COURSEWORK ASSESSMENT SPECIFICATION

Module Title:	*Web Application Integration*
Module Number:	*CM0665*
Module Tutor Name(s):	*Rob Davis*
Academic Year:	*2017-2018*
% Weighting (to overall module):	*100%*
Coursework Title:	*System Building Assignment*
Average Study Time Required by Student:	*50 hours*

Dates and Mechanisms for Assessment Submission and Feedback

Date of Handout to Students: ***15th January, 2018***
Mechanism for Handout to Students: ***Using eLP***
Date and Time of Submission by Student: ***Before 4.00pm 25th April 2018***
Mechanism for Submission of Work by Student: ***Using the assignment handler on eLP, method described in the assignment.***
Date by which Work, Feedback and Marks will be returned to Students: ***Within three working weeks of the submission date***
Mechanism for return of assignment work, feedback and marks to students: ***Via eLP***

Table of Contents

System Scenario

You are to write a library system for clients to be able to browse and search music CDs, to see information about them together with related cover art. Additionally, if a user logs in to the system, they should be able to save and view notes about a CD (album), said notes being only visible to the logged in user who wrote them.

The system itself will divide in two parts: server-side scripts accessing data in a database and in files which then returns that data to a client-side application.

General Overview of Requirements

Client-side

- Your application should be easy to use, with appropriate UI elements, and be built using appropriate Object Oriented coding techniques which demonstrate an MVC approach characterised by the design and coding of decoupled components which communicate with each other by using events or by implementing the observer pattern.
- The client-side application should be written in JavaScript using AngularJS. Your code must be largely in external linked js files. You can **only** use the AngularJS Application Framework using the modular coding style covered in the course tutorials, and using the AngularJS used in those tutorials. You may not use any other libraries (Bootstrap, jQuery, DOJO etc). Your javascript code should be well written, object oriented and validate, using jslint, without errors and be verbosely commented. Details of the settings you should use for jslint testing are in appendix IV. You can use html5, which validates, without errors, and which also observes the module's "house rules"
  (details of which are at: http://unn-isrd1.newnumyspace.co.uk/learn/#!/book/2/chapter/1/page/7)

Server-side

- The interaction between the client-side application and the database and other server-side files should be via scripts written in PHP. These scripts should use, wherever possible and appropriate, object oriented techniques like those covered in the semester one workshops.
- The database for the assignment is an sqlite one and all interaction with it should use the PHP PDO database abstraction layer.
- You should make sure you have php set to display **all** errors, **do not turn errors off**, or suppress errors with the @ operator.
- When we come to mark your work we may need to change to a different database, an sqlite one in a different location, or even a mysql one so, make sure your database connection information is centralised and easy to alter. This will mean you use:
  a singleton class to control connection to database with connection details contained in (*shown in decending order of preference*) either:
  - A registry class with xml config file.
  - Included in some sort of setenv.php file.
  - Embedded in the pdo singleton class which handles the connection.
- You should use a **single** php script to act as a controller for *all* http requests. This script should interface with appropriate record set and other classes to handle all database interactions. The client-side UI and the testing page should make **no** server calls other than ones to this one single controller; this file should be named index.php.

Implementation Detail

Please note that you may not be able to implement all the requirements that follow, this is to be expected. With a technical assignment such as this the requirements include some quite difficult parts that only a very strong student will be able to complete – this allows us to give a range of marks appropriate to ability. Look at the marking schema for some idea of the breakdown of marks.

Overview

Make sure you look at the database ERD in appendix I, this will help you better understand some of the detail that follows.

You will need to implement the following via a user interface written in JavaScript using AngularJS and supported by server-side scripts written in php.

Ordinary User functionality

1. A method of viewing all albums.
2. A means of choosing a genre from a list and showing only the albums of that genre.
3. A search facility to allow the user to see all albums that match the search condition.
4. A method of selecting a album to see a track listing and other appropriate detail.

Admin functionality

5. A means by which a registered user can login to the application. You should use server-side sessions to manage persistence of the login state which you will need to check both client and server-side to prevent access to that functionality available only to logged in users. The client should hold login state in memory only and not write or read from localstorage.
6. When a user is logged in and selects an album then the functionality in point 4 above should be extended such that the user can view and save a note for an album. The user should only see their own notes of course.

PHP Testing Page

7. You should create a separate testing page so that the PHP server-side scripts can be tested independently of the client-side application that uses them. The testing page should use the same single controller/service pipe as your client-side application and it will need to return all the data required by the client. More detail is provided below in Implementation Specifics.

Implementation Specifics

URL

The web page, from which your application & testing php page are linked, must be accessible as:

http://localhost/cm0665-assignment/

If, when marking, we use that url and no page loads then the work won't be deemed to have met the assignment requirements and will receive a mark of zero. Please, therefore, make sure you work and test with such an alias when developing your work.

So, in the cm0665-assignment directory, you should place an index.html file which hosts your application, or links to it, and which contains a link to a separate 'testing.php' file, do make this link obvious please (perhaps in a menu with your other functional links).

Logging In

Registered users should be able to login in and logout of the application via the application itself by typing their userID and password (see the database table 'user').

userID	password
rob@music.com	rob@music.com
jerry@music.com	jerry@music.com

Note that the passwords are stored in the database user table encrypted using password_hash() with PASSWORD_DEFAULT which means you will need to verify any password given by the user against that stored in the database using php's password_verify(). Once logged in you should ensure persistence across all pages by using server-side sessions. Logging in will mean the user has access to the admin functionality mentioned above. Logging out should clear all session information. Once logged in you should display the username somewhere in the application header. Obviously, logging in is handled by your index.php server-side controller which returns appropriate information to your client-side UI.

Do remember that you need to use sessions server-side to manage the user's login status so that even the testing page will allow certain actions only if the user is logged in. In addition the login status should be queried and stored appropriately within your client-side application to control access to certain information and actions.

Viewing all albums

There should be some easy way of seeing a list of all albums to show their title, the year, genre to which it belongs (not the code of course), the artist(s), and the total playing time of the album formatted appropriately. Think carefully about how you will order the album listing.

Choosing a genre

You should show the user a drop-down list of all the genres of music available. Naturally such a list will be generated dynamically from the database. Choosing a genre will change the album listing to show only those albums that match. You should not produce a separate listing of albums you must reuse the main album listing.

Searching

A means of searching for albums is needed. It doesn't need to be a multi-criteria search but any search term entered should search album titles and artists and display the albums where any of those fields match the search term. It should also be a contains not an equality search to allow me to type in part of a word, though you may want to ensure the user types more than 2 or three characters. The search must be a search implemented via data returned from php, it can't be just a filter implemented in AngularJS.

The display of matching albums should reuse the album listing component of course

Selecting an album

When the user clicks on an album to select it you should display a track listing for that album, any album summary information, and artwork. The album listing should remain visible to make it easy to select a different album.

The track listing should show: the track name, the track number, the artist, the playing time, and the size. The playing time is stored in milliseconds but should be displayed as hh:mm:ss, though for most tracks mm:ss is all that'll be needed.

Saving and viewing

When a user clicks on an album to see details of it, if that user is logged in, then, in addition to the track listing that appears they should also see a separate pane showing any notes *they themselves* have made for that album.

A logged in user should be able to write and save notes, creating a new note if none exists and updating the note if one is already there. Note that the primary key for the notes table prevents a user from writing more than one note for each album.

PHP testing

To allow us to test the functionality of just the php, independently of the client-side application, you should include a link to a separate html page called 'testing.php'. This page will include links to do at least the following:

- Show albums (e.g. cm0665-assignment/server/index.php?action=listalbum)
- Show genres (e.g. cm0665-assignment/server/index.php?action=listgenre)
- Show albums matching genre (e.g. cm0665-assignment/server/index.php?action=search&genre=2)
- Show search results (e.g. cm0665-assignment/server/index.php?action=search&term=dead)
- Show tracks for an album (e.g cm0665-assignment/server/index.php?action=showtracks&pk=2)
- Show notes (e.g. cm0665-assignment/server/index.php?action=shownote&pk=1)
- Allow a user to login and logout (will need a form)
- Allow a note to be changed or created (will need a form)

Your php marks will largely derive from being able to test the required functionality using your test page. Make sure your tests demonstrate all functionality, any functionality not provided won't receive a php mark.

In the example code above, I've use 'action' as the attribute, you may have implemented other attributes, which is fine of course, it's the functionality (and the way in which it's implemented), the data, and its ordering which matters.

Advice

- Please make sure you plan your code and the way different components and classes will work together *before* you start writing any code. Jumping in to writing code without proper planning is a sure way to waste time and create inflexible designs.
- Implement your solution using a variety of well designed classes. In the semester one work we produced almost all the classes, or ones very similar to, those you'll need for server-side code for this assignment.
- Make sure your code is indented and commented – your comments should be written first – writing pseudocode, which become comments, will help make sure your code makes sense. That is true for both the server-side and client-side code –look at the work on documentation for examples of good commenting.
- Attempt all parts of the required functionality, it's easier to get a few percent at the start of a question than get those last few percent trying to get a perfect solution to one part. In short, getting fewer than half marks for all parts is better than getting almost full marks on only one part.

- Make sure you test your code. Test that it works when you do what you're supposed to and when you do something you're not. Try at least the tests listed in appendix II.
- Try to get someone unconnected with your system to try it out, watch what they do –is it easy to use?
- Since some tables contain thousands of records you might want to consider implementing 'paging' by using the 'limit' clause in your sql. There are 1,100 albums and 11,000 tracks so displaying all of those all at once is probably not a good idea, though paging might prove too difficult for you to implement.
- Before finally submitting your zip file test it by unzipping to the D: temporary drive in the Pandon labs and run a web server from that drive and make sure your web-page and embedded application works using the url the assignment specifies:

  http://localhost/cm0665—assignment/

  If it doesn't work it probably means you've not created the zip correctly – zipping your web assignment directory and all sub-directories. Don't leave such testing until the last minute.

Assignment Meta Information

Learning Outcomes

The aim of this assignment is to allow you to adequately fulfil the following learning outcomes as specified in the Module Descriptor:
(http://nuweb.northumbria.ac.uk/live/webserv/mod.php?code=cm0665)

1. Ability to develop multi-tier systems with a database backend, taking into account security and transaction integrity
2. Build non-trivial server-side components
3. Integrate and test software components which reside on a either a web or database server
4. Build server-side components for structured data processing over the web
5. Specify, design and implement integrated systems for data processing over the web with mixed data sources.

Assignment Structure & Submission Details

This assignment constitutes 100% of the assessment for this module and will require you to use all you've learned in both semesters' classes. The aims of the assignment, which will lead you to the fulfilment of the module learning outcomes, are to:

1. Develop a web-based application using a UI developed using the AngularJS Framework, which retrieves, displays and allows editing of data stored in a database accessed by means of Object Oriented PHP server-side scripts. All database access being via the PDO abstraction layer allowing easy switching between at least sqlite and MySQL databases.

It is an individual piece of work.

Assignment Submission

You must submit the assignment ***Before 4.00pm 25th April 2018***.

Please do remember that a deadline is a deadline, work submitted even a minute late, without the appropriate approved technical or personal extenuating circumstances form, will be penalised according to the University Regulations. **Don't** leave things 'til the last minute and risk a slow server causing you to miss the deadline, servers always run slowly, especially when you're in a hurry.

To hand in your work you must zip all the files and subdirectories that make up your system into **one** zip file named thus: ***yourstudentID_music.zip***. Obviously, replace the part of the name that says 'yourstudentID' with your own studentID. Please also make sure you **only** use the zip format, rar or any other compression format won't be accepted. **Do not include ANY** image files in your zip, when your work is marked we'll use the original files we already have to save you uploading 33Mb of images.

The zip file you then submit to the link ">>View/Complete Assignment: CM0665 Assignment Submission" found in Assessment => Assessment – Music

You may not and must not hand in your assignment in any other way or to any other place, CDs, DVDs, zip files sent to tutors via email or handed-in at the School Office are not acceptable.

Viva-voce

As part of the normal marking of your work you may be asked to attend a viva-voce to talk about your work and explain the code you've written. Your ability to explain your work may affect any mark you might otherwise be given. The viva will normally take place in the assessment period and you may be required to attend up to the 25th May 2018, do make sure you're available. You'll be notified about whether you're required to attend a viva within two weeks of submitting your work.

Assignment Feedback

Your feedback will be available to you from within Gradebook. The marks themselves will be also be available from Gradebook which is accessible from eLP.

Marking Schema

A general overview of the marking criteria is included in appendix III.

	Marks
PHP Server-Side (50%)	
Coding standards: commenting, clarity, code design	8
Good use of classes: singleton, registry, record-set etc	5
Functionality: reading and writing appropriate data sets Approximately 4 marks for each of • ordered album list returned • choosing category returns matching albums • search term returns ordered albums • login process returns appropriate data and sets/clears session • choosing album shows required information • logged in user only sees own notes • logged in user only can save a note	28
Security: sessions, prepared statements, defensive programming	9
Total for this component	50%

Client-Side Application (50%)	
Coding standards: commenting, clarity, code design	10
Architecture: mvc with appropriate components, classes and events	10
UI: look and feel	5
Functionality: depth & breadth of required functional implementation Approximately 4 marks for each of • ordered album display • choosing category shows albums (with MV reuse) • search term shows albums (with MV reuse) • login/logout with appropriate display based on state • choose album displays required information • read, create, update note if logged in.	25
Total for this component	50%

Total 100%

Academic Regulations

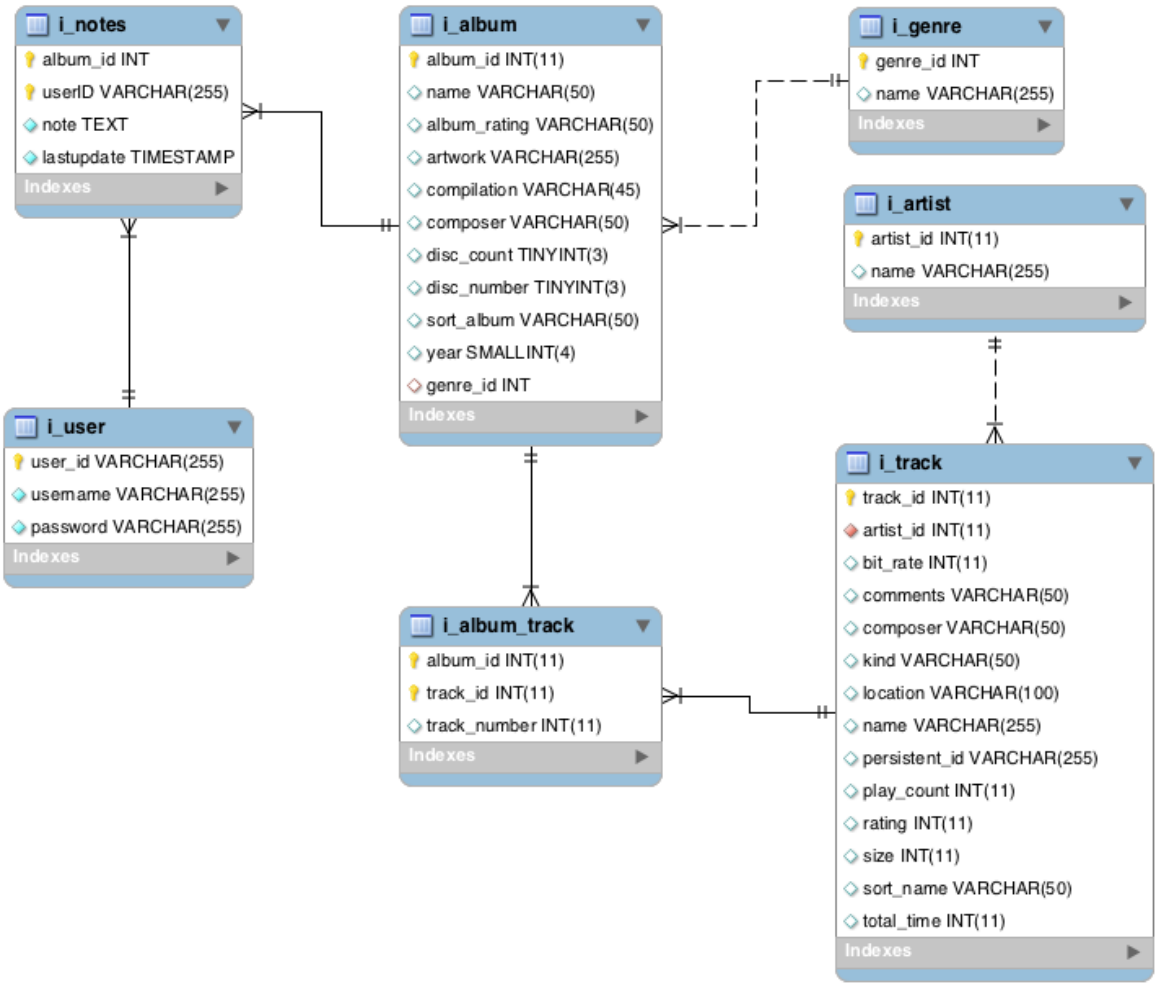
This is an individual form of assessment and all of the contents/coding of your web pages, for all parts of the assignment, must be entirely your own work.

You must adhere to the university regulations on academic conduct. Formal inquiry proceedings will be instigated if there is any suspicion of academic misconduct in your work. Refer to the University's regulations on assessment if you are unclear as to the meaning of these terms. The latest copy is available on the university website.

Appendix I

ERD –database structure

Primary keys are shown as underlined attributes, and foreign keys with an asterisk following the attribute name. The cardinality of relations is shown using 1…* rather than by using crows-feet. In fact all relations are one to many save that between track and artwork which is one to one. *(uses it_musicNew)*



Notes

Notes table has a primary key which is a compound key using albumid and userID and, therefore, will only allow each user to have one set of notes for each album. Attempting to create a second record using the same albumid and userID will cause a duplicate key database error.

Artwork is linked to an album, the album artwork is in a separate zip within the assignment zip file, note that I've also added a 'no image' image in the assignment zip file you can use if the album has no artwork.

The assignment zip file then, in addition to this document, contains the sqlite database, the no-image.jpg and a music_artwork.zip.

User: the userID is the user's email, the username is their full name. Notice that the password is stored encrypting using md5. Therefore, to compare any user entered password with the one in the user table you'll need to first md5 encrypt the entered password. PHP has its own md5 function you can look up.

Appendix II

Testing

Make sure your code behaves nicely if you choose a genre that has no matching albums. Test this by choosing a genre like: Son, Alt. Rock or Afro Peruvian

Similarly test that those few albums that aren't assigned a genre display correctly and that the genre field is blank: Electro Lounge, Stages, Graceland or Robâiyyâte Xayyâm

There are some albums with no artwork check these display and how do you cope with there being no image to display?

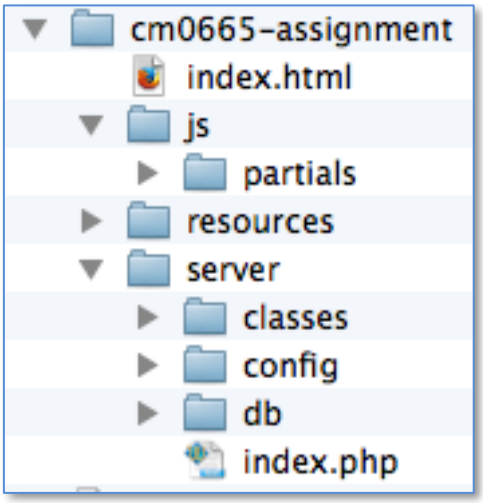
Albumid	**Name**
4	Roots Reggae
104	X&Y
330	Escondida
386	Los Lobos

You should not show system type attributes, they're of no interest to the user, so make sure you don't display any of the ids used in the tables, genre_id, artist_id, album_id etc. You will need to use them, of course, in list boxes or as hidden fields in order to link back to other tables but you shouldn't display them to the user. Check you don't.

Appendix III: Creating the assignment project

When handed in, your assignment must run using the url: *http://localhost/cm0665-assignment/* so all files must be accessible beneath your *WEB-ROOT/cm0665-assignment* directory. What I've called *WEB-ROOT* might be *U:\local-html* in the university, *c:\wamp\www*, or */Applications/MAMP/htdocs*, or *c:\inetpub\www* or something different at home, depending on your particular web setup.

Suggested Directory Structure



You should follow the structure covered in the AngularJS teaching material for this module. So, inside your *WEB-ROOT*, you should have a directory structure like that covered in the tutorials, perhaps like the image on the right for instance.

Assignment submission important

When your assignment is ready for hand-in simply do this:

1. Check that you can load your assignment page using the url: **http://localhost/cm0665-assignment/** and that the links on this page to your flash application and the php tests work correctly.
2. Zip the entire contents of directory cm0665-assignment (making sure you click "use folder/directory names" or its equivalent) but try to make sure you exclude the sqlite file and any image files to minimize the zip-file size. This zip file is what you hand-in.

Config.xml ?

If you've used a config.xml file as outlined in the teaching material for the Application Registry Class then, although you *wouldn't* do this in a production system, store your config file *inside* WEB-ROOT so that it's easier to zip and for me to test.

Appendix IV – Settings for JSLint with JavaScript

You must ensure all your JavaScript code is wrapped in IIFE structures with 'use strict' as the first directive and you must also check your js code reports no errors when pasted into jslint (http://jslint.com) with ALL options set as default except for 'messy white space' which can be ticked.

You may also need to add 'angular' as a global variable in the global variables… box, or you can add: /*global angular*/ as the first line in your script files.

It can take some time to remove errors and warnings when using jslint, it is very strict. So don't leave things until the last minute, get into the habit of checking your syntax constantly so you start to learn what constitutes good code. You'll save yourself a lot of pain and heartache and be quicker in the long run; and needless to say you'll write more robust code too.

Certain editors will syntax check JS for you (though you should also double check with jslint), if you have such an editor try creating a file name .jshintrc in your project root folder and paste this code in that file, that might help with stricter checking:

```
{
  "node": true,
  "browser": true,
  "esnext": true,
  "bitwise": true,
  "camelcase": true,
  "curly": true,
  "eqeqeq": true,
  "immed": true,
  "indent": 4,
  "latedef": true,
  "newcap": true,
  "noarg": true,
  "quotmark": "single",
  "undef": true,
  "unused": "vars",
  "expr": true,
  "strict": true,
  "trailing": true,
  "smarttabs": true,
  "forin": true,
  "latedef": true,
  "plusplus": true,
  "globals": {
    "angular": true,
    "console": true,
    "document": true,
    "SyntaxHighlighter": true,
  }
}
```