Keith Madden – Internet Development
Production Process Report
Due: 16/03/12

Abstract:

This document is intended to give an overview of the functionality and underlying technical features of the attached website. The website was built by Keith Madden for the purposes of submission for Dr Charlie Cullen's Internet Development module at the Digital Skills Academy. While some references will be made to the use & source of pre-existing code in this report, a refernce file has been included with a more concise documentation of said sources and reference points (That document, along with a copy of this pdf can be found in the site's 'docs' folder).

Report:

This website was built to emulate a college site, used by students and lecturers to view, edit, and register for modules. The role of an administrator was also considered, but was decided to have been catered to by the existing registration & module editing features, and as such no administrator centered features were developed. DIT was used as the stand-in college, and as such has it's logo displayed prominently on each page, as well as the inclusion of text from the actual www.dit.ie site in this site's Index page. Along with the index page, the site has 4 locations of functionality; The Student, Lecturer, Registration, and Help Desk pages. In the case of three of these pages, there is a login page included prior to access; two of which are password-based **(Note: All passwords for the site are the staff/student ID reversed. A student with an ID of 123 has a password of 321, and a Lecturer of ID 123001 has a password of 100321)**. These logins were deemed necessary due the the sensitive nature of the information on the ensuing pages. It is worth noting that these login pages implement password validation and Javascript re-directs for incorrect entries. The functionality and process of these sections is broken down and reviewed individually below.

Help Desk (form.php): The Help Desk page is a relatively straight-forward email form with built-in php validation. The form takes a users name, email, and their message and it's subject. The user's name is broken into first and last name, this is so as to return a more personalised message upon form completion. The validation program is based around the checkMail() function found in the lecture notes, which ensures a submitted email address is not using any invalid characters. The method implementing this function also checks to ensure the user has included a name. Differing from other similar validation methods, the form.php validation does not echo out a response to users when the check is run; instead it assigns a specific string value to a local variable depending on the check results. This variable is then echo'ed inside the body of the webpage, returning the output below the form. This was to allow for the function & method's placement in the <head> of the file, while maintaining an in-page output to the user. The inclusion of a fully operational email function was attempted, but due to SMTP issues had to be abandoned.

Students (studentH.php->student.php<-moduleDisp.php): The Students page allows for a user (presumably a student of the college) to login into their account to review the details of their modules. This viewing is achieved through the use of an AJAX request built around a <selection> drop down menu of the student's modules (identified by their unique module numbers). The selected value in the menu is sent by an XMLHttpRequest (using a function adapted from the W3schools site) to the moduleDisp.php page. This page queries the database and returns all information on the module into an html table.

Lecturers (lecturerH.php->lecturer.php->moduleEdit.php->update.php): The Lecturers' page is also login based, and allows for the editing of module information. Upon login, a lecturer is given a choice of the two modules they teach. These are presented as <submit> buttons. On selection of a

module button, the user is taken to the moduleEdit page, where the details of a module are reproduced inside of an editable form. Once altered, the module is updated through the use of the update.php file, which takes in the form information, and translates it into an associative array. The values of that array are then added to the database through the UPDATE query command.

Registration(moduleH.php->register.php->studentAdd.php): The Registration section was initially designed to cater to administrators, so that they could add new students to existing courses. However the relatively little information in the database allowing for course identification and editing, meant it was better to redesign the section to fit a student self-enrolling. As a result of this somewhat dual appeal, the login function was not included in the page, so that both course coordinators and would-be students can access it equally. To enter this section of the site, a user is required to have a course ID. Once this is given, the user is taken to a form, which they can fill out with their details; name, and module choices. Their student number is preassigned, and is held in a read-only input form. This number is gotten by adding the value of the first student number in the student table with the total count of 'student' entries/rows (If the student number (stu) count is equal to n, this gives us an ID of stu[0]+n=stu[n+1]). The course ID given in the previous page is carried over through a hidden input field in the form, allowing for it to be discretely carried across pages without the user needing to re-enter it each time. However, refreshing or reloading the page can cause a loss in this posted data, and as such an error check is in place on the preceding page to ensure a course ID has in fact been carried over. Once checks are completed, the studentAdd.php page pushes the data into an associative array (using the specially created associate() function, as in update.php), and sends it to the db using the INSERT INTO mysql query.

As can been seen, each section of the site implements a variety of techniques and functions to create a product that allows different types of users to access and interact with the collegeData database in varying ways. The site is designed to keep each of these processes as separate as possible, while at the same time allowing for cross-process activities (A newly-registered student is directed to the student section where they can review their module details, for example). In this way the site can appeal to a range of users simultaneously, through the application of php and it's associated technologies and techniques.