ASSIGNMENT 2 SPECIFICATION

Introduction

The aim of this assignment is to complete the implementation the “What happened When” (WHW) application.

The completed application will provide a "front end" to a database of information about the events covered by WHW - details of the structure of the database are given below. Users will be able to interact with the database at two levels:

- "Public" – for users who have not "signed up" for membership in WHW
- "Members" – for users who have "signed up" for membership in WHW and have "signed in" for a session

The interactions required for each of these groups and all of the features that the system should have are detailed below (see SYSTEM DESCRIPTION).

Due date and submission method

Due date: Wednesday 26 May 3pm (week 13 of semester)

NOTE: Submission will *not* be through MyLO, it will be made using the School of Computing & Information Systems submission system (see below for details of the submission method).

Late Submissions

- Late assignments will only be accepted in exceptional circumstances and provided that the proper procedures have been followed (see the School Office or the CIS web site for details). Assignments that are submitted late without good reason will be subject to mark penalties if they are accepted at all (see School office or web site for details on this as well).
- Forms to request extensions of time to submit assignments are available from the School of Computing office. Requests must be accompanied by suitable documentation and should be submitted before the assignment due date.

Marking

This assignment carries 20% of the marks for this unit. Remember that you must obtain at least 45% of the available marks for the in-semester assessment component to be able to obtain a passing grade in this unit.

Your submitted work for this assignment will be assessed against criteria that are related to the learning outcomes for this unit.

Here is a table that indicates the criteria, their weightings, and the associated learning outcomes for the unit. Details of the performance standards for each of the criteria are given below (see Marking Information). Your work will be judged against these performance standards.

Criterion	Weighting	Related unit learning outcome(s) (Numbers are as shown in unit outline)
Makes safe use of information from the client.	**Essential** – if you do not match the required performance standard for this criterion you will receive a *failing* mark for this assignment.	**2, 3, 4**
Follows acceptable programming practice (as defined in this specification and other unit materials)	**Important** – if you do not match the required performance standards for this criterion your mark for this assignment will be reduced by between 1 and 2 grade levels	**2**
Correct submission of work as specified	**2%**	**2**
Overall structure of application is as required	**5%**	**3**
Information about authorised users is maintained in a suitable database table.	**5%**	**2,3**
Authorinfo page contains a thoughtful and complete answer to the question posed.	**5%**	**2,3,4**
"Extras" section of every page contains the appropriate information.	**10%**	**2,3,4**
The "outside information" is correctly placed on the required pages. NOTE: The nature of the "outside information" required varies depending on the unit of enrolment and / or the campus of enrolment.	**8%**	**2,3** **5 (KXT309)**
System allows all users (public and members) to search the data collection.	**5%**	**2,3,4**

System displays the information about an event in a acceptable manner – including an indication of the tags that have been attached to this event where there are links to allow all users (public and members) see all other events that have been tagged with each of these tags.	**15%**	**2,3**
System allows *only* members, who are currently signed in, to add tags to events.	**15%**	**2,3,4**
Users are able to "sign up" to become members. The "sign up" operation functions as specified.	**10%**	**2,3**
"Sign in" and "sign out" operations (for members) operate correctly. (This means that an authenticated session is set up when the member signs in and comes to an end when they sign out.)	**10%**	**2,3,4**
Code is well written	**10%**	**2,3,4**

System Description

COMPONENTS OF THE SYSTEM

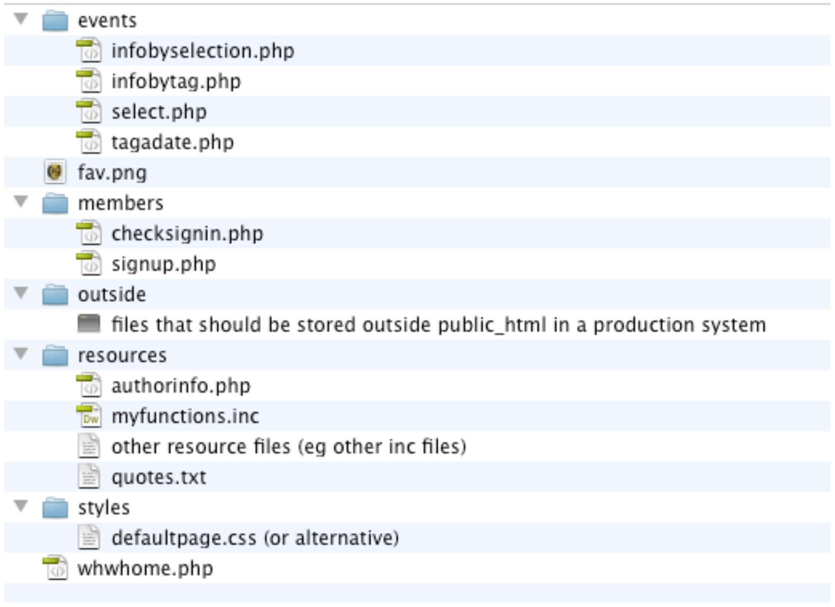
The completed WHW system (as required for this assignment) will have the following components (sections). See below for the detailed requirements for each of these sections

- The home page
- the author information page
- The "view events" section – there will be differences for the way in which this behaves for "the public" and "members"
- The members sign up section
- A database table to keep information about members
    - The structure of this table should be set up to have the fields to hold the following information:
        - the username
        - the password (in "encoded" form)
        - date of birth (optional)
        - the number of tags that this user has successfully added

- When the assignment is submitted this database table must contain (at least) information about a user with the following details:
  - username: jotomr
  - password: M&rker10
  - date of birth "empty"
  - number of tags successfully added - 0

File organisation

The files that you produce and submit for the programming part of this assignment should be organised in the following way.



```
events
    infobyselection.php
    infobytag.php
    select.php
    tagadate.php
fav.png
members
    checksignin.php
    signup.php
outside
    files that should be stored outside public_html in a production system
resources
    authorinfo.php
    myfunctions.inc
    other resource files (eg other inc files)
    quotes.txt
styles
    defaultpage.css (or alternative)
whwhome.php
```

NOTE:

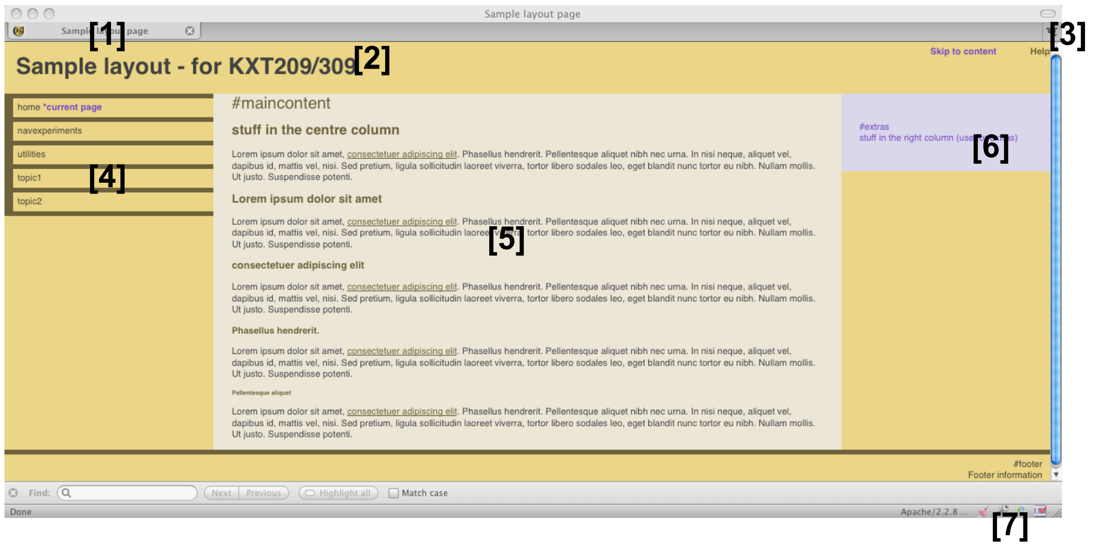
- These files should be in another folder labelled with your username.
- You will be supplied with the file `quotes.txt` (it can be downloaded from MyLO or `lawson` / `alacritas`). (Details about the contents below.) NOTE: This will only be necessary for students enrolled in the unit KXT209.)
- You are *not* required to use the files `fav.png` or `defaultpage.css` (as supplied in self studies and tutorials). If you choose *not* to use `defaultpage.css` you should make sure that the layout of your pages can be followed easily, and that it does not use deprecated elements (eg `<font>`) or attributes (eg `align`).

Page layout

Each page of the application that you submit should contain the following components – these are described in terms of the parts of a page that uses `defaultpage.css` for layout.

- The title [1] of the page and the "banner" text [2] should be appropriate for the page.
- The "utilities" links [3] (in the top right hand corner) should have links labelled
    - “Skip to content” – that links to the beginning of the maincontent `<div>` of that page
    - “Author information” – that links to the `authorinfo.php` page. This is the only page that does not need to have all the components listed here, it should contain the following information about the author (you).
        - Full name
        - Username
        - id number
        - Campus (Launceston or Hobart)
        - A list of resources you used in preparing this assignment.
        - A list of known problems that you have identified when testing your implementation and have not been able to fix.
        - Your written response to the question asked below.
- The navigation bar [4] in the *public* section (that is for users who *have not* logged on) should contain entries for:
    - home – that links to `whwhome.php`
    - events – that links to `select.php` in the `events` directory
    - member signup – that links to `signup.php`
- The navigation bar [4] in the *members* section (that is for users who *have* logged on) should contain entries for:
    - home – that links to `whwhome.php`
    - events – that links to `select.php` in the `events` directory
- The `maincontent` section [5] should contain the relevant information for each page (details below) If you have not implemented any section there should be still be a page of the correct name with a message in the `maincontent` section stating that this feature has not been implemented.
- The contents of the extras section [6] depends (in part) on whether or not the user has logged in (and on which page we are considering):
    - If the user has not logged in - there should be a "login" section, that allows the user to enter the username and password that they selected when they "signed up". Successful login in here should create an authenticated session where the user has access to the "members" features of the application.
    - If the user is currently logged in - there should be a "logout" button that will end the authenticated session.
    - In the pages `select.php`, `infobyselection.php`, `infobytag.php` the extras section should also contain a "tag cloud" that shows each of the tags that has been used, where the size of the word indicates the number of dates that have been tagged with this tag. Each word shown in the cloud should be a link that takes the user to the page infobytag.php which shows all of the events that have that tag. (See SS9 for more information.)

- The footer section [7] should contain the following information:
    - The "outside information" required – the nature of this depends on the exact unit you are enrolled in and / or your campus location (see below for details).
    - The name of the author
    - The date when the file was last modified.



Sample layout - for KXT209/309

#maincontent

stuff in the centre column

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus hendrerit. Pellentesque aliquet nibh nec urna. In nisi neque, aliquet vel, dapibus id, mattis vel, nisi. Sed pretium, ligula sollicitudin laoreet viverra, tortor libero sodales leo, eget blandit nunc tortor eu nibh. Nullam mollis. Ut justo. Suspendisse potenti.

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus hendrerit. Pellentesque aliquet nibh nec urna. In nisi neque, aliquet vel, dapibus id, mattis vel, nisi. Sed pretium, ligula sollicitudin laoreet viverra, tortor libero sodales leo, eget blandit nunc tortor eu nibh. Nullam mollis. Ut justo. Suspendisse potenti.

consectetuer adipiscing elit

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus hendrerit. Pellentesque aliquet nibh nec urna. In nisi neque, aliquet vel, dapibus id, mattis vel, nisi. Sed pretium, ligula sollicitudin laoreet viverra, tortor libero sodales leo, eget blandit nunc tortor eu nibh. Nullam mollis. Ut justo. Suspendisse potenti.

Phasellus hendrerit.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus hendrerit. Pellentesque aliquet nibh nec urna. In nisi neque, aliquet vel, dapibus id, mattis vel, nisi. Sed pretium, ligula sollicitudin laoreet viverra, tortor libero sodales leo, eget blandit nunc tortor eu nibh. Nullam mollis. Ut justo. Suspendisse potenti.

Pellentesque aliquet

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus hendrerit. Pellentesque aliquet nibh nec urna. In nisi neque, aliquet vel, dapibus id, mattis vel, nisi. Sed pretium, ligula sollicitudin laoreet viverra, tortor libero sodales leo, eget blandit nunc tortor eu nibh. Nullam mollis. Ut justo. Suspendisse potenti.

Detailed specifications for each page / section

The home page (`whwhome.php`)

The main content section of this page should have a brief introduction to the application.

The events section

When a user (public or member) chooses "events" from the menu they should be taken to the `select.php` page – on this page they will be given the chance of entering selection criteria to chose the events that they are to view. (See below for details of the selection criteria.)

The form(s) on `select.php` should be processed by `infobyselection.php` which should display a list of all of the events that match the search criteria submitted. This list should show the following information for each event (that matches the criteria). If more than one event matches the criteria, they should be shown ordered by year (earliest year first).

- The year.
- The short description of the event.

- The details of the event.
- A list of the tags that have been added to this event. Each one of these should be a link to the page `infobytag.php` – following that link will show the information about all of the events that have been labelled with that tag.
- For members who are logged in ONLY. A link labelled "tag this event" – which takes them to the file `tagadate.php` that allows them to specify the tag they want to add to the event. If the event has *not already* been tagged with that tag, then the tag will be added, if it *has* already been tagged the member will be thanked for their interest and told that that event is already tagged with that tag.

NOTE: In many cases the list of events will be quite long – you should consider implementing some form of pagination on the `infobytag`, and `infobyselection` pages.

The search criteria:

Users should be able to search for events by one or more of the following: You are expected to implement at least TWO (2) search criteria – you will receive credit it you successfully implement more (up to a total of 4) – here are some suggestion of search criteria you might use (if you implement more than 2 criteria you are expected to think some up for yourself.

- year (exact) – the user should be given a drop down list of all possible dates to select from
- year (fuzzy) the user should be given a text box in which to enter a year (or part of a year) to search for.
- words that might appear in the short description – the user should be given a text box in which to enter a word (or words) to search for in the shortdesc field.

NOTE:

1. The system should function correctly even if the user does not enter any search criteria. In this case, all events would be selected.
2. Where appropriate, the application should do a "fuzzy search" and come up with a list of all of the possible events that could match the selection criteria. (This should be listed in ascending order of the year.)
3. Remember that you need to deal with the case where there are no matching events.

Viewing events by tag

This section should be implemented by the `infobytag.php` script. The information about which tag should be included in the query string and the page should display the year, shortdesc, and details for all of the events that have been tagged with this tag. The events should be listed in ascending order of year.

Tagging a date

Remember this option is only available to members who are currently logged on. If a member clicks the "tag this event" link associated with an event (displayed on the `infobyselection` page) they should be taken to the `tagadate.php` page.

The steps for tagging a date are as follows: (Remember to think carefully about how to do this.)

- The information about which event is to be tagged should be "known" (consider how to store this information).

- The member selects the tag they want. (Consider how to store this information.)
- Attempt to insert this information in to the KXT209WHW database.
- Check and see what happened (remember the insert will fail if that event has already been tagged with that tag) and give the member the appropriate feedback.
- "Return" the user to the home page.

Member sign in

A member signs in by correctly entering a username and password in the "sign in" form in the extras section. These entries should be processed by the file `checksignin.php`. If the username and password match values that are stored in the database table you have set up for this purpose then an authenticated session should begin, show the user a simple "successful log in" message and redirect them to the main home page. If the username and password do not match, show the user a simple "log in failed" message and redirect them to the main home page.

Remember that once a member is logged in they must see a "log out" option in the extras section of every page they visit.

The member sign up page (`signup.php`)

This page should contain a form that a user can fill in to "sign up" as a member of WHW. Each field of the form should be styled to indicate whether it is required and, if applicable, a short description of the type of information to be entered. (The default CSS sheet for the unit has rules that make this relatively straightforward.)

The fields to be filled in are:

- username – This is a required field. To be valid the user must enter a value that contains only letters or digits and is not be a username that has already been taken (a username already in the database table you have set up to hold usernames etc).
- password – This is a required field. To be valid, it must be a value that contains at least 6 characters, which must not include any spaces (all other characters are acceptable).
- password check – This is a required field. To be valid it be a value that is exactly the same entry as the first password entry.
- date of birth – This is an optional field. Users should be given 3 drop down lists to enter this (it must be a valid date) – the options in the drop down lists must make it possible to enter no date.
    - day (1 – 31)
    - month ('jan' - 'dec')
    - year (1910 – 2009)
- An "antispam" CAPTCHA. Use the functions provided by the jpgraph library to create a 5 digit CAPTCHA and store the value in a cookie (called `test`). Give the user a textfield in which to enter the value displayed. When the form is submitted, a valid entry in this field will be one where the value matches the value stored in the cookie.

When the form is submitted it is to be validated (the valid values for each entry are indicated above).

- If any of the entries are invalid, the user will be shown the form again, showing:

  - A message beside invalid entries briefly stating the problem:
  - If the user had entered a valid username (but some other entries were invalid) – the valid username should be shown in that textfield.
  - The date that the user had entered should be shown as the selected options in the drop down lists. This is so even if the date was invalid – it will help the user see what was wrong and fix it.
  - The entries in the password and password check fields are *never* to be retained when the form is shown again.
  - There will be a new CAPTCHA image.
- If all of the entries are valid, the user will:
  - See a message that tells them they are now registered as a member and that they can now log in
  - Be redirected to the WHW home page.

Authorinfo page question to consider

Imagine that you are responsible for the team that is developing an application rather similar to this WHW application. Someone in a senior position in the organisation has suggested the "simple improvement" of allowing members to choose their own tags (rather than having to choose from a fixed list as in this application).

On the authorinfo page you should write a response to this proposal, in approximately one "page" (400 – 600 words). You should consider:

- the advantages and disadvantages of this proposal
- any additional programming that would be required
- any alternative methods you can think of that would give members some flexibility in the choice of tags without the disadvantages you have identified.

"Outside information" required in the footer

- If you are enrolled in the unit **KXT209 (Launceston or Hobart)** the footer of each page should contain a "random" quote – this is to be read from the file `quotes.txt` (this file can be downloaded from MyLO or lawson / alacritas).
- If you are enrolled in the unit **KXT309** the footer of each page should demonstrate a simple server-side "mashup" – you will be given a url that provides a "payload" of information (in the form of XML) that you are to extract and insert into the footer. Self study 11 will contain information about the way in which you can extract the information from an XML payload.
  - **Launceston** students use the url `http://alacritas.cis.edu.au/~rgibson/smashA2/source.php` This provides a random "useless fact" marked up in XML as in the following example:

```
<fact>
<number>3</number>
<text>
In ancient Rome it was considered a sign of leadership to be born with a crooked nose.
</text>
</fact>
```

  - The "outside information" should show the fact number followed by the fact.

- **Hobart** students use the url `http://lawson.cis.edu.au/~rgibson/smashA2/source.php` This provides a random "lightbulb joke" marked up in XML as in the following example:

```
<joke>
<question>How many philosophers does it take to change a lightbulb</question>
<answer>
Three.  One to change it and two to stand around arguing over whether or not the lightbulb exists
</answer>
</joke>
```

  - The "outside information" should show the question followed on the next line by the answer.

- NOTE: The url values given above are *not* typos – Launceston students are to use mashup information supplied from the Hobart server (on `alacritas`) and Hobart students are to use mashup information supplied from the Launceston server (on `lawson`).

Extra Information and Resources

DATABASE STRUCTURE

The database to be used for this application is KXT209WHW , which has 3 tables with the following structures.

The events table: (All students in the unit have SELECT access (only) on this table.)

Field	Type	Null	Key	Default	Extra
Eid	int(6) unsigned	NO	PRI	NULL	auto_increment
year	varchar(6)	NO		NULL	
shortdesc	varchar(200)	NO		NULL	
details	text	YES		NULL	

sample data in this table looks like this

Eid	year	shortdesc	details
2013	1559	tobacco introduced to Europe	Tobacco was introduced to Europe through Spain and Portugal in 1559.
2015	1584	potato introduced to Europe	In 1584 the potato was introduced to Europe.

2017	1610	chai introduced to Holland	"Tea, or ""chai,"" was first imported from China to Amsterdam in 1610."

The tags table: (All students in the unit have SELECT access (only) on this table.)

Field	Type	Nul	Key	Default	Extra
Tid	int(6) unsigned	NO	PRI	NULL	auto_increment
tagname	varchar(20)	NO		NULL	

sample data in this table looks like this:

Tid	tagname
27	Africa
37	archaeology
38	Australia
39	Austria
40	Belgium

The links table: (All students in the unit have SELECT and INSERT access on this table.)

Field	Type	Null	Key	Default	Extra
event	int(6) unsigned	NO	PRI	NULL	
tag	int(6) unsigned	NO	PRI	NULL	

The value in the event field must be the same as the Eid value of an event and the value in the tag field must the same as the Tid of a tag, both fields combined are the primary key. The way this table is set up means that an attempt to insert an entry where one of the values is not a key in the relevant table (eg INSERT into links values (9999,9999)) will cause an error.

sample data in this table looks like this:

event	tag
2011	55
2011	94
2018	42
2018	56

PROGRAMMING GUIDELINES AND RESOURCES

- The lecture notes / self study exercises / demonstrations / tutorial exercises provide "design patterns" for all of the coding you need to complete this assignment.
- All of the PHP files should have a PHP block at the beginning of the file that contains as much of the processing as possible. This block should always contain (at least)
    - a doc comment at the beginning (doc comments start with `/**` and end with `*/` this should provide information about the file including (where relevant) information about the "state maintenance" issues that need to be resolved for this script to function correctly and the ways in which these issues have been resolved.
- Wherever applicable, html components of the title bar, footer, and maincontent section of the pages should be generated by PHP code or "included" (`include_once`) from another file, rather than being hard coded in XHTML.
- Programs should make use, wherever possible, of built-in PHP functions. Always investigate these carefully before use and try to develop a solution that uses the most suitable function for the purpose. In particular you will generally be penalised if you write your own code to do some task that could have been done with the use of a built-in function.
- Code for tasks that are repeated should (if possible) be placed in (programmer defined) functions in the `myfunctions.inc` file.
- *No* deprecated HTML elements or attributes are to be used.
- *No* client side programming (eg Javascript) to be used.

SUBMISSION DETAILS

- Make sure that the folder in which you have all the files for this assignment (as specified in the diagram above) is enclosed in a folder that is labelled with your username (only).
- Download the word version of the School of Computing & Information Systems "assignment cover sheet" (follow the links from the home page), edit it to contain your details, and copy the edited version into the folder containing all the files for this assignment. In doing this you are deemed to have submitted a signed version of the cover sheet.

- Make a .zip archive of this folder. On a MAC this is very easy – in the finder select the folder to “zip”, right click and select the relevant option from the drop down list you will see (this will be labelled “compress …” or possibly “create archive”), the system will then create a .zip file. Say your username was astudent and all of the files for the assignment submission were in a folder labelled “astudent” then the archiving (zipping) process would produce a file called “`astudent.zip`”. If you choose to use a PC the method is slightly more complicated. Consult the School of Computing & Information Systems system documentation for the relevant information.
- Go to your home directory on `alacritas` (Hobart students) or `lawson` (Launceston students).
- You will find there a directory (folder) called `kxt209submit` or a directory (folder) called `kxt309submit`. (the “submit” folder) (If you do not have a submit folder, please contact the Computing Help desk.)
- Copy your `.zip` file into this directory. Open the submit folder to see that your file has been copied. If you want to resubmit, open the submit folder, remove the old version of your submitted work and copy in the new version.

Marking information

This assignment will be marked out of 100. The performance standards required for full marks for each section are shown below. Where your performance standard is below the levels shown here your mark for that section will be reduced.

Criterion	Performance standards	Maximum mark for this criterion
Makes safe use of information from the client.	• All data that is sent in an HTTP request is extracted from the superglobal variable before being included in the contents of a web page.	**N/A** **This criterion is essential** – if you do not match the required performance standard for this criterion you will receive a *failing* mark for this assignment.
Follows acceptable programming practice (as defined in this specification and other unit materials)	• All data received from the user is checked (validated) before being used to generate contents of the web page. • All data received from the user is checked (validated) and / "sanitised" before being used to generate a MySQL query. • All references (uri values) within the site are relative • There is no client side programming • No deprecated HTML elements or attributes are used	**N/A** **This criterion is important** – if you do not match the required performance standards for this criterion your mark for this assignment will be reduced by between 1 and 2 grade levels

	• Associative array keys that are strings are not actually strings • Display is not seriously broken • Minimal PHP processing within the HTML part of pages • No other serious violations of the programming practices required for this unit	
Correct submission of work as specified	• files correctly named • directory structure as specified	**2 (2%)**
Overall structure of application is as required	• layout of pages essentially as specified • utilities links work correctly • authorinfo page has contents as specified • navigation bar works correctly • appropriate banner headings for each page • "Extras" section appears correctly on all of the pages where it is required • "Outside information" appears correctly on the pages where it is required.	**5 (5%)**
Information about authorised users is held in a suitable database table.	• database table with required fields • new user correctly added to database table • information in the table used correctly during sign in and sign up	**5 (5%)**
Authorinfo page contains a thoughtful and complete answer to the question posed.	• advantages and disadvantages identified • extra programming requirements identified • alternative methods discussed	**5 (5%)**
"Extras" section of every page contains the appropriate information	• sign in seen when user is "public" • sign out seen when member is signed in • tag cloud seen where appropriate	**10 (10%)**
The "outside information" is correctly placed on the	• information correctly placed in footer • information extracted from the correct	**8 (8%)**

required pages.	source • information correctly displayed	
System allows all users (public and members) to search the data collection.	• several selection options available • search occurs even if no selections are made (finds everything) • search copes if no matching entries found	**5 (5%)**
System displays the information about an event in a acceptable manner – including an indication of the tags that have been attached to this event where there are links to allow all users (public and members) see all other events that have been tagged with each of these tags.	• information about each event clear and easy to read • tags for each event shown • tag information is link to correct page that gives correct output • paginated output • correct information extracted (for the selection options entered)	**15 (15%)**
System allows *only* members, who are currently signed in, to add tags to events.	• only members who are logged in can add tags • suitable interface for user to nominate the tags • database is protected from SQL injection attack • member receives correct information about outcome of tagging • `links` table in the KXT209WHW database is correctly updated • "number of events tagged" field in the members database table is updated correctly	**15 (15%)**
Users are able to "sign up" to become members. The "sign up" operation functions as specified.	• Form is displayed correctly (shows required fields and information) • Label text is correctly placed • It is possible for the user to *not* enter a	**10 (10%)**

	date • username field is correctly validated • password fields are correctly validated • date (if entered) is correctly validated • CAPTCHA id validated correctly • Form behaves correctly when first loaded • Form behaves correctly when there is some invalid entry (including retaining correct values where this is specified) • Form behaves correctly when all of the entries are valid	
"Sign in" and "sign out" operations (for members) operate correctly. (This means that an authenticated session is set up when the member signs in and comes to an end when they sign out.)	• "Sign in" form seen when user is "public" • Authenticated session begins when there is a successful sign in • "Sign out" option seen when member is signed in • Authenticated session ends when member signs out. • Session data is stored in $_SESSION	**10 (10%)**
Code is well written	• Code can be easily read and the logic followed • Uses built-in PHP functions where possible • "state information" included in the header comments everywhere that this is applicable • User written functions are used where possible and sensible • Follows correct naming conventions for variables and functions Complies with the required coding standards	**10 (10%)**

Help and hints:

- Read the specification carefully and make sure that you know what the PHP application needs to do.
- Look at the examples in self studies, tutorials, and lectures, to see whether you have seen similar tasks before.
- Try out techniques that you think will do the job.
- DO NOT neglect your self study and tutorial work in the unit to work on the assignment. Some of the activities may lead you to see what needs to be done in the assignment.
- You may seek help with this assignment in normal consultation times for this unit or via email to your lecturer. Here are some hints to make best use of help.
    - Make sure you know (or think you know) what your problem is.
    - Have details of the work you have done so far and the progress you have made on hand when you seek help.
    - The more specific you can be in your request for help the more immediately useful the help is likely to be.

PLAGIARISM AND CHEATING:

Practical assignments are used by the School of Computing and Information Systems for students to both reinforce and demonstrate their understanding of material which has been presented in class. They have a role both for assessment and for learning. It is a requirement that work you hand in for assessment is substantially your own.

Cheating

- Cheating occurs if you claim work as your own when it is substantially the work of someone else.
- Cheating is an offence under the Ordinance of Student Discipline within the University. Furthermore, the computing profession has ethical standards in which cheating has no place.
- Cheating involves two or more parties.
- If you allow written work, computer listings, or electronic version of your workbook to be borrowed or copied by another student you are an equal partner in the act of cheating.
- You should be careful to ensure that your work is not left in a situation where it may be stolen by others.
- Where there is a reasonable cause to believe that a case of cheating has occurred, this will be brought to the attention of the unit lecturer. If the lecturer considers that there is evidence of cheating, then no marks will be given to any of the students involved. The case will be referred to the Head of School for consideration of further action.