

ASSIGNMENT 1 SPECIFICATION

INTRODUCTION

The aim of this assignment is to begin the implementation of an application "What happened When" (WHW), the implementation will be completed in assignment 2.

The idea is that there will be a "data store" (of arrays for this assignment, this will be upgraded to a database in assignment 2) of "important" events (most of them will be important to somebody). Users will be given an interface to interact with this "data store" in various ways - the interactions required for assignment 1 are detailed below (see SYSTEM DESCRIPTION).

The data store contains information about events in the following form:

- The data about each event is held in an associative array which has elements with the following keys
 - `id` – a unique identifier for the event (eg E01)
 - `type` – the kind of event (eg birth)
 - `date` – the year in which the event took place (there may be many events that took place in the same year)
 - `shortdesc` – a brief statement of what the event was
 - `details` – some more detailed information related to the event
- There is an array `$events` that contains all of the events.
- See the code and comments in the file `fixedinfo.inc` for details of the other contents.

DUE DATE AND SUBMISSION METHOD

Due date: Wednesday 14 April 3pm (week 7 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 10% 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 <i>failing</i> 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	5%	2
Overall structure of application is as required	15%	3
Years section functions correctly	10%	1, 3
Quiz section functions correctly	20%	1, 3
Members signup section functions correctly	30%	1, 3
Code is well written	20%	2, 3

SYSTEM DESCRIPTION

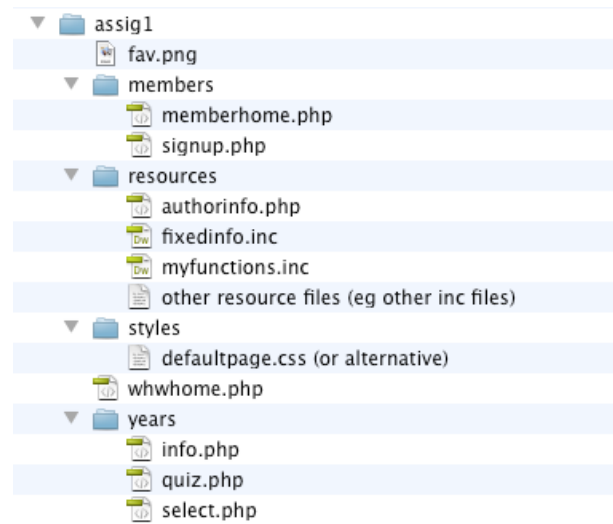
Components of the system

For this assignment you are required to implement the following parts of the system. See below for the detailed requirements for each of these sections

- the home page
- the years section (select page and information page)
- the quiz section
- the members sign up form section
- the author information page

File organisation

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



NOTE:

- These files should be in another folder labelled with your username.
- You will be supplied with the file `fixedinfo.inc` (it can be downloaded from MyLO). You *must not* make any changes to the file name or the structure of this file – that is add any other variables or functions. Your application will be tested with an alternative version of this file (same file name and structure but different data). The comments in this file explain the data structures found there.

- 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 ``) or attributes (eg `align`).

Page layout

Every page of the application that you submit (except the `authorinfo.php` page) 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.
- The navigation bar [4] should contain entries for:
 - home – that links to `whwhome.php`
 - years – that links to `select.php` in the years directory
 - quiz – that links to `quiz.php`
 - member signup – that links to `signup.php`
- 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 extras section [6] for every page except `signup.php` should contain a "quotation". This should be drawn randomly from the quotation collection given in the `fixedinfo.inc` file.
- The footer section [7] should contain the following information:
 - The name of the author
 - The date when the file was last modified.



Detailed specifications for each page

The home page (whwhome . php)

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

The years section

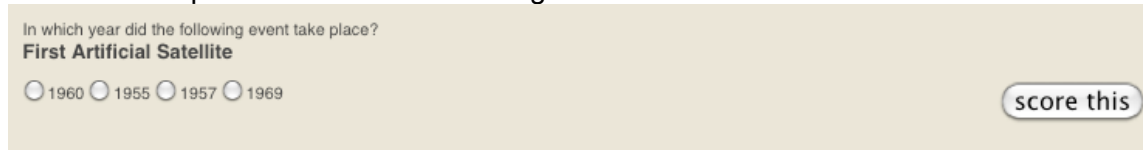
The page `select.php` should present the user with a list of the years (listed in ascending numerical order without duplicates) of the years for which information is available. This list of names should be generated from the information in `fixedinfo.inc` and each year should be a link to the information page (`info.php`).

The page `info.php` should extract information about the event (or events) that occurred in the selected year and display it in a simple table. The information about which year the user has selected should be conveyed (from `select.php` to `info.php`) in the querystring. (In a manner similar to that used the the `picopedia1` application shown in lectures and demonstrations.)

The quiz section

This will consist of a single page (`quiz.php`). When the user selects this option from the navigation bar they will see the page `quiz.php` with a single multiple choice question displayed. It will be a question about one of the events in the datastore variable `$events` (selected at random).

- The question will look something like this:

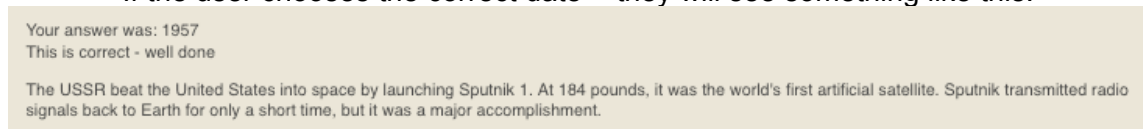


In which year did the following event take place?
First Artificial Satellite

☐ 1960 ☐ 1955 ☒ 1957 ☐ 1969

score this

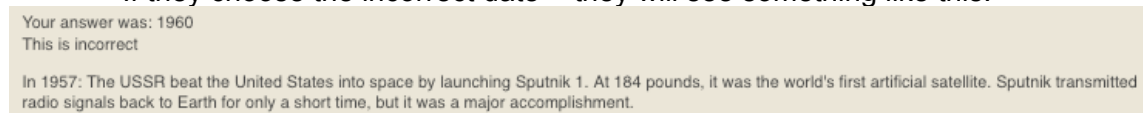
- The text "In which year did the following event take place" will be fixed.
- The "stem" of the question (here it is "First Artificial Satellite") will be the contents of the element with the key value "shortdesc" (for the selected question).
- There must be radio buttons with four possible dates, these must include the correct date plus 3 other dates (with no duplicates).
- The "score this" button causes the form to be submitted to the server.
- If the user chooses the correct date - they will see something like this:



Your answer was: 1957
This is correct - well done

The USSR beat the United States into space by launching Sputnik 1. At 184 pounds, it was the world's first artificial satellite. Sputnik transmitted radio signals back to Earth for only a short time, but it was a major accomplishment.

- If they choose the incorrect date - they will see something like this:



Your answer was: 1960
This is incorrect

In 1957: The USSR beat the United States into space by launching Sputnik 1. At 184 pounds, it was the world's first artificial satellite. Sputnik transmitted radio signals back to Earth for only a short time, but it was a major accomplishment.

- The text shown in each of these will be the contents of the element with the key value "details" (for the selected question).

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 (list in `fixedinfo.inc`)
- password – This is a required field. To be valid must enter 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 must enter 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)

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 data 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.
- If all of the entries are valid, the user will see a message that:
 - Tells them they are now registered as a member.
 - Provides a link to the page `membershome.php` (which need have no contents in the main content `<div>` other than a message to say that the contents of the page will be available later
 - Gives them an indication of the "strength" of the password they have selected. There are many ways of estimating the "strength" of a password. You are to use a fairly simple calculation method where the strength is determined by a score that is calculated as follows:
 - Start with a score of 0, add 1 to the score for each of the following:
 - number of characters is 6 or more
 - number of characters is 12 or more
 - contains both upper and lower case letters
 - contains digits (0 – 9)
 - contains "special" characters (*, @, ! etc)
 - It can be seen that the score will be a number between 1 and 5. This is the information that the user will be shown – something like "The password you entered has a strength of 3 (the maximum possible strength is 5)".

Programming guidelines and resources

- The file `fixedinfo.inc` is available (as a zip file) for download from MyLO
- 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 60. 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.	<ul style="list-style-type: none">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 <i>failing</i> mark for this assignment.
Follows acceptable programming practice (as defined in this specification and other unit materials)	<ul style="list-style-type: none">All data received from the user is checked (validated) before being used to generate contents of the web pageAll references (uri values) within the site are relativeThere is no client side programmingNo deprecated HTML elements or attributes are usedAssociative array keys that are strings are not actually stringsDisplay is not seriously brokenMinimal PHP processing within the HTML part of pagesNo other serious violations of the programming practices required for this unitApplication behaves correctly when the content of the datastore is changed	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
Correct submission of work as specified	<ul style="list-style-type: none">files correctly nameddirectory structure as specified	3 (5%)
Overall structure of application is as required	<ul style="list-style-type: none">layout of pages essentially as specifiedutilities links work correctlyauthorinfo page has contents as specified	9 (15%)

	<ul style="list-style-type: none">• navigation bar works correctly• appropriate banner headings for each page• "Extras" section appears correctly on all of the pages where it is required	
Years section functions correctly	<ul style="list-style-type: none">• List of years is correct – in order and without duplicates• Selecting a year produces the correct output for that year• Display of the data for a year is acceptable• Year input is checked (validated) before use	6 (10%)
Quiz section functions correctly	<ul style="list-style-type: none">• Random selection of the question• Stem and possible years are correct• Form control that is used only permits input of a single choice• HTML for the question <i>does not</i> contain the answer as a hidden component• Performs correctly when the answer is correct• Performs correctly when the answer is incorrect	12 (20%)
Members signup section functions correctly	<ul style="list-style-type: none">• Form is displayed correctly (shows required fields and information)• Label text is correctly placed• It is possible for the user to <i>not</i> enter a date• username field is correctly validated• password fields are correctly validated• date (if entered) is correctly validated• 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	18 (30%)

	<ul style="list-style-type: none">entries are validpassword "strength" is calculated correctly	
Code is well written	<ul style="list-style-type: none">Code can be easily read and the logic followedUses built-in PHP functions where possible"state information" included in the header comments everywhere that this is applicableUser written functions are used where possible and sensibleFollows correct naming conventions for variables and functionsComplies with the required coding standards	12 (20%)

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.

Checklist – when you can answer yes to all of these you are ready to submit the assignment.

Q: Are all the files contained in a folder (directory) that is labelled with your user name?

Q: Are all the files that are required present?

Q: Have you checked the function of all aspects of the application and documented any problems in the authorinfo page?

Q: If you have changed any file names or locations, have you rechecked the function?

Q: If you have checked that all of the uris are relative references? Your application will be moved to a different location for testing – any links back to your alacritas / lawson account will be noticed and penalised.

Q: Have you filled in a copy of the School of Computing & Information Systems assignment cover sheet and added it to the folder that you will submit?

Q: Have you made a .zip file of the final version of your folder?

Q: Do you have a submission folder for this unit (kxt209 or kxt309) in your alacritas / lawson account?