Coursework Assessment Specification

Module Title:	*Web Application Integration*
Module Number:	CM0665 and KF6012
Module Tutor Name(s):	*John Rooksby (Module Tutor), Kay Rogage.*
Academic Year:	*2018-2019*
% Weighting (to overall module):	*100%*
Coursework Title:	*System Building*
Average Study Time Required by Student:	*50 hours*

Dates and Mechanisms for Assessment Submission and Feedback

Date of Handout to Students:	*21st February, 2019*
Mechanism for Handout to Students:	*Blackboard*
Date and Time of Submission by Student:	*before 12:00 noon, 15th May 2019*
Mechanism for Submission of Work by Student:	*Upload via Blackboard.*
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 Blackboard*

Table of Contents

System Scenario

You are writing a film information system for a client to allow users to browse film and related information from the supplied database tables. In addition to browse functionality, if a system admin user is logged in, they should be able to save notes about a film and be able to retrieve and re-display any previously saved notes. Such notes would only be visible to the logged in admin user who wrote them.

The system itself will be in two parts: a client-side interface, and server-side scripts accessing information in a database and in files.

General Overview of Requirements

Client-side

- Your application should be easy to use 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. You may not use any other JavaScript libraries (jQuery, DOJO etc) and you must use AngularJS rather than Angular. Your JavaScript code should be object-oriented and validate using jslint without errors. The code must be well written and be verbosely commented. Details of the settings you should use for jslint testing are in appendix IV. You should use HTML5, which validates without errors, and which also observes the module's "house rules/standards" (these are on Blackboard). You may use CSS libraries if you wish, but you will not gain marks for this (if you use a framework such as Bootstrap, be careful not to link to or use any JavaScript library - you will lose marks if you do).

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 descending order of preference) either:
    - A registry class with config file.
    - Some sort of setenv.php file.
    - 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 that 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.

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 films.
2. A means of choosing a category from a list and showing only the films of that category.
3. A search facility to allow the user to see all films that match the search condition.
4. A method of selecting a film to see all the actors in the film and other appropriate detail.

Admin User Functionality

5. A means by which a registered admin user can login to the application. You should use Sessions to manage persistence of login state which you will need to check both client and server-side to prevent access to certain functionality which should be available only to logged in admin users.
6. When an admin user is logged in, then the functionality outlined in point 4 above should be extended to also allow the user to view and create notes.

PHP Testing Page

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 as your client-side application.

Implementation Specifics

URL

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

http://localhost/wai-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 wai-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. You do not need to make this link conspicuous and may lose marks if your site appears unprofessional (perhaps put a link to testing.php in a menu with your other functional links).

Logging In

Registered users should be able to login in and logout of the application. Any one of the two registered users (see the database table 'user') should be able to log in by typing their userID and password (for ease of testing the userID and password are the same).

userID	password
rob@sfilm.com	rob@sfilm.com
jerry@sfilm.com	jerry@sfilm.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 password_verify(). Once logged in you should ensure persistence across all pages by using Sessions. Logging in will mean the user has access to the menus and functionality for saving and retrieving saved notes. 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 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 films

There should be some easy way of seeing a list of all films to show their title, the first 100 characters of their description, the release year, the category to which it belongs (not the category id of course), the rating, and the date of the last update formatted appropriately.

Choosing a category

You should show the user a drop-down list of all the categories of films available. Naturally such a list will be generated dynamically from the database. You can implement a search filter on the category listing if you want.

Choosing a category will change the film listing to show only those films that match. You should not produce a separate listing of films, you must reuse the main film listing component or 'partial'. Since there are about 1,000 films you should not retrieve them all at once, you might want to consider whether you should retrieve only films matching a chosen category or to implement a 'paging' solution of some sort.

Searching

A means of searching for films is needed. It doesn't need to be a multi-criteria search but any search term entered should search at least category and film titles and display the films which match the search term. It should also be a 'contains' not an equality search so, for instance, if I type in 'air' it'll find 7 films (e.g. Affair Prejudice and Airplane Sierra). This search should really pass any search term to php and retrieve matches. An AngularJS Filter approach wouldn't be appropriate since that would mean you would have to retrieve all films.

Of course, the display of matching films should reuse the same film listing component used when a category is selected.

Selecting a film

When the user clicks on a film to select it, the film listing should remain visible, but you should also display full details of the film including all actors in it. All the film fields you display should be formatted appropriately.

Saving and viewing

When a user clicks on a film to see details of it, if they are logged in they should also see a separate pane showing any notes they've made for that film. This should be in addition to the information described above

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 film.

PHP testing

To allow us to test some of the basic functionality of just the PHP, independently of the client side application (SPA), should include a link to a separate html strict page called 'testing.php' (or .html). This page will include simple links that will invoke your PHP scripts just as your SPA does. You must include links in this testing.php page to do at least the following:

- Show films (i.e. *wai-assignment/index.php?action=listFilms*)
- Show search results (i.e. *wai-assignment/index.php?action=listFilms&term=air*)
- Show categroy results (i.e. *wai-assignment/index.php?action= listFilms &category=4*)
- Show actors for an film (i.e *wai-assignment/index.php?action=listActors&film_id=3*)
- Show notes (i.e *wai-assignment/index.php?action=listNotes&film_id=1*)
- Allow a note to be changed or created (using a form with method post)
- Allow a user to log in and log out (using a form with method post)

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.

Advice

- Please make sure you plan your code and the way different components 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. Remember: "*Weeks of programming can save you hours of planning*".
- 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 that you will 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 becomes comments, will help make sure your code makes sense. That is true for both the server-side and client-side code.
- Attempt all parts of the required functionality. It is 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 the required functionality works, and test that it is works appropriately when there is invalid input or actions. 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.
- Before finally submitting your zip file, test it by unzipping it and running an Apache server to make sure your web-page and embedded application works using the url the assignment specifies:

http://localhost/wai—assignment/

If it doesn't work it possibly means you've not created the zip correctly – by 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=kf6012)

Knowledge & Understanding:

1. Ability to develop a multi-tier system for data processing over the web using mixed data sources, including databases, taking into account security and transaction integrity.

Intellectual / Professional skills & abilities:

2. Plan and manage a development project and critically evaluate tools, software architecture and technologies appropriate for it.

Personal Values Attributes (Global / Cultural awareness, Ethics, Curiosity) (PVA):

3. Demonstrate professional and reflective practitioner attributes managing time and evaluating progress aiming for continuing development in the design, build, and testing of a secure web application.

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:

- 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.

If you've any questions about the details and requirements of the assignment there is a discussion board set up on Blackboard for that purpose.

Assignment Submission

You must submit the assignment ***before 12.00 noon 15th May 2019***.

Please do remember that a deadline is a deadline. Work submitted even a minute late, without the appropriate late submission approval, will be regarded as a non-submission and receive the

appropriate deduction. **Don't** leave things until the last minute and risk a slow server causing you to miss the deadline.

To hand in your work you must zip all the files and subdirectories that make up your system into **one** zip file named: yourstudentID_wai-film.zip. Obviously, replace the part of the name that say's '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.

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 Student Central 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. This would normally happen at any time up to the end of the semester two assessment period, though, exceptionally, might happen after that time.

Assignment Feedback

Your feedback will be available to you from within Blackboard. The marks themselves will be also be available from Blackboard.

Marking Schema

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

PHP Server Side (50%)

Coding standards, commenting, clarity, code design	10
Good use of classes: singleton, registry, record-set etc	5
Functionality: reading and writing appropriate data sets. Approximately 4 marks for each of: • ordered film returned • choosing category returns matching films • search term returns ordered films • login process returns appropriate data and sets/clears session • ordered actor listing given film_id • reads, creates and updates user/film note	25
Security: sessions, prepared statements, defensive programming	10
Total for this component	50%

AngularJS Client Side (50%)

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

Assessment Criteria, General Guidelines

The marks schema above should give you a more detailed view of how marks are allocated. However, the following table might be helpful for you to understand, in general terms, the criteria used. The first column indicates the marking range, which can be achieved by achieving the general assessment criteria indicated in the second column:

Mark	General Criteria
(0 – 29)	UI not dynamic or built using wizards or with too much reliance on libraries; or php not using pdo nor any attempt at classes. Most functionality missing.
(30 – 39)	UI poorly designed and not easy to use; or php too basic in design. Large amount of missing functionality.
(40 – 49) Pass	All essential elements have been achieved, although knowledge is adequate, somewhat limited – narrow and/or superficial. In the most part, the solutions provided are lacking in application architecture or design skills.
(50 – 59) Good Pass	Knowledge is up-to-date and relevant to an appropriate breadth and depth. The ability to apply theory to practice is illustrated by the usability, functionality and architecture of the client-side and server- side components, although not all tasks are completed totally correctly and/or efficiently. Some evidence of independent thought and presentation of work is of an acceptable level.
(60-69) Very Good Pass	Knowledge is up-to-date and relevant to an appropriate breadth and depth. A significant ability to apply theory, concepts, ideas and their inter-relationship is illustrated by the usability, functionality and architecture of the client-side and server-side components, although not all tasks are completed as efficiently as possible. Clear evidence of independent thought; presentation of work is fluent, focused and accurate
(70 – 100) Distinction	Exceptional scholarship shown in complex learning environments, through the application of theory to real-world contexts. All required practical elements have been correctly and efficiently completed, the application design meets all assignment requirements, clear evidence of independent thought is shown by the work, which is fluent, focused and well executed with an eye to possible future requirements.

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.

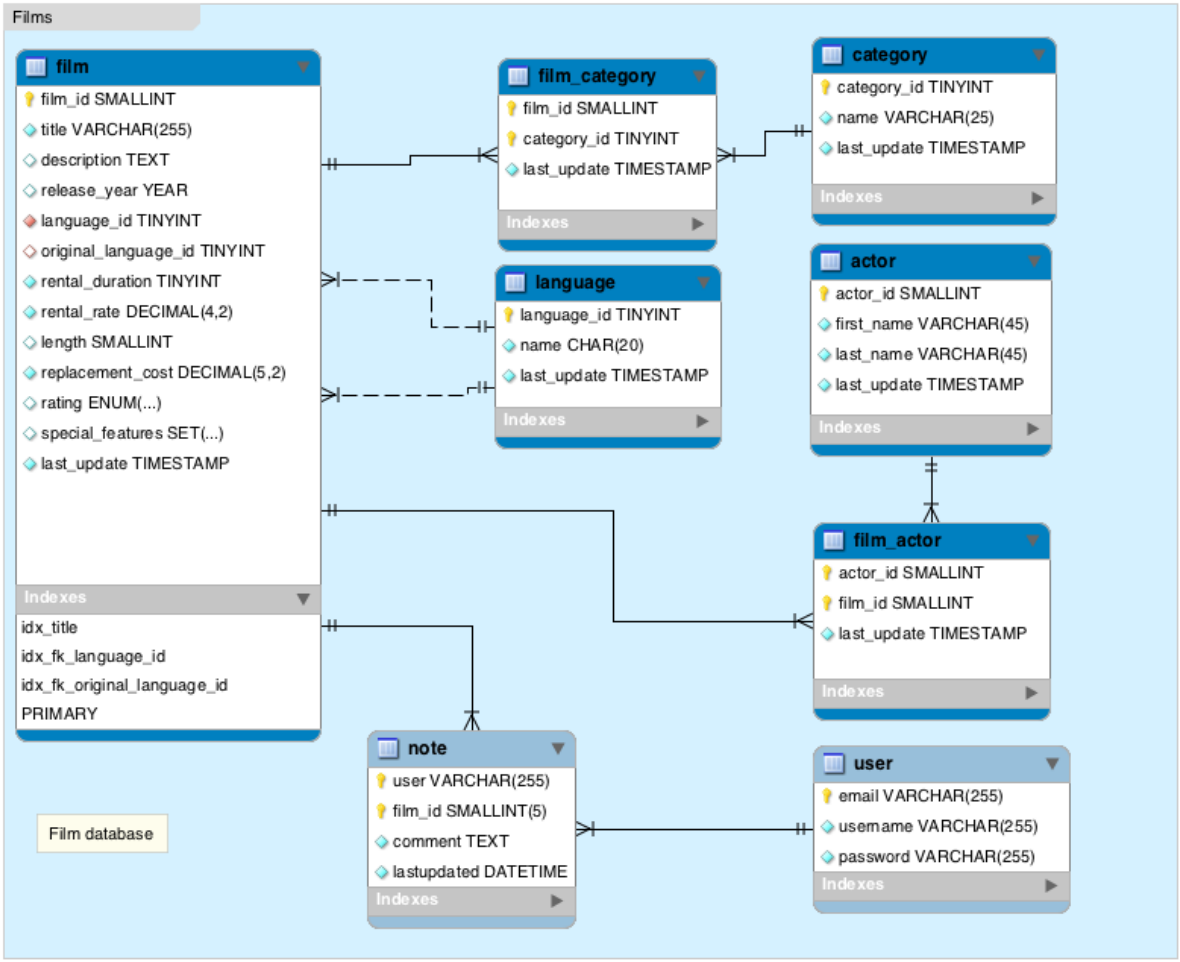
You must adhere to the university regulations on academic conduct. Formal inquiry proceedings will be instigated if there is any suspicion of misconduct or plagiarism in your work. Refer to the University's regulations on assessment if you are unclear as to the meaning of these terms. The latest copy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

Appendix I

Table structure & Relations

Primary keys (PK) are shown with a small key symbol to their left. Foreign keys are usually named the same as their matching PK except in the case of the 'note' and 'user' table, the foreign key field in 'note' is named 'user' which points to the PK user.email.

Note that in the current database all tables are prefixed with 'nfc_'



Notes

The *Note* table has a primary key which is a compound key using user and film_id and, therefore, will only allow each user to have one note for each album. Attempting to create a second record using the same user and film_id will cause a duplicate key database error.

User: the PK is the user's email, the username is their full name. Notice that the password is stored encrypted 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

System Testing

Make sure your code behaves nicely if you choose a category that has no matching films. Test this by choosing the category ‘foreign’.

Check you show, when a film is chosen, all the actors in it, film Dynosaur Academy (1) has 10 actors, and check what you do when there are no actors linked like with films: Drumline Cyclone (257), Flight Lies (323) and Slacker Liaisons (803).

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, film_id, category_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 format data appropriately: film titles, dates, numbers and things like the field ‘special_features’ which looks like CSV.

Check the ordering of data, which ordering makes the most sense, make sure you explicitly specify the ordering via an ORDER BY clause in your sql SELECT statements

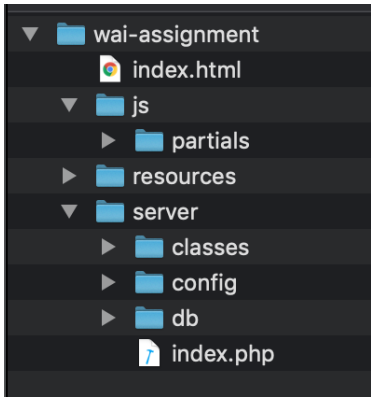
Appendix III

Creating the assignment project With AngularJS

When handed in, your assignment must run using the url: *http://localhost/wai-assignment/* so all files must be accessible beneath your *WEB-ROOT/wai-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 below for instance.



Assignment submission important

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

1. Create an *index.html* in the*WEB-ROOT/wai- assignment* folder with a link to your html application host file (called Main.html unless you changed it)
2. Check that you can load your assignment page using the url: **http://localhost/wai-assignment/** and that the links on this page to your flash application and the php tests work correctly.
3. Zip the entire contents of directory wai-assignment (making sure you click "use folder/directory names" or its equivalent) but try to make sure you exclude the.sqlite file to minimize the zip-file size. This zip file is what you hand-in.

Config ?

If you've used a config.xml / .json 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.

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,
	}
}
```