**CASE STUDY: Web-based IMS and CMS for managing an online estate agency** **SRN: 21027648**

As a freelance developer, you’ve been commissioned to undertake the following development project, made up of three main “Modules”: a “Web Portal”, an “IMS” and a “CMS”. More in detail:

1. The Web Portal is the official website of a family-run online estate agency and should be designed as a main framework from within which it would be possible to run their business and market their products and services. This would allow:
   1. the public to access an online show-room and look for available properties for sale, gather information for each property and contact the sales team for arranging a viewing; more advanced functionalities and facilities available to the public (for example, the possibility for customers to register their details and interest with the estate agency, to save searches on specific categories of properties or within certain locations, to chat with a sales-person, etc.) would be considered as an extra value added to the system and would be therefore suitably considered;
   2. staff of the estate agency to access (by using some secure login facilities) and run the following:
2. A database-driven web-based application to manage relevant details of the business, such as the properties for sale (including their purchases and sales), customers (both sellers and buyers), staff, events, services offered (e.g. property evaluation, independent mortgage brokerage, removals) etc. This application is the Information Management System (or IMS) and should be accessible and usable only by members of staff.
3. A Content Management System (or CMS), which, like the above IMS, should be manageable only by members of staff in order to populate and control what is published on the above-mentioned Web Portal. The CMS should support the easy management of the back-end of the online show-room, to include not only all needed details (e.g., photos, location, typology, price, etc.) of any property for sale, but also the possibility for staff to publish and advertise on the pages of the online show-room any relevant news for customers, like latest entries, special offers, etc.

The system overall could and should support several other functions (e.g. from administrators’ login to the possibility, for staff or customers, to choose from different graphic layouts of the online show-room), which you, as an expert, have been required to identify and/or suggest.

The software development part of the project is to be performed on any development environment and technology of your choice, as far as the final developed product (i.e., the system overall, made up of the above three modules) can be run from/on a (set of) server(s) you can control (likely, but not necessarily, on your own machine: for example, a local instance of Apache web server, MySQL database server, etc.), and can be used and tested via Mozilla Firefox, or any other browser which adheres to the same W3C standards Mozilla Firefox adheres to.

If you are unsure about which development technologies to use, we would recommend a simple “development platform” based on PHP and MySQL, ideally enhanced by a *PHP development framework[[1]](#footnote-1)* of your choice, and by any relevant client-side technology (HTML, CSS, JAVASCRIPT, etc.), again ideally enhanced by some available front-end facilities such as jQuery, Bootstrap, etc.: the easier way to start setting up your development environment is to adopt (and “adapt”) a WAMP stack, if you are working under MS Windows, a LAMP stack if you are working under Linux or a MAMP stack if you are working under Mac.

The development environment and the engineering process(es), which you are expected to set up, get familiar with and rely on, should include facilities such as a version control system (e.g., Git), unit testing (e.g., PHPUnit), web testing (e.g., Selenium) and bug tracking (e.g., Bugzilla). More details are discussed in the following sections of this document.

**CW2 – Coursework second delivery**

The second (and final) delivery [CW2] is worthy 60% of the assessment for this MSc module and should relate to the final release of the implemented system (all three modules should be included and fully working) and would consist of a second report and of a demonstration video in .avi or .mp4 format (possibly built with Camstudio or any other tool of student’s choice).

This report, expected to be of about 5,500 words, should be submitted in both .doc and .pdf format. As for the one produced for CW1, the report should be professionally formatted and should include a title page, contents page, executive summary, page numbers, appropriate headers that have your full name and university e-mail address, references, bibliography and any appendices if needed. Overall, it should appear as a one cohesive document. The main body of the report should be organised in five sections, as follows:

1. A first section, regarding testing of both the IMS and the CMS modules. As these two modules should work against a suitable number (between 20 and 30) of relevant functional requirements, for each requirement you should be able to identify at least one reasonably challenging test (e.g., how it would handle an attempted division by zero). All these tests should be provided into a list first and then (shortly) individually described. For say the three most relevant tests, one or two representative screenshots of the system should be provided for clarification purposes. It is the appropriateness and completeness of the testing exercise, as well as the successfulness of its implementation, that will be assessed [10%]
2. In the second section, one module only of the system (chosen by you) has to be thoroughly tested by using Bugzilla or a similar bug tracking system: relevant reports and a number of screenshots (between 10 and 15) should be included. By means of this section, you should provide evidence of how confident you are in using your bug tracking tool, and quality and relevance of that evidence is what you will be assessed against [10%]
3. All modules should be developed using an appropriate application of the Model-View-Controller (MVC) architectural pattern, or of any similar variation/evolution of this patterns (e.g., MVP, MVVM). Evidence of the application of the pattern should be given in a third section, in which you should explain how you (and/or the technology you’ve adopted for this project development) have pursued the MVC (or similar) pattern, also by referring to some shorts (possibly commented) pieces of code and to the system architecture as a whole, i.e. the way in which the files “making up the program” have been structurally organised. Larger chunks of code could be put into an appendix of the CW2 report and will not be considered within the 5,500 words limit. Assessment criteria would include quality and quantity of evidence provided, their relevance and how confidently the student has managed to master the MVC pattern (or one of its variations). [10%]
4. Software for this second and final delivery should be developed by relying on a Version Control System (VCS) of your choice. Evidence of this should be given by producing a fourth section of the document, in which the diverse versions of one file only (of students’ choice) have been retrieved from a VCS repository, with some accompanying screenshots of the VCS system related to the retrieving process. Again, quality and quantity of evidence, together with how the student has be able to demonstrate an achieved confidence with the technology and the approach, will be considered [10%]
5. A Post-mortem analysis and reflection of the overall development process, including what you have learnt, what you think you have done well, what you might have done better, and a plan of how you would implement further improvements, should be included in the fifth and last section of this report. Your post-mortem will be assessed against criteria such as depth of analysis, extent, an ability to self-reflect and critically assess your own learning achievements, and the quality by means of which you’ve developed this last portion of the CW2 submission [10%]

The video (to be named CW2.avi), no longer than ten / fifteen minutes, should provide evidence of how usable and rich in terms of functionalities and features is the system across its three modules. This part of the CW2 will be assessed against the quality of video in itself but also and above all against the actual features, the usability and the overall feel & look of the system [10%]. If the video is not commented live by you, it should at least include some subtitles and/or captions supporting its comprehensibility.

**MARKING SCHEME AND CRITERIA**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CW2** | Nothing of merit: 0-1 | Clear fail: 2-3 | Marginal fail: 4 | Clear pass: 5 | Good: 6 | Very good: 7 | Excellent / Outstanding: 8-10 |
| 1. **Testing** |  |  |  |  |  |  | **9** |
| Student has provided an outstanding section dedicated to testing, with an exhaustive and insightful explanation of the several types of testing exercises, approached and technologies (including Cypress) they pursued, all backed up by suitable evidence. Very well done! | | | | | | | |
| 1. **Bug Tracking System (and reports)** |  |  |  |  |  |  | **8** |
| Suitable evidence provided: Student adopted Backlog for this section of the assignment and full evidence of suitable usage has been provided  Relevance, appropriateness, confidence: Very high: Backlog has all relevant functionalities for tracking and managing bugs  Overall usage and professionalism shown: Excellent | | | | | | | |
| 1. **Application of the MVC architectural pattern** |  |  |  |  |  | **7** |  |
| Suitable evidence provided: Student has provided very good evidence both of pursuing the MVC pattern by a combination of approaches (mostly Strapi for the M and C component, and other JS solutions for the V component) and of understanding the pattern in general.  Relevance, appropriateness, confidence: Very good  Overall usage and professionalism shown: Very good | | | | | | | |
| 1. **Usage of a Concurrent Version System** |  |  |  |  |  |  | **8** |
| Suitable evidence provided: Student has provided suitable explanation and evidence of a comfortable, advanced usage of Github, including analytics features.  Relevance, appropriateness, confidence: Fully appropriate.  Overall usage and professionalism shown: Excellent | | | | | | | |
| 1. **Post-mortem** |  |  |  |  |  |  | **8** |
| Depth of analysis, extent, criticism: Student has provided a detailed and insightful post-mortem analysis of their achievements through this module  Self-reflection, learnt lessons: Excellent overall, as student has rightly argued how engineering software is so much more than just coding.  Report structure and overall readability: Easy to read, professionally formatted. | | | | | | | |
| 1. **Video (system demonstration)** |  |  |  |  |  |  | **9** |
| Demonstration coverage (functional & non-functional requirements, details): The video has covered in clear and extensive manner all relevant developed features the system comes with.  Quality of the system overall (analysis, client-complexity orientation, final achievements): The system has been developed in a very professional manner: all main functionalities are there, the system seems easy to use, accessible and reasonably responsive, and with a professional feel & look.  Video (quality of script plan, delivery, clarity): Very well designed and delivered overall. | | | | | | | |

**Final mark: …………49…………………. (max 60) Numeric grade (percentage): …82… (max 100%)**

**An indicative scale for quality of the work (scaled out of 10) which will be applied appropriately to each of the above items follows.**

Please note that, according to new assessment and award regulations, grades at MSc level should be interpreted as follows.

0-19: Nothing of merit

19-39: Clear fail

40-49: Marginal fail

50-59: Clear pass

60-69: Good

70-79: Very Good

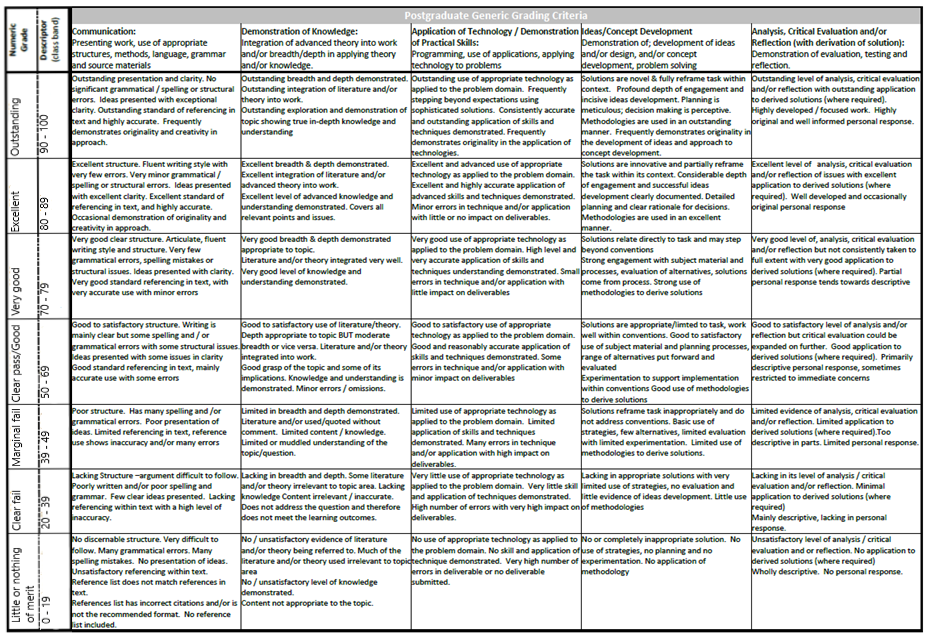
80-89: Excellent

90-100: Outstanding

Descriptors in the following table (see following page), therefore, should be interpreted accordingly (for example, where it says “Excellent structure”, it should be interpreted as “Very good structure”).

It should also be noted that:

1. Students have access to electronic journals via the UH VPN
2. Students have access to online books via Safari
3. The link to the LIS page on how to use the HARVARD Referencing System is in the Subject Toolkit for Computer Science, on Studynet (you need to login within StudyNet first): <http://www.studynet1.herts.ac.uk/ptl/common/LIS.nsf/lis/computerscience>
4. All resources for developing the system and any component needed for assessment purposes can and will to be discussed with relevant class discussions on Studynet.



1. See <https://www.google.co.uk/#q=PHP+development+framework> for a number of sources and relevant frameworks available on the web. [↑](#footnote-ref-1)