Coursework 2: Database + web interface project

Due date – 22nd December 2021, 3pm, submitted via Moodle

Coursework Deliverables

You should submit the following deliverables in a **single ZIP file** by the due date.

WARNING: If you submit a RAR formatted archives instead of ZIPs we will mark your submission as ZERO marks.

The ZIP file should be named "DIS-COMP4039-CW2-yourusername-studentID.zip" and should contain the following:

1. A cover page file named "yourusername-studentID_CW2_cover.txt" containing your name, your student ID and your username (please use **.txt / plain text file format**). In this file you **must** have the URL of your working system (i.e., a link to your site on mersey.cs.nott.ac.uk), and a list of all files you submitted for this coursework (including those contained in the ZIP file (below)). **In this file you must also list & specify the origin of ALL resources (e.g., frameworks, assets, styles, JS libraries, code adapted / used from internet sources, etc.) that have used that are not solely your own work—if you do not do so we may consider the absence of such a declaration as indicating plagiarism.**
2. A file named "yourusername-studentID_InstallationFiles.zip" which contains installation files for your system contained in a single ZIP file. These should consist of one or more SQL (**.txt / plain text file format**) files to construct the database (**NOTE: assume case sensitivity in attributes / table names**), and one or more PHP / HTML files that will constitute the web front-end. **If SQL files are not present we may be unable to mark your coursework and you will receive ZERO marks.**
3. Two documentation files (**PDF only please or else these components will be marked ZERO**):
    a. one named "yourusername_UserManual.pdf" which contains the user manual (as described below); and
    b. one named "yourusername_Technical.pdf" which contains the technical manual.

Late submissions will be accepted, but they will be penalised at the University standard rate of 5% per working day.

Plagiarism Policy

Any material that you submit which is not your own work **must be clearly identified as such**. This includes (but is not limited to) written text, figures, diagrams, code (e.g., PHP, JS), SQL statements, and markup (HTML, CSS). Any material that is not

your own work and is not clearly identified and referenced will be considered a case of plagiarism. Plagiarism will be treated seriously, and will be dealt with through the University's disciplinary procedures. Any code that is directly used to solve the problems of creating, populating, or searching the database must be entirely your own work. Text that is not your own must be identified as such by typesetting it in italics and placing it in quotation marks and following it immediately by a reference to the source.

For example, "The database is such an integral part of our day-to-day life that often we are not aware that we are using one" (Connolly and Begg, 2002). For text references full bibliographic information should be reported in the cover page file, such as "Connolly and Begg, Database Systems (third edition), Addison Wesley, Harlow, 2002".

Coursework Specification

For this coursework we will use same scenario as for DIS-COMP4039 Coursework 1A (**coursework-1a-scenario.pdf**). You will need a backend database and you are free to either use your own or adapt the one provided (**coursework-2-database.sql**). Note this will probably need modification.

You should implement an interactive web frontend for your database using HTML and PHP (as well as other technologies such as CSS or JavaScript, as you see fit). The database should be stored on mysql.cs.nott.ac.uk and the website should be served from mersey.cs.nott.ac.uk (see **Exercise 7** for important setup details you must follow). The URL (which will point to an address on mersey) is to be submitted as one of the deliverables (as noted above). **Make sure you test your installation on mersey in good time prior to the deadline** even if you did not develop it there otherwise you may be penalised. (We also suggest testing your work in XAMPP.)

The task is to create a usable end to end system that will allow officers to record and retrieve information about vehicles, people and traffic incidents. You must submit your complete source code as a package of files that could be installed on any computer that has MySQL / Apache / PHP installed (e.g., LAMP). We will be using Google Chrome (available on CS machines) to test your coursework (so make sure web interface elements function correctly in that browser).

Marking criteria

Coursework 2 is worth a total of 40% of the marks for this module (reminder: Coursework 1A & 1B is collectively worth 20%, and the final exam 40%). Coursework 2 is marked out of 80 (each section being worth 10 marks). The mark will be scaled to provide the final mark for this coursework. Most marks are allocated for specific required features, although to achieve the higher end of marking we encourage you to go beyond what is required. There are a few general mark categories that are awarded for the coursework as a whole which will also benefit from creatively going beyond the bare minimum of functionality specified in the questions. Finally, we note that marking will take into account specific code fragments (e.g., PHP, JS, SQL, etc.) that you have been provided with by convenors.

Core required features [60 marks total]

1. A police officer should be able to log into the system using one of the following usernames/password combinations:

Username	Password
McNulty	plod123
Moreland	fuzz42

   Once logged in the police officer should have the ability to change their password.

   IMPORTANT NOTE – in any real world system involving a database and web frontend, security would be paramount. However, addressing security is beyond the scope of this module, so you should ignore this. For example, usernames and passwords can be stored as plain text fields in your database – do not attempt to encrypt them or secure your database in any way.

   [10 marks]

2. The police officer should be able to look up people by their names or their driving licence number (by typing either of these in to the system). If the person is not in the system it should give an appropriate message. This search should not be case sensitive and it should work on partial names (e.g., “John”, “Smith” and “John Smith” would all find John Smith. If there are several people with the same name they should all be listed.

   [10 marks]

3. The police officer should be able to look up vehicle licence plate. The system will then show details of the car (e.g., type, colour etc.), the owner’s name and license number. Allow for missing data in the system (e.g., the vehicle might not be in the system, or the vehicle might be in the system but the owner might be unknown).

   [10 marks]

4. The police officer should be able to enter details for a new vehicle. This will include the licence number, make, model and colour of the vehicle, as well as its owner. If the owner is already in the database, then it should be a matter of selecting that person and assigning them to the new vehicle. If the owner is not in the database they should be added (along with personal information including the license number).

   [10 marks]

5. The police officer should be able to file a report for an incident and retrieve existing reports. Filing new ones will involve submitting a textual statement, recording the time of the incident and the vehicle and person involved. If either the person or the vehicle are new to the system, then appropriate new entry/entries to the database should be added. The officer should be able to record the offence from a list of offences contained in the database. NOTE: a fine is not stored at this stage, because the fine will be issued by a court at a later time, and added by an administrator. The officer should be able to retrieve and edit the report.

[10 marks]

6. An administrator should be able to log into the system with the username “Daniels” and the password “copper99”. The administrator should be able to do everything that an ordinary police officer can do with the following additions:
   - The administrator should be able to create new police officer accounts
   - The administrator should be able to add fines to the database

[10 marks]

Marking criteria: Notes on marking the above features					
0 (zero)	1-3	4-5	6-7	8-9	10
No attempt is made to implement this feature.	An attempt has been made to implement the feature, but it is very seriously flawed and there is no evidence that this approach will fulfil the requirements of the specification.	The approach is fundamentally sound, but the code is flawed such that it doesn't produce the correct results. Code that nearly works but has serious bugs would fall into this category.	Code works and produces the expected results. Form-elements should be implemented correctly and the database should be queried and updated using appropriate SQL. The web UI design is likely to be very basic (e.g., simple forms) but has some provision for error handling, feedback etc.	Previous requirements are met, and also the user input is validated so that user errors produce sensible results. CSS (and possibly JS) should be used to control the user interface. The user interface design should be of fairly high quality, but it is likely to lack the polish of a professional solution. Meets and exceeds core functionality required.	Previous requirements are met AND the code is of high quality, well documented and easily reusable. Web UI should look professional and follow standard usability guidelines (e.g., consistency). Goes creatively beyond core functionality.

Other required features [20 marks total]

Documentation. You should produce two documentation manuals – a user manual and technical documentation. The *user manual* should explain to a user (i.e., a police officer or an administrator) how to use the system. It is extremely important that all of the software features you have implemented should be described in this manual (otherwise you might not get marks for them!). The user manual should be no more than 500 words, though it may have screen shots if you so wish.

The *technical manual* should contain all of the information that would be needed if you were handing this project over to another developer to take it further. It should explain how to install your software (you may assume that MySQL, and a web server implementing PHP are already installed and configured). It should also contain a detailed description of your database, together with a brief rationale for your design. It should contain an explanation of how each of your SQL queries work, an explanation of how your PHP works and a list of all of the files that you have included with your distribution.

[10 marks]

User interface. This is the visual design, interaction design and usability. You might pick up a few marks for creating a beautiful user interface, but not many. What is more important is that your system is usable, in this case with minimal or ideally no training. We strongly advise you primarily examine Nielsens's 10 heuristics and secondarily look for existing best practices in web interface design to guide you.

[10 marks]