CMSC-433 Project 1: UMBC Advising

USER MANUAL

Group 1: Adam Ostrove, Jeremy Aguillon, Matt Law

Video: https://youtu.be/thGwWJTakjI

Table of Contents

Executive Summary

UMBCAdvising is a web application developed to allow a student to indicate prior coursework in order to both visualize remaining available classes and submit this information to an advisor. This document is divided into two main sections. The first describes the functionality of the application and how to use it from the student's perspective. The second describes the setup of the server and structure of the code and database for the sake of developers.

User Manual

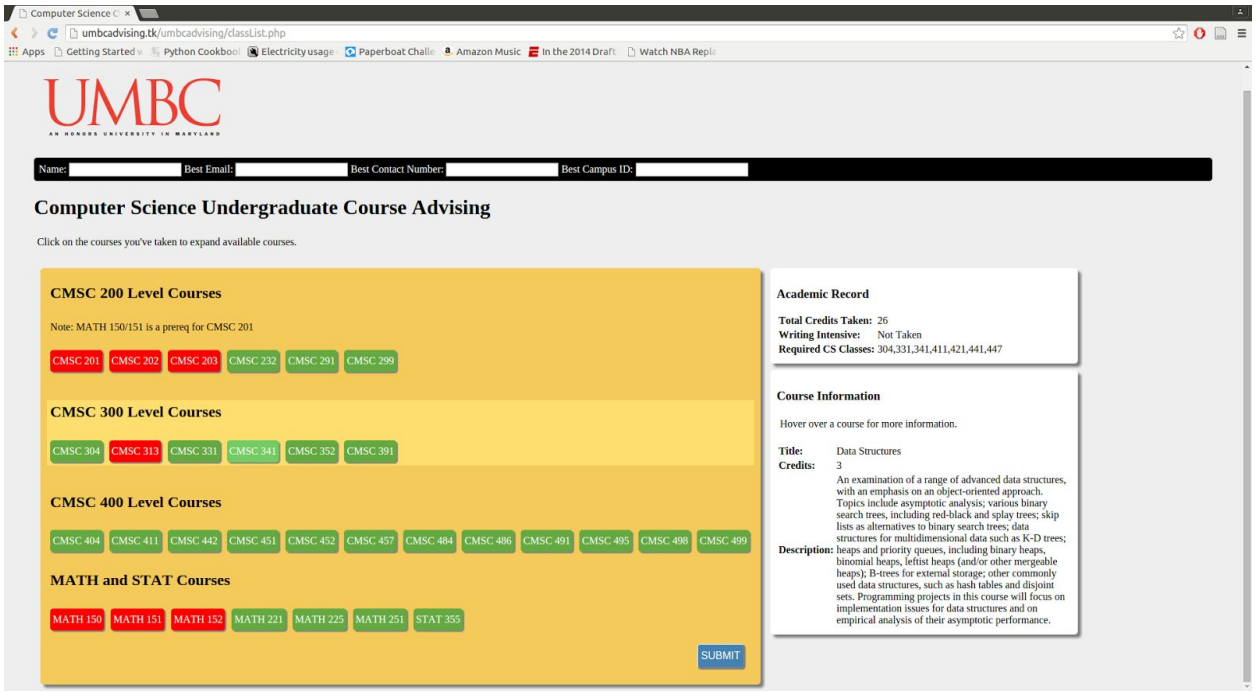

Figure 1: The UMBC Advising User Interface

Student Instructions for the Advising Interface

UMBCAdvising can be accessed at http://tinyurl.com/umbcadvising or http://umbcadvising.tk/umbcadvising/classList.php

.

To use the advising interface to determine your available classes, please do the following:

1.) Enter your first and last name, email address, contact number, and campus ID in the appropriate spaces in the user information bar at the top. This will help your advisor associate your classes with you. Please note that the email must be in the format of address@domain, the phone number should be a 10-digit number (5555555555), and the campus ID is two letters followed by 5 numbers.

2.) In the large, yellow course container you will see available courses categorized into 200, 300, and 400 CMSC courses and MATH/STAT courses. On the right side are two boxes. The upper box, titled “Academic Record”, tracks how many credits you have as you indicate that you have taken classes (see below), as well as whether you have fulfilled your writing intensive requirement and which of the required CS classes are still left to take. The bottom box displays information about any class that you hover over. In Figure 1, CMSC 341 is highlighted and its information is displayed in the lower box.

3.) Please click on courses that you have either taken or met equivalent requirements for. As you click on courses, they will open up eligibility for additional courses, which will appear in the appropriate section. For example, if you had AP credit that covered MATH 150, you would click on MATH 150, which would turn red to indicate that it has been taken. Since MATH 151 and CMSC 201 are dependent on MATH 150, clicking on MATH 150 will add these to classes to the interface.
4.) When you are finished, please hit the blue submit button. You may see an error message asking you to fix one of your user information fields. If this is the case, please hit back and re-complete the form. Otherwise, you should see a report indicating your inputs that will be passed on to your advisor.

Developer's Guide

Preliminaries (connecting to the server, etc.):

The application is run on a standard LAMP stack (Linux, Apache, MySQL, PHP), with much of the user interface handled client-side via raw JavaScript. The server is hosted on a Digital Ocean droplet at the following IP address:

IP Address: 45.55.201.73

For your convenience, a free .tk domain has been pointed to that address, **http://umbcadvising.tk** and the server can be ssh'ed at that ip or domain using root credentials of user=***root***, password=***Lupoli#433***

Please note that .tk domains, as free domains, can be suspended at any time; if the domain is not working, use the IP address instead.

All live code is located at **/var/www/html/umbcadvising**. The code is also hosted in a private git repo at github.com. Please contact law8@umbc.edu to request access to the repo if desired.

User Interfaces

The user interface is divided into three main containers: the classtree div where the courses are displayed to the user to select, and divs for academic status and course information to the right.

Course Display

PHP is used to automatically generate a hidden checkbox and associated label for each course offering. JavaScript is then used to parse a dictionary of courses, dependencies, associated credits, titles, and descriptionsnd descriptions into a custom data structure which is then iterated to determine which courses are available to the student and hide the rest.

Academic Status
This is simply a div with a table inside which is used to update the number of credits taken, whether a writing intensive course has been taken, and the CS requirements remaining. All three of these are tracked with global variables that are updated each time a course label is clicked.

Course Info
This is also a div with a table displaying course title, course credits, and course description. These are populated using JavaScript on a mouseover event associated with each label which queries the classes dictionary and populates the table with the relevant information.

Course Dependencies Data Structure

The course dependencies are represented via a simple graph, where each node is a course with a boolean value indicating whether it has been taken and two arrays of pointers, one consisting of prerequisite nodes for that course (the parent nodes), and one consisting of nodes that that course is a prerequisite for (the child nodes). On page initialization, a JS dictionary is parsed and used to initialize this data structure. The nodes each have member functions that check its parent functions to determine course availability, mark a node as taken, and mark a node as not taken, recursively clearing all its taken descendents as well. Every time a change is made to the data structure, JS is used to update the user view accordingly.

Database Design

The database is consists of a set of three phpMyAdmin/mySQL tables. The phpMyAdmin application can be accessed at **http://umbcadvising.tk/phpmyadmin** with **username:** root, **Password:** Lupoli#433. The Project1 database stores the tables for this project. The three tables are as follows:

courses: A list of all courses. Courses use their course id (i.e. cmsc201) as the primary key. The table also stores the course title and course credit value. Note that many independent studies courses have a variable credit value. These courses have a credit value in the database of 0.00 and it is up to the advisor to inform students of the credit value of any given independent studies course. A future update should create a column for class description into this table.

students: A list of all students. The student's campus id is used as the primary key. The table stores all of the personal information that a user enters, including campus id, name, email, and contact number.

coursestaken: A lookup table for courses that a student has taken. The primary key is simply an auto-incrementing integer in order to distinguish rows. The other columns are foreign keys of a student's campus id to course id. Each pair represents that a student with a certain id has taken a certain course. This many-to-many relationship is set up to CASCADE on delete or update. That is: if a student or course is deleted or updated, any records referring to that course or student will also be deleted or updated. Due to the many-to-many relationship established by this table, you can query it for all students who have taken a given course, as well as all courses that a given student has taken.

Input Validation and Submission

All form information is sent to processing over a post request. As much validation has been built into the form itself as possible. For example, contact info fields are required, and the email field will reject non-email entries. Name and email are reasonably validated by the html, so those two fields are simply escaped using php's htmlspecialchars() function. The other two fields are also escaped in such a way, but are also subject to regex validations. This ensures that only seven character campus ids are valid, and they must be two letters followed by four digits. Contact numbers must be exactly ten digits long to be considered valid. The list of courses taken is sent by a set of checkboxes. Since the value of these checkboxes can't tampered with, those values are all assumed valid.

Before any information is stored in the database, A check is made on the **students** table. If a student with the same id is discovered in that table as in the form submission, that user has already submitted a form and not be allowed to submit another until they are removed from the database.

On a successful validation, the student is inserted into the **students** table. Then, a record is created in the **coursestaken** table for each course they have taken mapping their student id to the course id. The user then receives a confirmation message as well as a summary of their course and credit history.

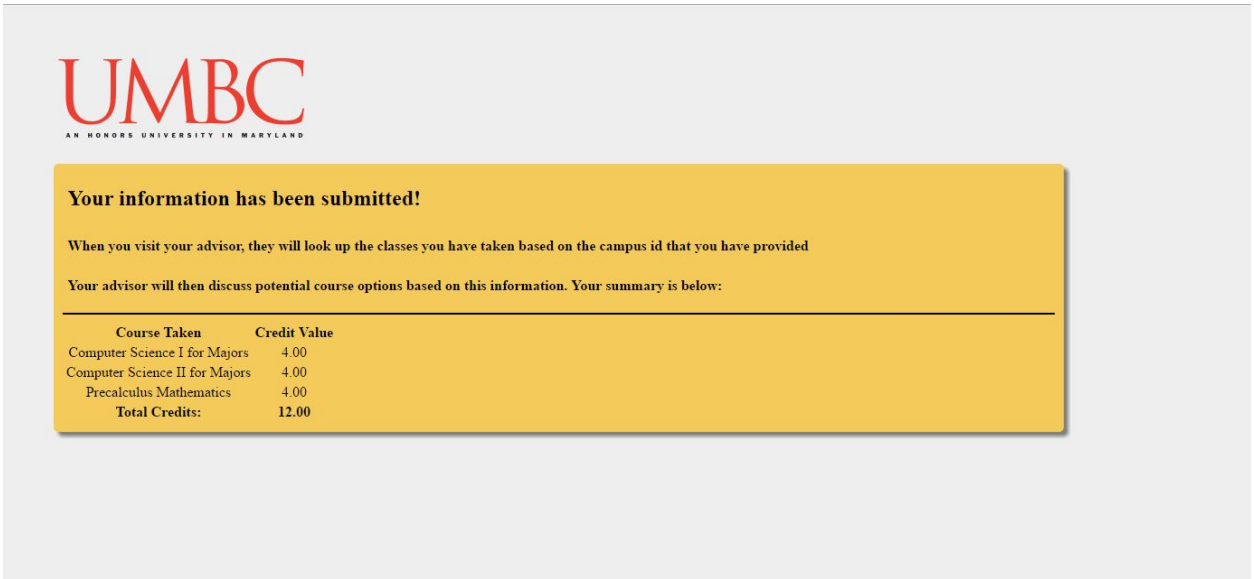

Figure 2: A form submission was successful

On a failed validation, the user receives a list of errors as well as a button to return to the form to recreate their submission. This page will also appear if a connection to the database cannot be established before the data insertion. Future versions should restore the values that the user originally entered.

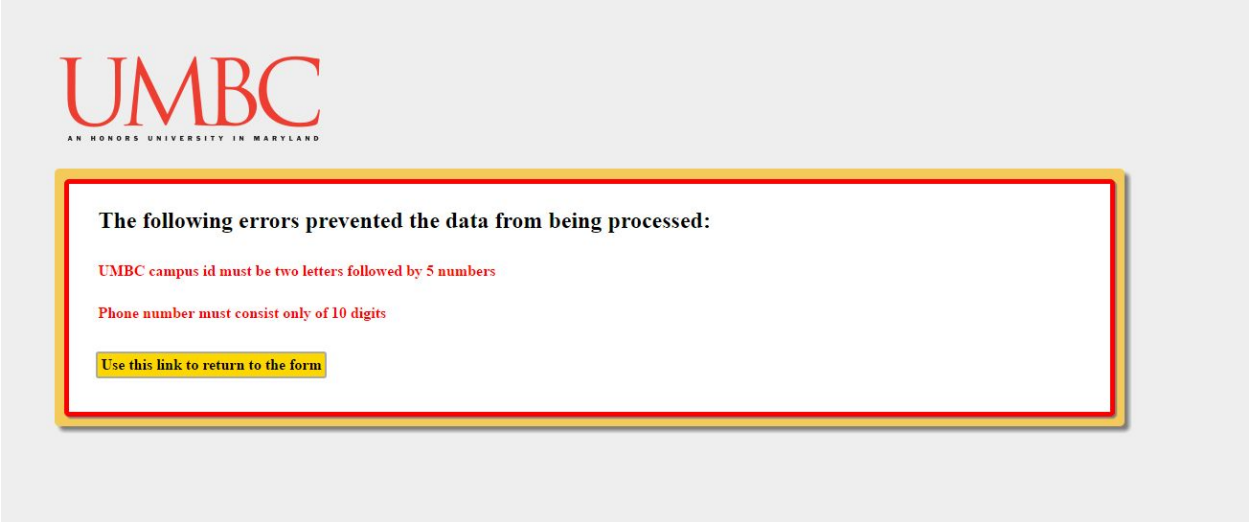

Figure 3: A submission error. In this case, campus id and contact number were invalid