Collegiate Times Grades

By: James O'Hara, Hang Lin

CS4624 Multimedia, Hypertext, and Information Access

Virginia Tech

Blacksburg, Va.

May 4, 2014

Client: Alex Koma, Managing Editor, Collegiate Times

Table of Contents

Section I. Executive Summary

The Collegiate Times grades database project is the public dissemination of the Virginia Tech grades database amassed by the Collegiate Times, the student newspaper of Virginia Tech, on their new website. The Collegiate Times has collected Virginia Tech grades data over the past decade using Freedom of Information Act requests and then provided this data to users for free on their website. However, the Collegiate Times has recently changed websites and there is no longer a way for the public to see or manipulate this data. Working with Alex Koma, managing editor of the Collegiate Times, this project provides a simple and maintenance-free solution for the Collegiate Times to put this data back in the public sphere by creating a web page that facilitates public access to the data.

The system is made up of a series of four dynamic dropdown menus that are filled using information from the database itself. The first dropdown menu is populated when the page loads and the remaining dropdowns populate once a selection is made, giving users some direction. Users search the database by specifying a subject and course number for a class at Virginia Tech. If they prefer a narrower set of results, they can also specify a year and semester for their chosen course. The program then provides the average grade point average and the percentage of students that received each letter grade for every year and semester section of the course requested. The data is both sortable and paginated. The project was programmed using HTML, PHP, SQL and Javascript.

The project is available at http://database.collegemedia.com/databases/grades/grades.html. Students can use this project to look up the average grades in courses they are thinking about taking during course registration and add/drop. Other potential users could use the data to research grading trends by year or to investigate the grade distribution in Virginia Tech courses. There are numerous other uses for the data and providing this data in a public setting has increased the information and knowledge available on course grades at Virginia Tech.

Section II. User's Manual

Part I. How to Use

The following is an illustrated step-by-step guide to using the Collegiate Times grades database as developed by this project.

To use the database, first open an internet browser and navigate to the Collegiate Times grades database page at http://database.collegemedia.com/databases/grades/grades.html.

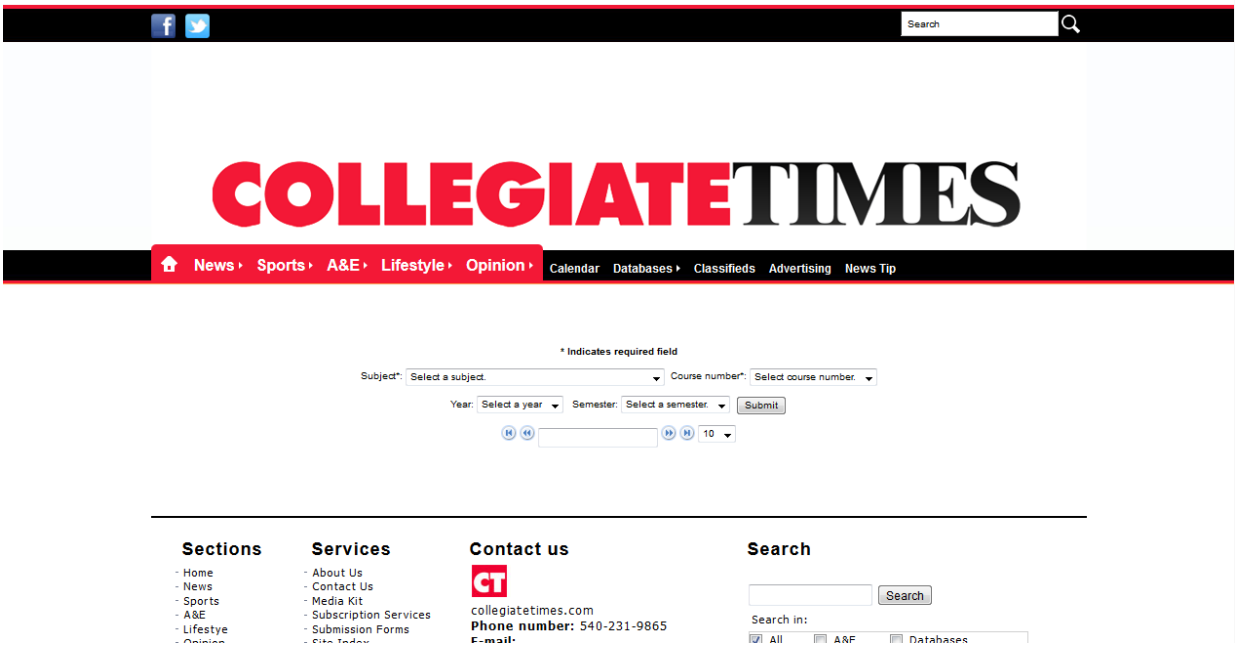

Figure 1. Collegiate Times grades database

To find information on a specific course there are two criteria that are required: the 'Subject' and the 'Course number'. First select the subject that the course belongs to and then select its course number. Only course numbers for classes within the subject are shown in the dropdown menu.

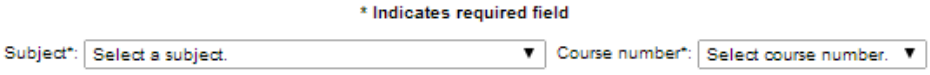

Figure 2. Required fields 'Subject' and 'Course number'

Once a subject and course number are selected, a user has the option of specifying the 'Year' and 'Semester'. To find the grades for a course in a specific year, or year and semester, use these optional dropdown menus. First select the year and then the semester of the desired course.

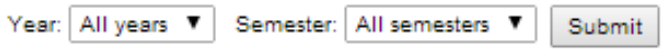

Figure 3. Optional fields 'Year' and 'Semester'.

The result of the selection will be displayed in a table. Each column displays different information in the following order: Subject, Course number, Year, Semester, GPA, %A, %B, %C, %D, %F. The GPA column shows the average grade point average for that specific class section and the columns after that show what percent of students got each letter grade.

Subject	Course number	Year	Semester	GPA	%A	%B	%C	%D	%F
German	2105	2001	Fall	2.83	33.3	38.1	14.3	9.5	4.8
German	2105	2002	Fall	3.39	51.7	37.9	10.3	0	0

Figure 4. Example of the table of results outputted.

Information can be sorted in ascending or descending order. To do this, click on the small black triangle (▲▼) next to the column title. If ▲ is next to the column title, then the data is currently sorted by that column and in ascending order. If ▼ is next to the column title, then the data is currently sorted by that column and in descending order. If both are shown, then the data is not being sorted by that column.

Subject	Course number	Year	Semester	GPA	%A	%B	%C	%D	%F
Computer Science	2114	2009	Fall	3.38	53.8	35.9	7.7	2.6	0
Computer Science	2114	2009	Fall	2.94	34.4	40.6	15.6	6.3	3.1
Computer Science	2114	2009	Fall	3.37	48.6	45.7	2.9	2.9	0
Computer Science	2114	2009	Fall	2.92	24.1	65.5	0	0	10.3
Computer Science	2114	2010	Fall	2.96	40.7	33.3	14.8	7.4	3.7
Computer Science	2114	2010	Fall	3.24	42.3	38.5	19.2	0	0
Computer Science	2114	2010	Fall	3.16	43.3	36.7	20	0	0

Figure 5. Sorting by year in ascending order.

When there are too many results to be shown on a single page, the results are split into multiple pages. To navigate through these pages, use the following buttons located at the bottom of the result table.

▌◄: navigates to the first page

◄◄: navigates to the previous page

►►: navigates to the next page

► ▌: navigates to the last page

Figure 6. Page navigation

The maximum number of results shown per page is initially set to 10. If desired, this can be changed by selecting a different value from the dropdown choices next to the page navigation. The table can display 10, 20, 30 or 40 records on one page.

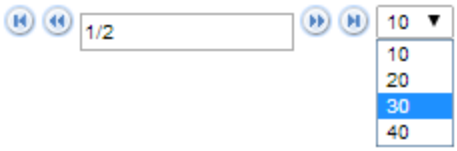

Figure 7. Results per page

Part II. Uses

This database has many uses depending on who the user is. The main use is for current and incoming students who want to research courses before choosing classes to take. Students could compare average grades in different courses or between sections of the same course. With this database, students can plan out their classes by looking through the information on courses they plan

on taking and picking the best section based on the results of their search. This gives students the most information possible before making their choice of class.

Another potential use for this database is for someone who wanted to analyze the grading trends for a single course across years and semesters. The analyst can find information for each individual course and determine a trend from the results of their query on the database. It can also determine if changes in the instructor, class time, or class type affects the grades. For example, the results could show that a certain course has a higher GPA when taught online, while another course has a higher GPA when taught in person. Instructors can test their teaching methods and course syllabi to see if they are improving the students' grades and use that information to plan for the future of the courses they teach.

There are a number of other potential uses that aren't the main ones detailed above. By providing an easy way for users to interact with the data and providing plenty of raw data, it is up to the users to determine the limits on how the data can be used. The future proofing of the code allows additional data to be added by the Collegiate Times without having to change the code, so users can be assured that they are working with the most up-to-date information.

Section III. Developer's Guide

Part I. Introduction

The main focus of the Collegiate Times grades database project was to create a simple way for users to access and query the database of Virginia Tech course grades collected by the Collegiate Times. To do so there were six design principals behind the project: prominently display data, must be easy to

understand, data must be sortable, data must be paginated, queries must be easy and productive, and it must be identifiable with the Collegiate Times brand. The development of the project followed these design principals and all future work should be based on these same principals.

The full project can be found on the server hosted by the Collegiate Times at database.collegemedia.com or in the project's VTechWorks entry in the grades.zip file. The data is stored in SQL tables and can be modified using myPHPAdmin. The data is collected in the database dbcolleg_Grades. The main table is "grades" which contains the grade information for courses specified by a subject id and course id. The subject a specific subject id refers to can be found in the "subjects" table and the same can be done for the course id and the "courses" table. The code for the web page that users' access can be found on the database.collegemedia.com server under the file path /public_HTML/databases/grades. For account information to access the full project, contact the Collegiate Times. Portions of the code and data are shown below to illustrate the development of the project and to assist in future developer needs.

The code making up the project is made up of one main HTML file and four PHP files that query the grades database to dynamically fill the dropdown menus. Additionally, the latest version of jquery, the jquery tablesorter extension, a CSS file to style the table that displays the data and a number of images for the tablesorter display are included to display the data in a more user friendly manner. Finally an index.HTML file is included to prevent users from accessing the folder these files are contained in without permission.

Part II. The Header Functions

The main file includes wrapper code to emulate the look of the Collegiate Times website around the main code that allows the user to query the database and view the results of those queries. The header block contains wrapper code as well as the initialization of the Javascript commands to fill the

dynamic dropdown menus and the tablesorter. The snippet of code below highlights the four methods used to create the dropdowns and to initialize the tablesorter. Line numbers are provided so that this portion of the code can be found easily in the actual file.

```
442     <script type="text/javascript" src="jquery-latest.js"></script>
443     <script type="text/javascript" src="jquery.tablesorter.js"></script>
444     <script type="text/javascript" src="jquery.tablesorter.pager.js"></script>
445   <script type="text/javascript">
446     $(function() {
447             $(document).ready(function() {
448                 $("#course").load("getter4.php");
449             });
450         });
451    </script>
452   <script type="text/javascript">
453     $(function() {
454             $("#course").change(function() {
455                 $("#class").load("getter.php?choice=" + $("#course").val());
456             });
457         });
458    </script>
459    <script type="text/javascript">
460     $(function() {
461             $("#class").change(function() {
462                 $("#year").load("getter2.php?choice=" + $("#course").val() + "&choice2=" + $("#class").val());
463             });
464         });
465    </script>
466     <script type="text/javascript">
467     $(function() {
468             $("#year").change(function() {
469                 $("#semester").load("getter3.php?choice=" + $("#course").val() + "&choice2=" + $("#class").val() + "&choice3=" + $("#year").val());
470             });
471         });
472    </script>
473    <script type="text/javascript">
474     $(document).ready(function() {
475         $("#myTable")
476             .tablesorter({widthFixed: true, widgets: ['zebra'], sortList: [[3, 1]]})
477             .tablesorterPager({container: $("#pager"), positionFixed: false});
478     });
479    </script>
```

Figure 8. Lines 442-479 of grades.HTML. The functions initialize the dynamic dropdowns and tablesorter.

The first function is called when the web page is accessed and uses the file getter4.PHP to query the “subjects” table of the grades database to populate the first dropdown that asks users to select a subject. The form and select that is populated by this function are defined later in the body of grades.HTML. A portion of getter4.PHP is shown below, with account information removed to protect the interests and security of the Collegiate Times.

```php
    //Select database
    $dbhandle = mysql_connect(localhost, $username, $password) or die("Unable to connect to MySQL");
    $selected = mysql_select_db($database, $dbhandle) or die("Could not select examples");

    //SQL Query
    $query = "SELECT DISTINCT `id`,`name` FROM `subjects` ORDER BY `name` ASC";

    //Do Query
    $result = mysql_query ($query);
    $o = "<option value='3'>Select a subject.</option>";
    while($row=mysql_fetch_array($result))
    {
        $o .= "<option value='" . $row['id'] . "'>" . $row['name'] . "</option>";
    }
    echo $o;
    mysql_close();
?>
```

Figure 9. Lines 8-24 of getter4.PHP. This populates the "subjects" dropdown.

A default value of three is given as this is the first record in the "courses" table of the grades database and prevents an error if the user does not make a selection. All other dropdowns have similar default values to return that first record. The code sets the value of each option equal to the id in the "subjects" table, this id is then referenced in the "grades" table in association with a specific course. The text displayed with the option is the name of the subject that is referred to by that subject id. To help in ease of understanding the results are sorted in alphabetical order by name and appear the same way in the dropdown.

Once a user selects a subject from the subjects dropdown, the second Javascript function is called and calls getter.PHP passing it the argument "choice," which is equal to the subject id of the subject the user just selected. The code in getter.PHP queries the "courses" table of dbcolleg_Grades and fills the courses dropdown with course numbers associated with the subject chosen. The value of the option is equal to the id in the course table and the text displayed is the actual course number. For example, a user could select the subject Computer Science and then the course number 4624, the associated ids in the table would be stored as the values.

```php
    //Select database
    $dbhandle = mysql_connect(localhost, $username, $password) or die("Unable to connect to MySQL");
    $selected = mysql_select_db($database, $dbhandle) or die("Could not select examples");
    $choice = mysql_real_escape_string($_GET['choice']);

    //SQL Query
    $query = "SELECT `id`, `number` FROM `courses` WHERE `subject_id` LIKE $choice ORDER BY `number` ASC";

    //Do Query
    $result = mysql_query ($query);
    while($row=mysql_fetch_array($result))
    {
        echo "<option value='" . $row['id'] . "'>" . $row['number'] . "</option>";
    }
    mysql_close();
?>
```

Figure 10. Lines 8-23 of getter.PHP. This code dynamically fills the course number dropdown.

These two dropdowns are the only ones required to submit a query to the grades database. However, a user can also specify a year and semester if they wish to. As with the course number the years dropdown is dynamically filled using getter2.PHP. This file queries the grades table for years where the subject and course id match the ones requested by the user. If a year is chosen, then the fourth Javascript function calls getter3.PHP which queries the grades table for semesters where the subject id, course id and year matches the ones requested by the user. The final Javascript function in the header tag of grades.HTML initializes the jquery tablesorter extension. This extension will present the data in a paginated and sortable fashion, initially sorting the data displayed by year and allowing users to sort it by any way they choose. Tablesorter displays the data in a more visually appealing way and satisfies the design requirements for paginated and sortable data.

Part III. The Body Section

The HTML form itself is inside the body tag starting on line 988 of grades.HTML. The form is sparse and is made up of four selects: subject, course number, year and semester, along with a button to submit the form. Each select is made up of one default option that informs the user that they need to select an option to query the database in the way they wish. An asterisk is placed next to the two

required fields and the user is informed of such.

```
988   <body>
989     <form name="myform" action="" method="GET" class = "margin">
990       <fieldset>
991      <p><h6>* Indicates required field </h6></p>
992      <p>Subject*: <select id="course" name="course">
993                 <option value="3">Select a subject</option>
994               </select>
995       Course number*: <select id="class" name="class">
996               <option value="3954">Select course number.</option>
997               </select></p>
998       <p>Year: <select id="year" name="year">
999                <option value="">Select a year</option>
1000              </select>
1001        Semester: <select id="semester" name="semester">
1002               <option value="">Select a semester.</option></select>
1003      <input type="submit" name="submit_form" value="Submit" /></p>
1004      </fieldset>
1005    </form>
```

Figure 11. Lines 988-1005 of grades.HTML. The HTML form the user uses to query the grades database.

The next part of the body is the PHP code that handles when a user submits a form. To support selecting all years or all semesters these options are left blank by default and if statements are used to check if the user selected a specific year and semester or not. To help users confirm the options they selected are what is being displayed the “subjects” table is queried to pull the subject name selected and store it in the $subject variable and the courses table is queried to pull the course number selected and store it in the $course variable. For each possible year/semester combination the choice the user made is echoed and a query is built using the options specified by the user. The database is then queried

using the user built query and the result matrix is then stored in the $result variable.

```
1006
1007  <?php
1008  if($_GET['submit_form'] == "Submit"){
1009
1010  //Get subject name
1011  $subjectquery = "SELECT DISTINCT `name` FROM `subjects` WHERE `id` LIKE '".$_GET["course"]."'";
1012  $subjectresult = $db->query($subjectquery) or die($mysqli->error.__LINE__);
1013  $subject = $subjectresult->fetch_assoc();
1014  //Get course number
1015  $coursequery = "SELECT DISTINCT `number` FROM `courses` WHERE `id` LIKE '".$_GET["class"]."'";
1016  $courseresult = $db->query($coursequery) or die($mysqli->error.__LINE__);
1017  $course = $courseresult->fetch_assoc();
1018
1019  if(empty($_GET['year'])){
1020      echo "<h3>Displaying results for ".$subject["name"]." ".$course["number"].".</h3>";
1021      $query = "SELECT year, semester, gpa, percent_as, percent_bs, percent_cs, percent_ds, percent_fs FROM grades WHERE course_id LIKE '".$_GET["class"]
          ."' AND subject_id LIKE '".$_GET["course"]."'";
1022  }
1023  else {
1024    if(empty($_GET['semester'])){
1025      echo "<h3>Displaying results for ".$subject["name"]." ".$course["number"]." in ".$_GET["year"].".</h3>";
1026      $query = "SELECT year, semester, gpa, percent_as, percent_bs, percent_cs, percent_ds, percent_fs FROM grades WHERE course_id LIKE '".$_GET["class"]
          ."' AND year LIKE '".$_GET["year"]."' AND subject_id LIKE '".$_GET["course"]."'";
1027    }
1028    else{
1029      echo "<h3>Displaying results for ".$subject["name"]." ".$course["number"]." in ".$_GET["semester"]." ".$_GET["year"].".</h3>";
1030      $query = "SELECT year, semester, gpa, percent_as, percent_bs, percent_cs, percent_ds, percent_fs FROM grades WHERE course_id LIKE '".$_GET["class"]
          ."' AND year LIKE '".$_GET["year"]."' AND semester LIKE '".$_GET["semester"]."' AND subject_id LIKE '".$_GET["course"]."'";
1031    }
1032
1033  }
1034
1035  $result = $db->query($query, MYSQLI_STORE_RESULT);
```

Figure 12. Lines 1007-1035 of grades.HTML. The query is built using the information provided by the user.

The variable $o is then used to build the table step-by-step and is initialized with the header of the table. Then $result is looped through row by row and the data is added to $o. Once all of the queried data has been added to the result, the closing table tags are added to $o and it is echoed back to the page, where tablesorter presents it in a user friendly manner. Finally, the connection to the database is closed.

```
1036  $o = '<table id="myTable" class = "tablesorter"><thead><tr><th>Subject</th><th>Course number</th><th>Year</th><th>Semester</th><th>GPA</th><th>%A</th><
      th>%B</th><th>%C</th><th>%D</th><th>%F</th></tr></thead><tbody>';
1037
1038  while(list($year, $semester, $gpa, $percent_as, $percent_bs, $percent_cs, $percent_ds, $percent_fs) = $result->fetch_row()) {
1039      $o .= '<tr><td>'.$subject["name"].'</td><td>'.$course["number"].'</td><td>'.$year.'</td><td>'.$semester.'</td><td>'.$gpa.'</td><td>'.$percent_as.'<
          /td><td>'.$percent_bs.'</td><td>'.$percent_cs.'</td><td>'.$percent_ds.'</td><td>'.$percent_fs.'</td></tr>';
1040  }
1041
1042  $o .= '</tbody></table>';
1043
1044  echo $o;
1045  $db->close;
1046  }
1047  ?>
```

Figure 13. Lines 1036-1046 of grades.HTML. This outputs the results of the query to the page as a table.

The final part of the code in the body defines the pager part of the tablesorter extension. This supports the pagination of the data and places the images that allow the users to navigate the paginated

table. This is where the images detailed in the introduction (first.png, prev.png, next.png, last.png) of this section are used.

```
1048  <div id="pager" class="pager">
1049      <form>
1050          <img src="first.png" class="first"/>
1051          <img src="prev.png" class="prev"/>
1052          <input type="text" class="pagedisplay"/>
1053          <img src="next.png" class="next"/>
1054          <img src="last.png" class="last"/>
1055          <select class="pagesize">
1056              <option selected="selected"  value="10">10</option>
1057              <option value="20">20</option>
1058              <option value="30">30</option>
1059              <option  value="40">40</option>
1060          </select>
1061      </form><br><br><br><br><br>
1062  </div>
1063  </body>
```

Figure 14. Lines 1048-1063 of grades.HTML. The pagination HTML div tag.

Part IV. Conclusion

The code was developed so that it will be future proof. Any changes or additions made to the data as defined in dbcolleg_Grades will be reflected on the page immediately as every dropdown used is dynamically filled using the data itself. The tablesorter jquery extension displays the table in a visually appealing, sortable and paginated manner. Regardless of the amount of output from a specific query, the table will be sortable and paginated on the user side. This improves user experience and allows for a greater flow of information. All future development should work within the framework provided and should keep the amount of maintenance required of the Collegiate Times to a minimum.

Section IV. Lessons Learned

Part I. Introduction

The Collegiate Times grades database project has been developed over the entire spring semester of the 2014 school year at Virginia Tech. Working in conjunction with the staff at the Collegiate Times and with Professor Fox the project created a page that allowed users to query the grades data amassed by the Collegiate Times. The data is sortable and paginated and the query form makes user interaction easy. The code is also created to reduce the amount of future maintenance needed by the Collegiate Times. Over the course of the project a number of problems were encountered and meetings held with Alex Koma, managing editor of the Collegiate Times to adapt to the problems and to update on the progress of the project. The project imparted valuable lessons on working on a client-based project and the work schedule expected in developing a new piece of hypertext and multimedia software.

The main concern was balancing the knowledge and desires of the computer scientists developing the page and the Collegiate Times' expectations and desires for how the final product would work. Working with a hands-on client presented both benefits and additional problems that needed to be dealt with as the project progressed. However, by the end of the project both parties were happy with the final product. Additionally, it was clear that the final product benefited from so many unique voices providing input as the design, development and evaluation process was completed.

Part II. Timeline/Schedule

Work on the project began the first week of February when the SQL files that populated the grades database on the original Collegiate Times website were downloaded and uploaded to the new server located at database.collegemedia.com using myPHPAdmin. Research into past and prospective

future user and Collegiate Times needs was also conducted at this time and six design guidelines were developed from this research.

By the second week of February the initial design based on these design guidelines was created using wireframes and the client was briefed on the design choices. During this meeting the client requested a few changes to the way the user filled out the form to query the databases. These changes were agreed to by the development team and a new design specification was created and final approval was given by the client.

The development team then used these wireframes to create an initial prototype of the full web page. The initial prototype supported querying by course number, course name, year and semester. The results of the query were then outputted using the ids in the “grades” table and was paginated and made sortable by the tablesorter jquery extension. The initial prototype was a standalone and did not have any notable links to the Collegiate Times website yet.

The team then presented the work completed up to this point and the work yet to be completed to the CS 4624 class the first week of March. Since the client was unable to attend the presentation a separate meeting was set up with just the client where the team gave the same progress report. Input that was provided by the professor and class after the presentation was relayed to the client and a discussion on the next steps in the development stage was held. Some changes discussed were implemented and an updated design was approved. By the end of March, the final prototype was completed.

At the end of March and the beginning of April usability inspections were performed on the final prototype with experts from both the Collegiate Times and the field of computer science. The Collegiate Times provided three experts with knowledge of the previous implementation, user needs and the needs of the Collegiate Times staff. These experts suggested that the dropdowns be sorted, that required information be noted as such and to provide an explanation of the database. Four computer

science majors were then asked to evaluate the page. Suggestions from the evaluation included preventing users from injecting data by using dropdowns with preset values for all inputs, limiting query size and to report back what the user's query was before displaying the data.

The results were tabulated and summarized by the evaluation manager and presented to the client with the development team's recommendations. After the meeting the team decided to implement all of the main suggested changes excluding the explanation of the database, which was determined to fit better on the database splash page where users could choose between the salaries database and grades database. The final changes were implemented and completed by the end of April and a link was provided to the Collegiate Times to be used at their discretion. The final report and presentation was completed on May 1.

Part III. Problems and Solutions

A number of problems were encountered in the development process, mainly with the organization of the file system where the Collegiate Times kept the backups of their SQL files. Other problems included the dynamic dropdowns not populating, displaying the subject name and course number rather than the subject id and course id and getting the tablesorter code to execute correctly. While these problems created delays in the development process, the solutions to them led to a better final product.

On two different occasions the development team found data that they initially did not know existed and suspect that there is at least one more SQL file out there that has instructor data that could not be located. The late discoveries of data changed the design, but in the end created a better final product. Initially the file with course information was located in a different location than the grades file, once this file was located it was possible to specify by a specific course number and course name, rather than just the generic ids that were found in the grades file. On the second occasion, the subject table

was found. This significantly improved the way users queried the database, as the number of subjects was much lower than the number of course numbers. This allowed for the narrowing aspect the initial design wanted as users entered more information and prevented the process from being cluttered with too many options.

There was one issue with missing data that could not be resolved during the development of this project. The subjects table has a number of subject ids with no corresponding subject names. This created a problem as it meant the first spots of the subjects dropdown menu are blank. The client was informed of the issue and is working on a solution; however it was not resolved by the end of the project development. The way the code was designed however, once those missing subject names are determined, or the offending records are removed from the database, the front end will be updated automatically. No further development work will have to be done by the Collegiate Times in response to this issue.

The other problems were more insidious. The way the queries were built in the getter PHP files was originally incorrect. The queries required that each variable being checked be placed in diagonal single quotes, while the query had straight single quotes. Once this issue was discovered and fixed, the next problem was with passing and receiving the variables correctly. This was just a case of developer error and was quickly remedied. The problem itself was not serious; however the delays it created were not beneficial to the project as a whole.

Because the option values were set to the subject id and course id instead of the subject name and course number as selected by the user, displaying the query they made and putting that information into the table meant displaying internal ids that did not make sense to the user. To rectify this problem the development team queried the database with the subject id and course id specified by the user after they submit the form and storing those values in variables. Those variables are then used in the displays instead of the option values, thus showing more relevant information to the user.

The final issue was derived from the process of testing the prototype on the public HTML server and then removing it when that testing was completed to prevent users from stumbling upon it before it was completed. On one occasion the process of re-uploading the project had an error and the tablesorter jquery library was not fully uploaded, causing an error when it attempted to execute it in grades.HTML. After some code testing the problem was discovered and fixed and the only harm done was delays to development time.

While none of the problems had a lasting effect on the final product, they were a series of nuisances that delayed development. They also provided valuable lessons to the development team in proper documentation and file organization, how to build correct SQL queries, using the HTML form and SQL queries to the developer's advantage and the process of using an FTP client with a web server. The development process would not have been complete without some problems and these provided an excellent learning opportunity to the development team.

Section V. Acknowledgements

The project team would like to thank the Collegiate Times and in particular managing editor Alex Koma for allowing us to undertake this project and the incredible amount of feedback and guidance they provided during the entire process. This project could not have been completed without the constant dialogue we received from them. The team would also like to thank Dr. Fox for helping them develop a solid project proposal and for making sure the project advanced at a rate needed to be completed. For more information on the project, the Collegiate Times' managing editor can be reached by email at managingeditor@collegiatetimes.com or in the Collegiate Times office in Squires Student Center, room 365.