test and evaluate a user interface to a piece of software:

- **Appropriate**:
  The interface should be designed with the type of user and the tasks they will be performing in mind. Ideally it should mirror the real world as much as possible, so that it uses language and concepts familiar to the user. Exactly how this is done can be a matter of fashion as well as functionality. The term **skeuomorphism** refers to the way some interfaces try to mimic their physical counterparts. A radio app for instance, may have controls which look like tuning knobs and switches even though they do not aid its functionality and in some cases hinder it.
- **Customisable**:
  The interface must be able to be changed to suit the needs of the user, without compromising its functionality. A customisable interface provides shortcut keys for experienced users, with menus and dialogue boxes for novice users. Users should be able to add frequently used tools to a menu or toolbar and change colours or contrast if required. If the software is to be sold and used in different countries, regional versions should be available in local languages and character sets.
- **Accessible**:
  The interface must be customisable to the extent that users who need a **screen reader**, require high contrast screen displays or who have other eyesight problems can still operate it comfortably.
- **Consistent**:
  The interface should provide menus and dialogue boxes so that commands and other operations are grouped together in a logical way. For example, the contents of the file and edit menus in many applications will all contain many of the same familiar options.
- **Controllable**:
  The interface should support the user so that they always have the option of undoing crucial operations. There should always be an "escape" from a sequence of operations (e.g. a cancel button).
- **Helpful**:
  The interface should provide help on request, and it should be available in the context of the task being undertaken. Documentation should be comprehensive and always be available if required.

History of user interfaces Go online

If you have access to the internet why not watch this video on the history of user interfaces: https://www.youtube.com/watch?v=U1Oy4X5Ni8Y

4.1.3 Alpha testing

Alpha testing is the stage of the development cycle where the full software is first able to run but it may not contain all the features that are planned for the final version. Alpha testing usually takes place on the software developer's premises.

Alpha testing is typically done for two reasons:

1. To reassure clients that the software is in a working condition but not released to them.
2. To find errors that may only be found under operational conditions.

Alpha testing is performed on an early version of the software. However since it will not have all the intended functionality, it will have core functions and will be able to accept inputs and generate outputs in accordance with user specifications.

Usually, the most complex or most used parts of the code are developed more completely during the alpha phase, in order to enable early resolution of design questions that might arise. Nevertheless due to the very nature of testing an entire system, alpha testing traditionally occurs quite late in the software development cycle. Unfortunately in many cases, time for testing is squeezed very tightly at the end of the development. Thus compromises in the quality of testing procedures are made.

Ideally alpha testing should be conducted with as much independence as possible from the development team and any of the personnel that performed other forms of testing up to that point. Independent scrutiny can bring a fresh view point to the project.

However in the real world, not many organisations have the resources for an independent alpha test team. Consequently testers have to wear both hats. An alternative is to co-opt personnel closer to the customer, such as consultants and implementers to do the testing but this, consequently adds to the overall costs of the project.

Questions on alpha testing

Consider whether each of these statements is true or false:

Q1: Alpha testing is the testing of a complete program or system.

a) True
b) False

. . . . . .

Q2: Alpha testing checks for errors and if any are found they are not fixed.

a) True
b) False

. . . . . .

Q3: Alpha testing is carried out by end users of a program.

a) True
b) False

4.1.4 End user (beta) testing

End user testing is testing which is done by the client in the case of bespoke software systems, or by potential customers in the case of commercial software, before it has been officially released to users. As its name suggests, end user testing allows the client / customer to check the system is fit for purpose. If it meets all the requirements the client will accept the system, otherwise further changes will need to be made by the developer.

End user testing is sometimes called **beta testing** to distinguish it from **alpha testing**.

Alpha testing is important, but because it is done by programmers who may be associated with the project, their tests can suffer from unintentional bias or just the inability to imagine the misunderstanding which a novice user may experience. Some software companies release alpha versions of their software to the public to get feedback relatively early in the development of the software. When this happens the company make clear that the software is an early release and will contain errors. Beta testing, because it is done by people who will actually be using the software, can spot problems and bugs which have previously gone unnoticed.

Commercial software companies may release beta versions of their software to selected users such as computer journalists or current users of previous versions of the software. These individuals receive the software free or at reduced cost, and earlier than ordinary members of the public in return for recording any bugs they might find. The company gains valuable information on bugs which they can fix before the final release.

4.1.5 Learning points

Summary

- Testing of any sort should be accompanied by documentation.
- Integrative testing checks to make sure that individual modules work together as planned.
- End user (beta) testing is done by users who were not involved at the implementation stage and who are therefore less likely to show unintentional bias in favour of the application.
- Usability testing is concerned with the quality and appropriateness of the user interface.
- Accessibility testing is concerned with how well an application and its user interface caters for users with disabilities.
- Programmers / developers will make use of stubs and drivers if the code they are testing is part of an incomplete project, for example during component testing.

4.2 Evaluation

Software and software based systems can be evaluated on a variety of factors, such as functionality, reliability, usability, efficiency, maintainability and others. Quite often these were only done when the system was complete but now they are given more importance and used during the design and development stages.

4.2.1 Fitness for purpose

In Higher Computing Science we evaluated whether a website was fit for purpose if it met the following criteria:

- Has the intended purpose of the website been met?
- What were the requirements of the website and have they been met?
    - User requirements:
        - Can the users find what they are looking for?
        - Can the user access all pages of the site?
        - Do all links (internal & external) work?
    - Functional requirements:
        - Do all planned sections of the page appear in the correct place?
        - Does the navigational bar appear?
        - Are all style rules applied?
        - Do all scripts used in the page work?

We would apply the same criteria now but will also take into account the extra features our websites now use, such as:

- Media queries.
- Use of forms.
- Session variables for the site.
- PHP/integrations with other software applications (programs/databases).

4.2.2 Maintainability

The maintainability of a website can be evaluated in the same way you would a software project. A well-designed website will make good use of external stylesheets and scripts which makes the basic page HTML easier to view.

Any new websites being created should make use of the semantic elements provided by HTML 5 and contain internal commentary to help document the website's development .

Semantic elements

Good use of semantic elements introduced in HTML 5 help to reduce the number of IDs and classes that a web developer needs to create in order to produce complex sites. This in turn simplifies the stylesheets that the web site would need to use. The semantic elements can be thought of as being similar to meaningful variable names in software development.

For example if a <footer> element is being used you would expect this to appear at the bottom of your webpage and a navigation <nav> element would likely store the hyperlinks for the webpage.

Internal commentary

As with any software you create, your website's code should make use of internal commentary in all the documents it uses, such as:

- HTML pages use a block style comment, this allows comments to use multiple lines.

```
<!-- Comments -->
```

or

```
<!--
  Comments
-->
```

- Cascading Stylesheets use a block style comment system as well.

```
/*
  Comments
*/
```

- JavaScript can create single line or block style comments.
  - Single line comments only last until the end of the line.

```
// Comments
```

  - Block style comments, just like HTML / CSS, can span multiple rows.

```
/*
  Comments
*/
```

4.2.3 Robustness and reliability

There is often confusion between the terms robustness and reliability. Software can be evaluated as being reliable and robust.

A program is robust if it can cope with problems that come from outside and are not of its own making (e.g. corrupt input data). Reliability is an internal matter. A program is reliable if it runs well, and is never brought to a halt by a design flaw.

When the program is complicated, the distinction between the two terms is not always clear. When a machine hangs it is not always obvious whether this is due to a failure in robustness or reliability.

Robustness

The design team should try to ensure that the design is robust. The resulting software should be able to cope with:

- mistakes that users might make.
- unexpected conditions that might occur.

These should not lead to wrong results or cause the program to hang.

Examples of an unexpected condition could be:

- something going wrong with a printer (it jams, or it runs out of paper);
- a disc drive not being available for writing, because it simply isn't there (the user's forgotten to put in the flash drive);
- the user entering a number when asked for a letter.

Reliability

A reliable program:

- always produces the expected result when given the expected input;
- is designed correctly to do the task specified.

4.2.4 Efficiency

When a new system or application is released, it should also be efficient. It should not make unreasonable demands on the hardware, databases and other components it needs to run, or the communications between them.

But higher efficiency can lead to less maintainable software if programmers use shortcuts which are effective, but difficult to understand.

When evaluating a release for efficiency, the team will use the acceptance criteria for the project.

4.2.5 Usability

Following functional / operational testing the software can be evaluated against its usability. Usability is a combination of factors including:

- **Intuitive design**:
  A nearly effortless understanding of the architecture and navigation of the site.
- **Ease of learning**:
  How fast a user who has never seen the user interface before can accomplish basic tasks.
- **Efficiency of use**:
  How fast an experienced user can accomplish tasks.
- **Memorability**:
  After visiting the site, can a user remember enough to use it effectively in future visits.

- **Error frequency and severity**:
  How often users make errors while using the system, how serious the errors are, and how users recover from the errors.
- **Subjective satisfaction**:
  If the user likes using the system.

4.3 Summary

Summary

You should now be able to:

- describe, exemplify, and implement the following:
  - integrative testing;
  - usability testing based on prototypes;
  - final testing;
  - end user testing;
- evaluate a software solution in terms of:
  - fitness for purpose;
  - maintainability:
    - perfective;
    - corrective;
    - adaptive;
  - robustness;
  - usability.

4.4 End of topic test

End of topic test Testing and evaluation Go online

Q4: Which documentation would need to be revisited after the testing stage when previously undetected problems are discovered?

a) The source code.
b) The test data used.
c) The requirements specification.
d) The test plan and expected results.

. . . . . .

Q5: Which of these groups of people will not be involved in integrative testing?

a) Developer staff.
b) Programmers.
c) Clients.
d) Beta testers.

a) a and b
b) c and d
c) b and c
d) a, b and d
e) c only

. . . . . .

Q6: What type of testing is performed on the finished product prior to acceptance by the client?

a) Alpha testing.
b) Beta testing.

. . . . . .

Q7: Evaluating a website on it's ease of use, intuitive design and memorability would be looking at?

a) Robustness.
b) Reliability.
c) Efficiency.
d) Usability.

. . . . . .

Q8: Which of these will help to aid maintainability when used in an HTML document?

a) Internal stylesheets.
b) Internal commentary.
c) Using classic HTML tags, IDs and classes.
d) Reduce the number of IDs and classes by using semantic elements.

a) a and b
b) c and d
c) b and c
d) b, c and d
e) All of them

. . . . . .

Q9: David has created a new website for his martial arts club which is fully HTML 5 compliant through the use of semantic elements. Members can log in and JavaScript provides interactivity. Some users have complained that it is too slow and often crashes while using it.

What type of evaluation should have detected these problems?

a) Maintainability.
b) Usability.
c) Fitness for purpose.
d) Efficiency.

a) a and c
b) b and d
c) a, b and d
d) b, c, and d
e) All of them

Topic 5

Web design and development test

Web design and development test Go online

Q1: A doctor's surgery has created a new online booking system for patients needing an appointment. Once the user has successfully logged in and completed the request form with their personal information, the details are sent to a server-side script called bookings.php. Which is the best option to use?

a) <form action = "bookings.php" method = "get">
b) <form action = "bookings.php" method = "post">

. . . . . .

Q2: In the doctor's appointment system there are two types of user of the system, the *patient* and the *reception staff*. Each day the reception staff will check the booking requests and then confirm the date/time of each booking in the system. Which user is the primary actor and which is the secondary actor?

. . . . . .

Q3: What kind of use case relationship is indicated by a dashed line and an arrow pointing towards the Dependent Use Case?

a) Association.
b) Include.
c) Extend.
d) Generalisation.

. . . . . .

Q4: What kind of variables can be used to store values across multiple web pages?

a) Global.
b) Temporary.
c) Local.
d) Session.

. . . . . .

Q5: What character is used at the start and end of a PHP script?

a) !
b) $
c) ?
d) *

. . . . . .

Q6: Complete the missing form values in this script.

```
<form _______ = "update.php" method = "_______">
  Username: <input type = "text" name = "username">
  <br>
  New Password: <input type = "password" name = "password1">
  <br>
  Re-enter Password: <input type = "_______" name = "password2">
  <br>
  <input type= "_______" value="Update">
</form>
```

.....

Q7: When processing an HTML, form the POST method is slightly more secure than the GET method because it:

a) encrypts the data sent to the server.
b) sends the data via an HTTP tunnel rather than within the URL.
c) sends the data via HTTP headers rather than within the URL.
d) routes data through a number of servers for security.

.....

Q8: The *action* attribute of an HTML form element specifies:

a) the script which will process the data.
b) the name of the form.
c) the CSS class of the form.
d) a JavaScript action for the form.

.....

Q9: Code injection is an attack which:

a) acts as a 'man-in-the-middle' collecting data between client and server.
b) uses a faulty SSL certificate to allow attacks on the server.
c) exploits poorly secured form variables to execute code on the server.
d) uses CSS selectors to execute JavaScript on the client.

.....

Q10: Which one of the following techniques is used to represent the motivations, goals and frustrations of users so that the development team can understand their needs?

a) Wireframe.
b) Use case.
c) Persona.
d) User scenario.

.....

Q11: Complete the PHP statement so it can connect to the MySQL database within the script.

```
<?php
  //set up the connection variables
  $server="sports";
  $username="USERname";
  $password="passWORD";
  $database="esports";

  $mysqli = ________ (_________) _________
?>
```

.....

Q12: Complete the PHP statement to execute a query (*$query*) stored in a variable *$mysqli*.

$result = ________->________(________);

Glossary

***AMP**

a combination of an operating system, Apache web sever, MySQL and PHP commonly used to provide web services.

Acceptance criteria

the agreed critera which have to be met before the product can be handed over to the client.

Actor

Individuals who interact with a system in a Use Case Diagram or a Sequence Diagram.

Agile

an approach to software development that is flexible and focuses on working software over documentation and complete sets of requirements.

Alpha testing

tests performed by the programming team during the implementation phase.

Beta testing

tests performed by a group of clients or users prior to accepting the software.

Budget

the amount of money available to complete a project.

Client

usually a web browser or app that has requested some information via the internet

Client-server model

the client-server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients

Code injection

the exploitation of a computer bug that is caused by processing invalid data. Injection is used by an attacker to introduce (or "inject") code into a vulnerable computer program and change the course of execution.

Constraints

the variables that limit the activities of the project: scope, time, cost and quality.

Cost

the financial value of a resource used to deliver part of a project (may also apply to the whole project).

CSS

Cascading Style Sheets, a style sheet language used to describe the look and formatting of a document written in a markup language. It is currently on version 3 of the CSS standard (CSS3).

Driver

a piece of code designed to test a function or procedure in an incomplete program.

GET method

a means of passing data to a script via the URL.

HTML

Hypertext Markup Language, a standardized system for tagging text most commonly used to define the structure and content of a document.

JavaScript

an object-oriented computer programming language commonly used to create interactive effects within web browsers.

Milestones

a significant stage or event in the development of a project.

Objectives

what the project is intended to achieve.

PHP

a scripting language generally used in website development and can be embedded in to HTML.

Primary Key

is a single column or combination of columns that uniquely identifies each row in a table.

Product backlog

is the single most important artifact in the Scrum approach to agile development. The product backlog is, in essence, an incredibly detailed analysis document, which outlines every requirement for a system, project, or product.

Project manager

the person in overall charge of the planning and execution of a particular project.

Quality

the standard of something as measured against other things of a similar kind

Regular expression

a sequence of symbols and characters which define a string or pattern to be searched for within a longer piece of text.

Scope

the detail of what a project is to achieve. Changes to the scope of the project have a direct impact on the time and cost of the project.

Screen reader

software used by users who are blind or have impaired vision which reads the test on a web page or the interface of a program.

SCRUM

is an iterative and incremental agile software development framework for managing product development.

Server

a computer or computer program which manages access to a centralised resource or service in a network (e.g in an office)

Skeuomorphism

when an interface mimics the hardware which an application is replacing.

Sprint

a period of development when a fixed set of product items (the sprint backlog) is developed.

Stub

a short section of code which stands in for a missing sub-program when testing a partially completed program.

Timescale

a period of time to complete an activity.

URL

Uniform Resource Locator, a reference (an address) to a resource on the Internet. A URL has two main components: Protocol identifier: For the URL http://scholar.hw.ac.uk/, the protocol identifier is http . Resource name: For the URL http://scholar.hw.ac.uk/ , the resource name is scholar.hw.ac.uk/ - the resource can include the server, port, path and filename of the resource plus any internal anchor and/or data.

Use cases

is an approach used in system analysis to identify, clarify, and organize system requirements. Use cases are made up of sets of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

User persona

is a representation of the goals and behavior of a hypothesized group of users. In most cases, personas are created from data collected from interviews with users.

User scenarios

describe the stories and context behind why a specific user or user group comes to your site or use your application. They note the goals and questions to be achieved and sometimes define the possibilities of how the user(s) can achieve them on the site.

User stories

are short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

Waterfall model

is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance.

Web server

a computer system that processes requests via HTTP, the basic network protocol used to distribute information on the internet

XSS

Cross-site scripting, a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side script into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy.

Answers to questions and activities

Topic 1: Analysis

Activity: Types of feasibility (page 12)

Q1:

A project that cannot be delivered in the time available because of the complexity of the task	has an issue with schedule feasibility.
A project which has insufficient finance to hire the necessary staff to complete the project when required	has an economic feasibility issue.
A project that processes banking transactions to indicate spending habits in contravention of data protection laws	has an issue with legal feasibility.
A project which cannot proceed because the software to link two data sources into the project is not available	has an issue with technical feasibility.

Review of Use Case Diagrams (page 19)

Q2:

Use Case diagrams are made using **Actors** that will interact with the system. The system is indicated by a **system boundary** where all the use cases will be placed. A Use Case in the diagram will be represented by an **ellipse**.

There are 5 relationships we need to use in a Use Case Diagram:

Association

Include — this is a use case that is always **actioned** but not directly by the Actor.

Extend — This Use Case is not directly accessed by the Actor and doesn't always run.

Generalisation of an **Actor**

Generalisation of a Use Case

End of topic test: Analysis (page 19)

Q3: a) A person or entity that interacts with the system.

Q4: c) how actors interact with the system.

Q5: c) economic and technical

Q6:

- Time
- Scope
- Legal
- Cost

Q7:

- End User Requirements
- Scope
- Boundaries
- Constraints

Topic 2: Design

Wireframe design (page 25)

Example answer

1. Body
   Background colour - Purple
   Font - Arial
   Text colour - black
2. Header
   Height - 120px
3. Main heading <h1>
   Text size - 20px
4. Sub heading <h3>
   Text size - 14px
5. Navigation bar
   Height - 50px
6. Navigation link
   Width - 100px
   Height - 50px
   Mouseover: Background color - Orange
   Mouseout: Default
7. Content
8. Text <p>
   Text size - 12px

End of topic test: Design (page 35)

Q1: b) False

Q2: b) Wireframe.

Q3: a) version of the application drawn on paper with which users can interact.

Q4: a) a web server and a web browser making requests.

Topic 3: Implementation

Ten green bottles (page 58)

Expected answer

```php
<?php
  $count = 10;

  echo "<h1>Green Bottles!</h1>";
  echo "<br />";
  echo "<br />";

  while ($count > 0) {
    echo "<div>"

    echo $count . " Green bottles standing on the wall,";
    echo "<br>";
    echo $count . " Green bottles standing on the wall,";
    echo "<br>";
    echo "and if one green bottle should accidentally fall,";
    echo "<br>";

    $count--;
    if ( $count === 0 ){
      echo "There'll be no green bottles standing on the wall.\n";
    } else {
      echo "There'll be $count green bottles standing on the wall.\n";
    }

    echo "</div>";

  }
?>
```

Quiz: Revision (page 63)

Q1: c) process data at the server side.

Q2: a) nearly all client-side validation can be circumvented.

Q3: c) it validates data before it leaves the browser reducing the number of resubmissions for errors.

Q4: d) pulls site data from a database to create site pages.

Q5: c) server-side language module.

Player table (page 80)

Expected answer

To display player:

```
<?php
  $host="localhost";
  $username="root";
  $pword="root";
  $database="esports";

  require("db_connection.php");
?>

<html>
  <head>
    <title>Database Query Player</title>
    <style>
      table, td, th {border: 1px solid black;}
    </style>
  </head>
  <body>
    <?php
      //set up a query to use
      $query_string = "SELECT * FROM player";
    ?>

    <table>
      <thead>
        <th class="outline">Username</th> <th>Realname</th>
           <th>Password</th>
        <th>Email</th> <th>Message</th> <th>Terms and Conditions</th>
      </thead>
      <tbody>
        <?php

          if ($result = $mysqli->query($query_string)) {  //run the
             query

            while ($row = $result->fetch_object()) {
              echo "<tr>";
              echo "<td>" . $row->username . "</td>";
              echo "<td>" .  $row->realname . "</td>";
              echo "<td>" .  $row->pword . "</td>";
              echo "<td>" .  $row->email . "</td>";
              echo "<td>" .  mb_convert_encoding($row->message,
                'utf-8', 'iso-8859-1') . "</td>";
              echo "<td>" .  $row->terms_and_conditions . "</td>";
              echo "<td><a href=edit-data-player.php?username=" .
                urlencode($row->username) . ">Edit</a></td>";
              echo "<td><a href=delete-data-player.php?username=" .
                urlencode($row->username) . ">Delete</a></td>";
              echo "</tr>";
            }

            $result->close();  //free the $result set (clear it)
          }
          $mysqli->close();
        ?>
      </tbody>
```

```
    </table>

    <a href="add-player-row.php">Insert a new row</a>

  </body>
</html>
```

To edit the player values:

```
<?php
  $host="localhost";
  $username="root";
  $pword="root";
  $database="esports";

  require("db_connection.php");
  $username = filter_var($_GET['username']);
?>

<html>
  <head>
    <title>Database Update</title>
    <link rel="stylesheet" media="screen" href="styles.css">
  </head>
  <body>
    <?php
      //set up a query to use
      $query_string = "SELECT * FROM player WHERE username = '".
          $username . "'" ;

      if ($result = $mysqli->query($query_string)) {  //run the query
        while ($row = $result->fetch_object()) {
    ?>

    <form class="signup_form" action="update-row-player.php"
       method="post" name="signup_form">

      <!-- keep track of the original primary key value to update the
          row -->
      <input type="hidden" name="usernameid" value="<?php echo
          urlencode($row->username); ?>">

      <!-- collection of form associated elements goes here -->
      <ul>
      <li>
        <h2>Edit Data</h2>
        <span class="required_notification">* Denotes Required
            Field</span>
      </li>

      <!-- start of user ID sign up fields -->
      <li>
        <label for="username">Username:</label>
        <input id="username" type="text" name="username"
        value="<?php echo $row->username;?>" size="40" maxlength="40"
        placeholder="jensmith72" pattern="^[A-Za-z0-9]+$" required />
```

```
43          <span class="form_hint">A combination of letters and numbers
               only.</span>
44        </li>
45        <li>
46          <label for="realname">Real Name:</label>
47          <input id="realname" type="text" name="realname"
48          value="<?php echo $row->realname;?>" size="40" maxlength="40"
49          placeholder="Jenny Smith" required />
50          <span class="form_hint">Please enter your firstname and
               lastname.</span>
51        </li>
52        <li>
53          <label for="email">Password:</label>
54          <input id="pword" type="password" name="pword"
55          value="<?php echo $row->pword;?>" size="40" maxlength="60"
56          placeholder="**********" required />
57          <span class="form_hint">Please enter a password</span>
58        </li>
59        <li>
60          <label for="email">Email:</label>
61          <input id="email" type="email" name="email"
62          value="<?php echo $row->email;?>" size="40" maxlength="60"
63          placeholder="jenny.smith@example.com" required />
64          <span class="form_hint">Use the format "name@domain.com"</span>
65        </li>
66          <!-- end of user ID sign up fields -->
67
68        <!-- start of additional fields -->
69        <li>
70          <label for="message">Message:</label>
71          <textarea id="message" name="message" cols="40" rows="6" >
72            <?php echo $row->message; ?>
73          </textarea>
74          <span class="form_hint">A brief message, why you want to sign
               up.</span>
75        </li>
76        <li>
77          <label for="terms_and_conditions">Agree to terms and
               conditions</label>
78          <?php
79            if ($row->terms_and_conditions == '1') {
80          ?>
81          <input id="terms_and_conditions" type="checkbox" value="1"
82          checked name="terms_and_conditions">
83          <?php
84            } else {
85          ?>
86          <input id="terms_and_conditions" type="checkbox" value="1"
87          unchecked name="terms_and_conditions">
88          <?php
89            }
90          ?>
91        </li>
92        <li>
93          <button class="submit" type="submit">Submit Form</button>
94        </li>
```

```
      <!-- end of additional fields -->
      </ul>
    </form>

    <?php
        }
        $result->close();  //free the $result set (clear it)
      } else {
        echo "Error in Game Table Query";
      }
      $mysqli->close();
    ?>
  </body>
</html>
```

To update the row:

```
<?php
  $host="localhost";
  $username="root";
  $pword="root";
  $database="esports";

  require("db_connection.php");
  $usernameid = filter_var($_POST['usernameid']);
  $username = filter_var($_POST['username']);
  $realname = filter_var($_POST['realname']);
  $pword = filter_var($_POST['pword']);
  $email = filter_var($_POST['email']);
  $message = filter_var($_POST['message']);
  $terms_and_conditions = filter_var($_POST['terms_and_conditions']);
?>

<html>
  <head>
    <title>Database Update</title>
  </head>
  <body>
    <?php

      //set up a query to use
      $query_string = "UPDATE player SET
        username = '". $username . "',
        realname = '" . $realname . "',
        pword = '" . $pword . "',
        email = '" . $email . "',
        message = '" . $message . "',
        terms_and_conditions = '" . $terms_and_conditions . "'
        WHERE player.username = '". $usernameid . "'" ;

      if ($result = $mysqli->query($query_string)) {  //run the query
        //show successful output
        echo "Completed: Updated Row in Table";
        //navigation back to records
        echo "<a href=database4.php />Back</a>";
      } else {
```

```
            //show error because it's failed
            echo "Error: Unable to Update Row in Table <br /><br />";
            //display a list of the MySQL errors that occurred
            print_r($mysqli->error_list);
            //navigation back to records
            echo "<a href=database4.php />Back</a>";
        }
        $result->close();  //free the $result set (clear it)
        $mysqli->close();
    ?>
    </body>
</html>
```

To insert player form:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sign-up form</title>
    <link rel="stylesheet" media="screen" href="styles.css">
  </head>
  <body>
    <!-- start form for signup -->
    <form class="signup_form" action="insert-row-player.php"
       method="get" name="signup_form">

    <!-- collection of form associated elements goes here -->
    <ul>
      <li>
        <h2>Sign Up</h2>
        <span class="required_notification">* Denotes Required
           Field</span>
      </li>

      <!-- start of user ID sign up fields -->
      <li>
        <label for="username">Username:</label>
        <input id="username" type="text" name="username" value=""
        size="40" maxlength="40" placeholder="jensmith72"
        pattern="^[A-Za-z0-9]+$" required />
        <span class="form_hint">A combination of letters and numbers
           only.</span>
      </li>
      <li>
        <label for="realname">Real Name:</label>
        <input id="realname" type="text" name="realname" value=""
        size="40" maxlength="40" placeholder="Jenny Smith" required />
        <span class="form_hint">Please enter your firstname and
           lastname.</span>
      </li>
      <li>
        <label for="email">Password:</label>
        <input id="pword" type="password" name="pword" value=""
        size="40" maxlength="60" placeholder="**********" required />
        <span class="form_hint">Please enter a password</span>
```

```
      </li>
      <li>
        <label for="email">Email:</label>
        <input id="email" type="email" name="email" value="" size="40"
        maxlength="60" placeholder="jenny.smith@example.com" required />
        <span class="form_hint">Use the format "name@domain.com"</span>
      </li>
      <!-- end of user ID sign up fields -->

      <!-- start of additional fields -->
      <li>
        <label for="message">Message:</label>
        <textarea id="message" name="message" cols="40"
           rows="6"></textarea>
        <span class="form_hint">A brief message, why you want to sign
           up.</span>
      </li>
      <li>
        <label for="terms_and_conditions">Agree to terms and
           conditions</label>
        <input id="terms_and_conditions" type="checkbox"
        name="terms_and_conditions">
      </li>
      <li>
        <button class="submit" type="submit">Submit Form</button>
      </li>
      <!-- end of additional fields -->
    </ul>
    </form>
  </body>
</html>
```

To insert player row:

```
<?php
  $host="localhost";
  $username="root";
  $pword="root";
  $database="esports";

  require("db_connection.php");
  $player = filter_var($_GET['player']);
  $realname = filter_var($_GET['realname']);
  $pword = filter_var($_GET['pword']);
  $email = filter_var($_GET['email']);
  $message = filter_var($_GET['message']);
  //store the correct value for terms and conditions
  $terms_and_conditions = filter_var($_GET['terms_and_conditions']);

  if ($terms_and_conditions == "on") {
    $terms_and_conditions = 1;
  } else {
    $terms_and_conditions = 0;
  }
?>

```

```
<html>
  <head>
    <title>Database Insert</title>
  </head>
  <body>
    <?php
      //set up a query to use
      $query_string = "INSERT INTO player VALUES (
          '" . $username . "',
          '" . $realname . "',
          '" . $pword . "',
          '" . $email . "',
          '" . $message . "',
          '" . $terms_and_conditions . "'
          )";

      if ($result = $mysqli->query($query_string)) {  //run the query
        echo "Completed: Inserted Row in Table";  //show successful
           output
        echo "<a href=database4.php />Back</a>"; //navigation back to
           records
      } else {
        //show error because it's failed
        echo "Error: Unable to Insert Row in Table<br />";
        //display a list of the MySQL errors that occurred
        print_r($mysqli->error_list);
        //navigation back to records
        echo "<br /><a href=database4.php />Back</a>";
      }
      $result->close();  //free the $result set (clear it)
      $mysqli->close();
    ?>
  </body>
</html>
```

To delete player row:

```
<?php
  $host="localhost";
  $username="root";
  $pword="root";
  $database="esports";

  require("db_connection.php");
  $game = filter_var($_GET['username']);
?>

<html>
  <head>
    <title>Database Delete Player</title>
  </head>
  <body>
    <?php
      //set up a query to use
      $query_string = "DELETE FROM player WHERE username = '".
      $username . "'" ;
```

```
20
21        if ($result = $mysqli->query($query_string)) {  //run the query
22          echo "Success, the row for player=" . $username .
23          " has been deleted.<br />";
24          //navigation back to records
25          echo "<br /><a href=database4.php />Back</a>";
26        } else {
27          //show error because it's failed
28          echo "Error: Unable to Delete Row in Table<br />";
29          //display a list of the MySQL errors that occurred
30          print_r($mysqli->error_list);
31          //navigation back to records
32          echo "<br /><a href=database4.php />Back</a>";
33        }
34        $result->close();  //free the $result set (clear it)
35        $mysqli->close();
36      ?>
37    </body>
38  </html>
```

End of topic test: Implementation (page 103)

Q6: c) Viewport

Q7: b) False

Q8: b) displays output.

Q9: POST

Q10: Username, server, password, database.

Q11:

```
1  require("db_connect.php");
```

To import the file "db_connect.php" you can use either `require` or `include`, but only `require` returns a fatal error.

Q12: `echo "<h1>Welcome to Scholaria " . $User . "</h1>";`

Q13: a) `$mysqli->query($query);`

Q14: c) filter_var

Topic 4: Testing and evaluation

Answers from page 110.

Q1: a) True

Q2: b) False

Q3: b) False

End of topic test Testing and evaluation (page 116)

Q4: b) The test data used.

Q5: c) b and c

Q6: b) Beta testing.

Q7: d) Usability.

Q8: c) b and c

Q9: b) b and d

Topic 5: Web design and development test

Web design and development test (page 120)

Q1: b) <form action = "bookings.php" method = "post">

Q2:

Primary: patient.

Secondary: reception staff.

Q3: b) Include.

Q4: d) Session.

Q5: c) ?

Q6:

```
<form action = "update.php" method = "post">
  Username: <input type = "text" name = "username">
  <br>
  New Password: <input type = "password" name = "password1">
  <br>
  Re-enter Password: <input type = "password" name = "password2">
  <br>
  <input type= "submit" value="Update">
</form>
```

Q7: c) sends the data via HTTP headers rather than within the URL.

Q8: a) the script which will process the data.

Q9: c) exploits poorly secured form variables to execute code on the server.

Q10: c) Persona.

Q11:

```
<?php
  //set up the connection variables
  $server="sports";
  $username="USERname";
  $password="passWORD";
  $database="esports";

  $mysqli = new mysqli ("sports", "USERname", "passWORD", "esports") ;
?>
```

Q12: $result = $mysqli->query($query);