Coursework Assessment Specification

REFERRAL

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:	*26 June 2019*
Mechanism for Handout to Students:	*Blackboard*
Date and Time of Submission by Student:	21/8/2019 16:00
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 a web developer working for a company that specialises in developing web-based solutions. Your company is trying to win a contract to design a dynamic and interactive web site for GeoWorld. You have been given the task of developing a small prototype of part of the final solution; a facility to show geographical information and to allow admin users to update the name of the Head of State for a country. All this information is stored in both MySQL and sqlite databases.

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 listing or other means of choosing a continent. Selecting a continent will show all the countries on that continent.
2. A search facility to allow a user to see only the countries that match the search term in the chosen continent (searching only country name).
3. Choosing a country from the listing above will show country information in more detail and perhaps even city information, certainly information about the Capital city. You should **not** show the Life Expectancy, GNP, nor Head of State however.

Admin User Functionality

4. 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.
5. An admin user, but not an ordinary user, should be able to see country information for Life Expectancy, GNP and the Head of State. They should also be able to edit and save changes to the Head of State. It would seem ideal, since you already have the functionality for choosing a country and seeing country detail from requirement 3, that selecting a country would allow the admin user to change the Head of State, if they wish, without the need for further navigation steps.

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 'w_User') should be able to log in by typing their userID and password. Below is the structure and contents of the w_User table in the database.

userID	username	password	Role
rob@geo.com	Rob Davis	robGeo1234	Admin
jerry@geo.com	Jerry Garcia	jerryGeo1234	Admin

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.

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 continents (i.e. *wai-assignment/index.php?action=showContinents*)
- Show countries on a continent (i.e. *wai-assignment/index.php?action=showCountriesOnContinent&continent=EU*)
- Show search results (i.e. *wai-assignment/index.php?actionsearch&term=belg&continent=EU*)
- Show country information (remember the information shown will vary if the user is logged in or not) – (i.e. *wai-assignment* /service.php?action= showCountryInfo&a3code=BEL)
- Allow a Head of State to be changed but only if logged in (i.e. wai-assignment/service.php?action=changeHofS&a3code=FRA&HofS=Emmanuel%20Macron)

Do please also look at the marking schema below to see how the php will be marked to help you decide on the testing functionality you need to show.

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 on ***21/08/2019 16:00***.

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-geo.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 country data returned • choosing continent returns matching countries • search term returns ordered countries • login process returns appropriate data and sets/clears session • appropriate country information (data and formatting) • reads, creates and updates Head of State	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	5
UI Look and Feel	10
Functionality: depth & breadth of required functional implementation Approximately 4 marks for each of: • ordered country display • choosing continent shows countries (with MV reuse) • search term shows countries (with MV reuse) • login/logout with appropriate display based on state • choose country displays country and related detail • read, create update Head of State 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.

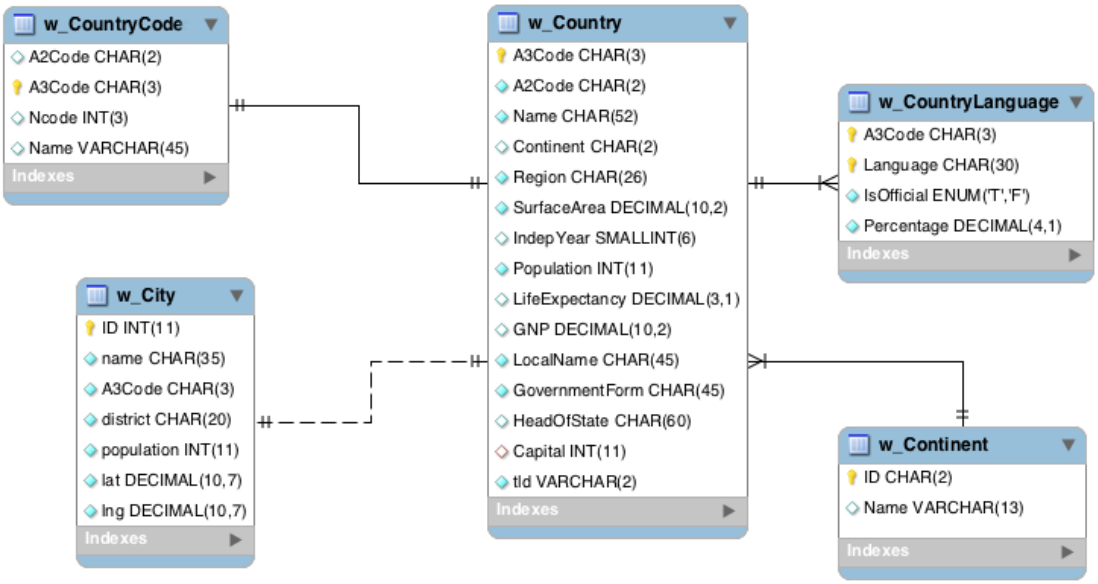
Appendix I

Table structure & Relations

The data in these tables is derived, and is a sub-set of, un-normalised data from geonames.org. It is a snapshot and is therefore out of date with regards to things like Population and GNP etc..

The tables in the database are:

w_City, w_Continent, w_Country, w_CountryCode, w_CountryLanguage and w_User. The relationships between them and the names & data types of their fields are shown in the ER diagram below. w_User is not shown here because it has no relations with other tables, look at the table itself to see its field names and data-types.



Notes

Countries have two codes, an A2Code and an A3Code, the A3Code is the more up to date (ISO- 3166 alpha 3) and the one set as the primary key in the tables that use it. In the w_CountryLanguage table, since each country may have more than one language there is a compound primary key of A3Code and Language.

In w_Country the tld field is the 'Top Level Domain' for that country, I think the purpose of all other fields are self-explanatory given their names.

Table sizes

Table name	Number of records
w_City	4,079
w_Continent	7
w_Country	233
w_CountryCode	249
w_CountryLanguage	984

Note that 16 records in w_CountryCode don't have matches in w_Country and in w_CountryLanguage some records reference countries which no longer exist. The appropriate outer join select statements will confirm if other data discrepancies exist. How significant such discrepancies are depend on your application's data use.

Appendix II

System Testing

Obviously you should check and test all your pages, test they meet the requirements, and show the data asked for. It should be easy for the user to 'navigate' the information shown: choosing a continent, clicking to see country information; searching for a country, and updating country information for Head of State with the minimum number of mouse clicks but without overloading the user with information not needed at the time.

Country display

How does the user chose a continent, is it easy, are you displaying the continents in an appropriate way? Can they easily choose another continent without backtracking? When showing countries on that continent are you showing the basic information for the user's interest without overloading them, but then, when they select a country, are you displaying the detailed information clearly and in a way that looks good, with appropriate formatting to make it easy to read?

Country search

Your country search should, ideally, update or filter the existing list of countries, and of course only within the continent chosen. They may only want to type in part of the country name. However, you should ensure they type in at least 3 characters and if they type fewer make sure it doesn't search but displays an appropriate message. What happens if nothing is found?

Editing Head of State

Make sure you can only edit if logged in. Perhaps only show Save buttons or the like if the user has actually changed the Head of State. Ensure that you can't update the Head of State using direct calls to your server pipe, in other words checking for logged in state should happen both server and client-side.

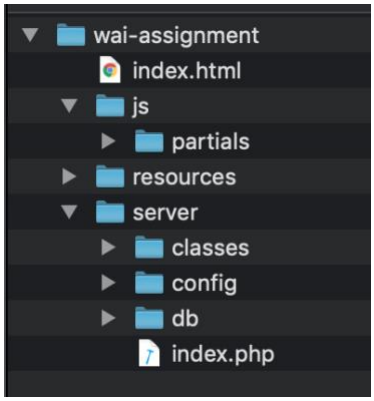
Appendix III

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,
	}
}
```