GLOBAL JOURNAL OF PURE AND APPLIED SCIENCES VOL. 17, NO.4, 2011: 487-502
 ISSN 1118-0579
www.globaljournalseries.com, Email: info@globaljournalseries.com

A SOFTWARE APPLICATION FOR THE PROCESSING OF STUDENTS RESULTS

E. O. UKEM AND E.O. ONOYOM-ITA

(Received 26 October 2010; Revision Accepted 30 September 2011)

ABSTRACT

The processing of students' results is found to be rather tedious, especially when carried out manually, and when the number of students is large. It is time-consuming and error prone. The process becomes a lot easier and much more accurate when automated, because the computer is capable of accepting and storing raw data, processing it, and storing the results until needed. In this work, a computer software application was developed to facilitate the automated processing of the results. The software was developed using the Waterfall Software Life Cycle model. It was designed in the form of a database capable of running on a network. It has four sessions, namely, the Super Administrator, the Staff/Administrator, the Staff, and the Guest. Software tools employed included MYSQL Relationship Database Management System, Dreamweaver Integrated Development Environment, PHP, and JavaScript. When tested, the developed software worked well and produced expected results. Screenshots of some of the outputs are included here.

KEYWORDS: Students' results, automated processing, software application, database, output.

INTRODUCTION

All human activities have witnessed the application of one type of machine or the other, and these machines have performed well in their areas of application. One of such machines that have experienced a rapid development is the computer. The computer, in the area of information technology, among other areas, is capable of accepting and storing raw data, processing the data according to a set of instructions, and storing the results of the processed data for future use. It is a very flexible and versatile tool, hence its adaptability to almost all areas of human endeavour. All that is needed for the machine to perform in a different area is to change the processing instructions, called commands, and it would revert its operational mode to meet the desired purpose. Its global acceptance is also attributable to its ability to handle a very large volume of complex routine jobs, often at very high speed and accuracy. The objective of this paper is to present an instance of the application of this tool to the processing of the students results.

At present, the processing of students results in the University of Calabar is largely manual in nature, with its attendant short-comings. The process is time-consuming and prone to errors. In the Departments, examination results are always published late, wrong grades are sometimes entered, and a student's grade point average may be wrongly computed, leading to wrong conclusions on class of degree awarded. All this can bring about a lot of frustration to all concerned. The software application in this article is intended to bring relieve by providing for timely and accurate processing of students results using the processing power of the computer.

BACKGROUND

The University of Calabar, Nigeria, in line with world-wide trends, is making effort to computerize or automate its information system. The processing of students results is one of such areas that are only partially automated at present. Many Departments in the University, for example, use Microsoft Excel spreadsheet to process results, and this cannot be said to be full computerization, as only a few of Excel's capabilities are put to use, even though Excel provides prowess that gives the programmer the power to control all aspects of the program and the tool for the creation of custom solutions for data manipulation and analysis (Ekpenyong, 2008). The students grading system is the Five-Point Grading System, establishment by the Nigerian Universities Commission (NUC) in 1989. At present all universities in Nigeria use this system, which is also called the "Carry Over" system because a student is allowed to attempt a course a maximum of three times or carry over a maximum of two times before a final "Fail" grade is recorded if he/she still does not pass the course. A few courses, such as the General Studies and Communication Skills (or GSS) courses, are exempted from this rule. Such courses must be passed for a student to graduate. The NUC five-point grading system is shown in Table 1 below.

E. O. Ukem, Department of Physics, University of Calabar, Calabar
E.O. Onoyom-Ita, Department of Physics, University of Calabar, Calabar

Table 1: The NUC Five-Point Grading System

SCORE	GRADE	GRADE POINT	REMARK
70-100	A	5	Excellent
60-69	B	4	Very Good
50-59	C	3	Good
45-49	D	2	Average
40-44	E	1	Pass
00-39	F	0	Fail

Each course offered in a degree programme is allocated a number of credit hours. Because the courses vary in their needs and scope, some are allocated greater credit hours than others. The measure of performance of a student in any course is given by the grade-points scored in that course. The grade-points scored by a student in any course are obtained by multiplying the value of the grade by the credit hours of the course. When the grade-points of each of the courses offered are summed up, the total grade-points are obtained. The Grade-Point Average (GPA) is computed by dividing total grade-points by the sum of credit hours of all the courses offered in that period. Thus:

$$GPA = \frac{\text{Total Grade-Points of the courses offered in that year}}{\text{Summation of the credit hours of the courses}}$$

The Cumulative Grade-Point Average (CGPA) in any year is obtained by dividing the cumulative sum of the total grade-points over the years by the cumulative sum of the credit hours over the same period. Thus,

$$CGPA = \frac{\text{Cumulative sum of Total Grade-Points of the courses from Year One to the Present Year}}{\text{Cumulative sum of the Credit Hours of the courses from Year One to the Present Year}}$$

The CGPA is a very important measure, as this is what determines whether a student can move forward or be made to repeat a year or even withdraw totally. The final CGPA determines the class of degree awarded to the candidate on completion of the the programme.

Calculating and tracking CGPA for each student is rather tedious when done manually, and is prone to error. With a computer the task becomes much easier, faster, and more accurate, provided a suitable software application is used. With the appropriate software in place, the system would simply capture raw scores as entered by individual lecturers for various students in the different courses, and then process.

MATERIALS AND METHODS

The application was developed using the Waterfall software life-cycle model, which is called the traditional model because it was the first widely used software development life cycle. It was put forward about thirty years ago and has since then been successfully used by a wide variety of respected software organizations (Stephen, 1992). It is part of Structured Software Engineering or the Structured Paradigm, which is regarded as the classical approach to software engineering. The Structured Paradigm is the older of the two earliest approaches to formal software engineering. (The other approach is Object-Oriented Software Engineering). The Waterfall model is a document driven model; that is, the previous phase is carefully tested and reviewed before proceeding to the next phase, and each accepted phase is duly signed off (Curtis, 1998). The application was developed in the form of a database capable of running on a network (Onoyom-Ita, 2008). The decision to implement it in the form of a database was informed by the realization that various types of data would need to be held, and a database has advantages over other forms of file systems. Such advantages include data consistency, data integration, data sharing, data independence, data control and minimal data redundancy (Tata, 1995). Network capabilities were included to enable various users of the application to access it from their offices if attached to a network, rather than go to the computer room for every transaction.

The materials for this work are:

- Adobe Dreamweaver CS5 2007 edition, an Integrated Development Environment (IDE). This was used to create the Graphic User Interface (GUI) and to write the codes. The GUI was preferred to command line operation because it makes for greater ease of use and enhances user-friendliness of the application.
- MYSQL Server, a Relational Database Management System (RDBMS), which was used to create the database tables and data. The Relational system has been the most commonly used of all databases, and has various features that make it simple yet elegant (Williams and Sawyer, 2003).
- Personal Home Page Pre-Processor (PHP), used to communicate with and manipulate the database. PHP is a server-side scripting language that is used to develop web based applications that interact with relational database management systems (Jay & Brad, 2007).
- JavaScript, also used to communicate with and manipulate the database. JavaScript is a scripting language that is used in developing client-based and server-based applications (Goodman, 2003).

The application was designed to have four distinct sessions administered by the Super Administrator, the Staff/Administrator, the Staff, and the

Guest, respectively. A Head of Department is authenticated as a Super Administrator, and is expected to perform the following functions:

- Register students in the Department
- Register staff in the Department
- Assign an examinations officer
- Register course for a particular year
- Assign course coordinators to various courses

The Staff/Administrator has the authentication of the examinations officer of a unit and should perform the functions:

- Process students' results in his unit. This includes calculating the GPA and CGPA.
- View all the students' result in his unit

The Staff should have the authentication of a course coordinator and be able to enter students' grades for the courses he coordinates as well as edit the grades he entered.

The Super Administrator and Staff/Administrator must, in the first instance, be registered staff (that is, must have Staff authentication) before they can have the additional authentication. The architectural design of the application is illustrated in the block diagram of Fig. 1. The architectural design defines the software in terms of the major components and interfaces (Mazza & Depablo, 1994).

The application has provision for four categories of users, hence the detailed design is divided into four sections accordingly, namely Guest, Staff, Staff/Administrator, and Super Administrator. The detailed design is the process of defining the lower-level components, modules, and interfaces. The physical model outlined in the architectural design phase is extended to produce a structured set of components specification that is consistent, complete, and coherent. The detailed design phase is sometimes also called the implementation phase, because the developer writes the codes, documents, and tests the actual software (Pressman, 2001). Figure 2 is a block diagram of the detailed design of the Staff section, as an illustration. Other sections have similar structures (but not reproduced here).

A reasonable measure of security for the system is provided by the use of passwords for each authorized user. Additionally, limited access is granted to different users depending on their roles in the system. Some have read-only access and others read/write access as may be required. To further enhance security, some fields in the database are made to be display-only (that is, non-editable) and so cannot be altered by any user.

Coding of the application produced around 4,000 lines of code, some of which are shown here in the Appendix. Only a few snippets are reproduced here, as the whole code listing would take up too much space. The first code snippet shown is the portion that creates the database with its tables, and populates some of the tables as desired (see Appendix).

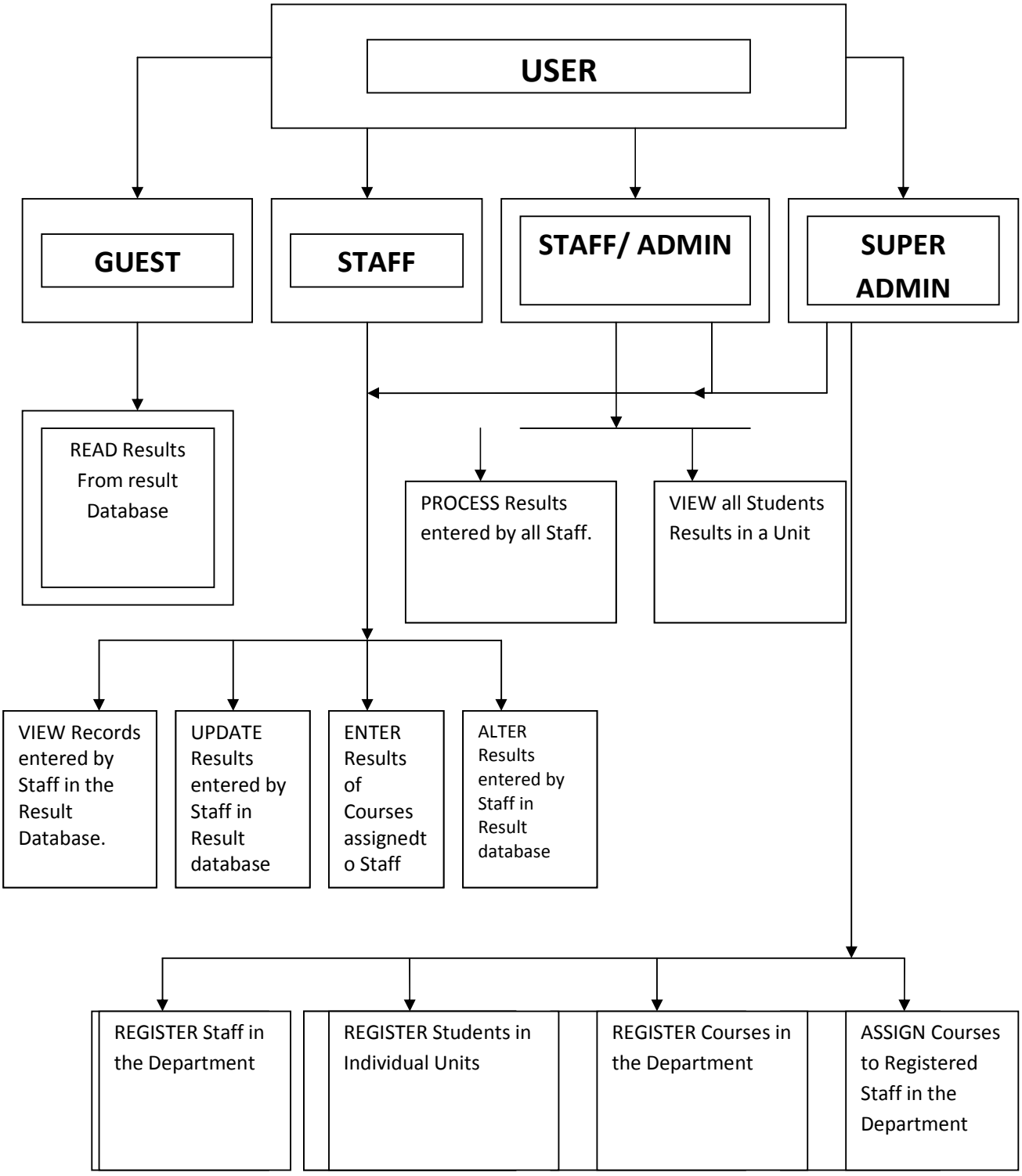

Fig. 1: Block diagram of the Architectural Design of the software application

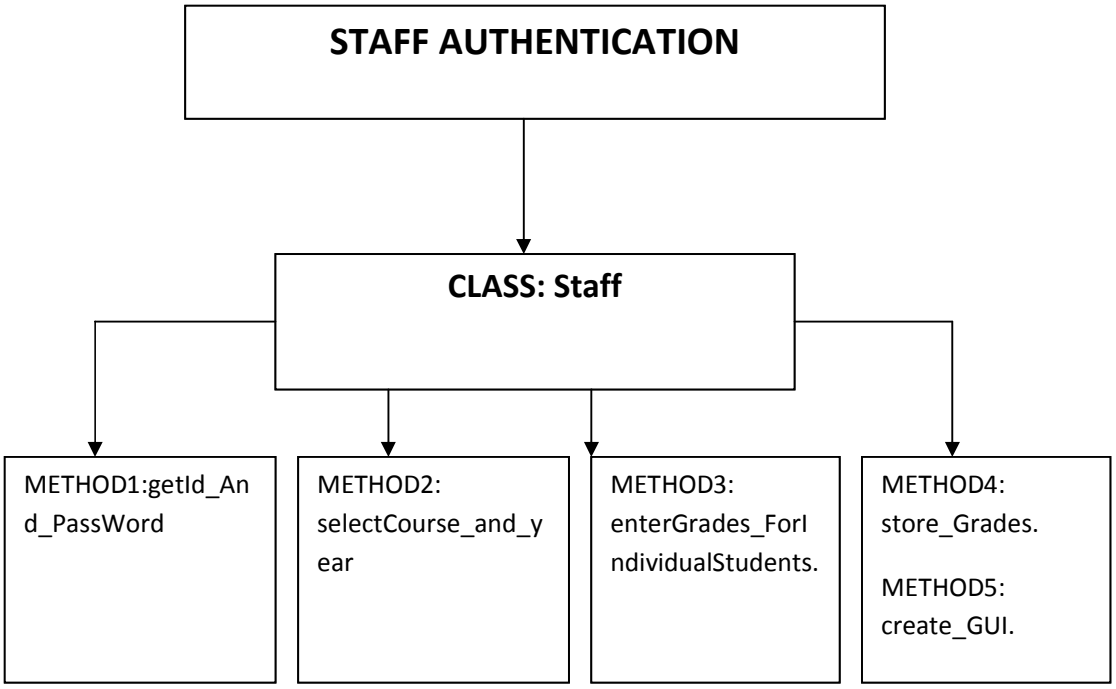

Fig. 2: Block Diagram of Detailed Design of the Staff User section of the application

RESULTS AND DISCUSSIONS

The developed software application was run on the system and found to operate as expected. The opening screen (or home page) appears as shown in Fig. 3. The Super Administrator logs in by typing his Username and Password. Figure 4 shows the Super Administrator screen. Similarly, figures 5 and 6 show the Exams Officer's and Course Coordinator's screens respectively. A sample output is shown in Figure 7. This shows the result of a fictitious student in the 200 level (year 2). The result shows the GPA for each year and the CGPA for the entire period. The application does this for each student and for each year of study. Figure 8 shows the results of this same fictitious student in the final year (500-level for the Electronics and Computer Technology programme of the Department).

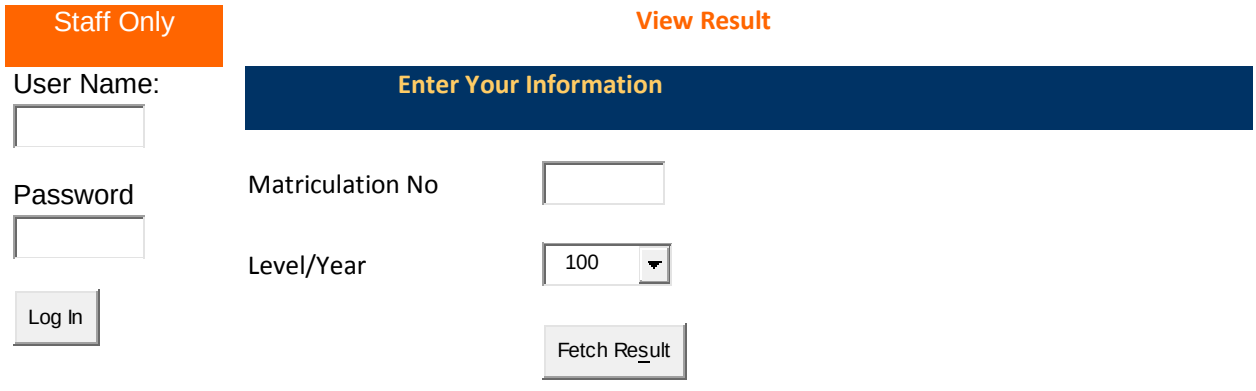

Fig. 3: The Opening Screen of the Software Application

The application provides a convenient approach to students' results processing. It is reasonably secure in that no unauthorized person can gain access to alter the data. In particular, students have no access to tamper with their grades, and not even their patrons (on the staff) can effect unauthorized changes. The information obtained from the system has a high degree of accuracy, because all computations are automated. Errors may only occur during data entry (keying in of the raw scores) and efforts can be made to keep them to the barest minimum. When all data has been entered, computations are carried out very speedily by the system, and required information is available immediately. Only data entry, being a human-dependent activity, may delay the process.

Fig. 4: The Head of Department's (Super Administrator's) screen (or environment)

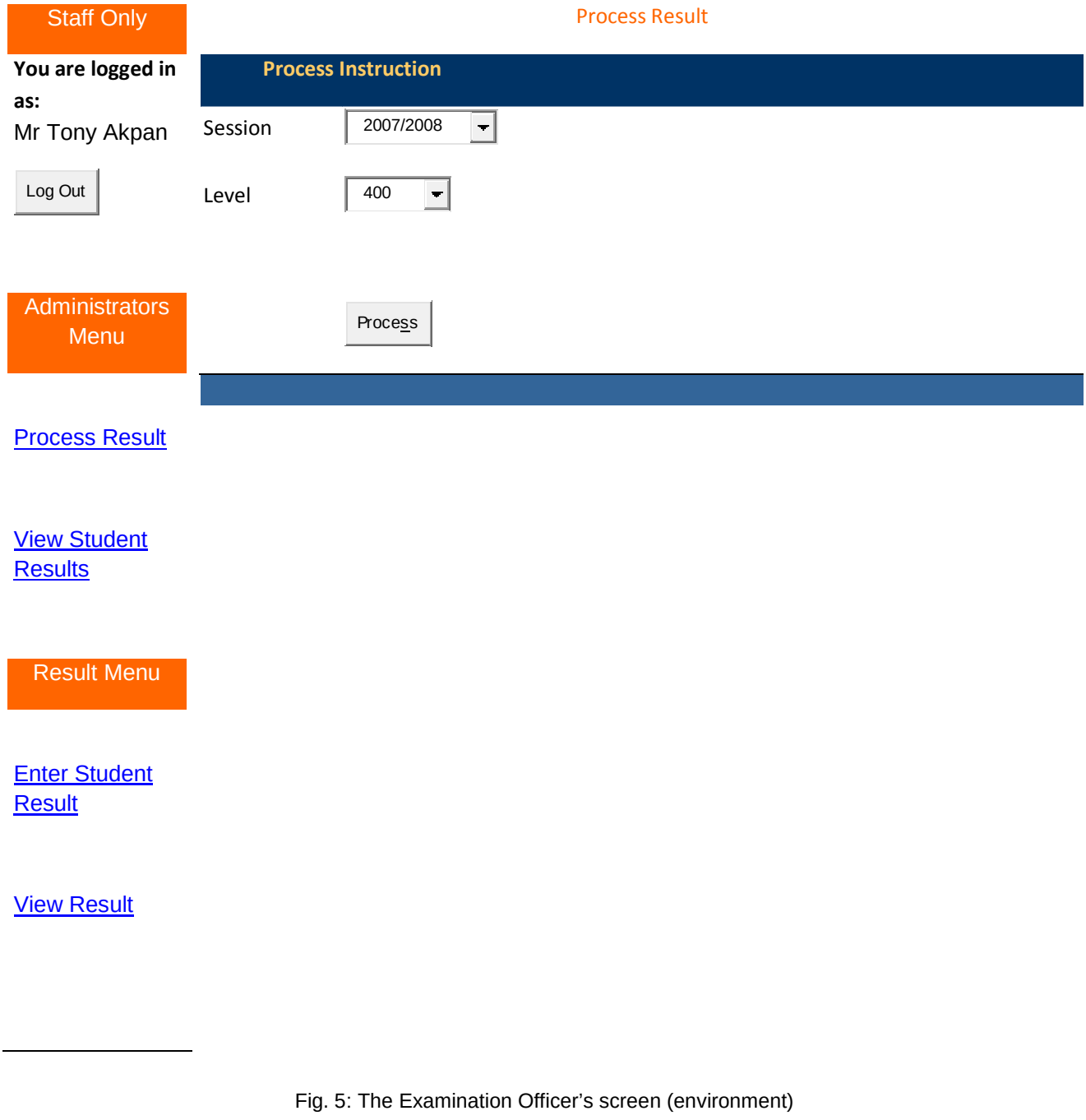

Fig. 5: The Examination Officer's screen (environment)

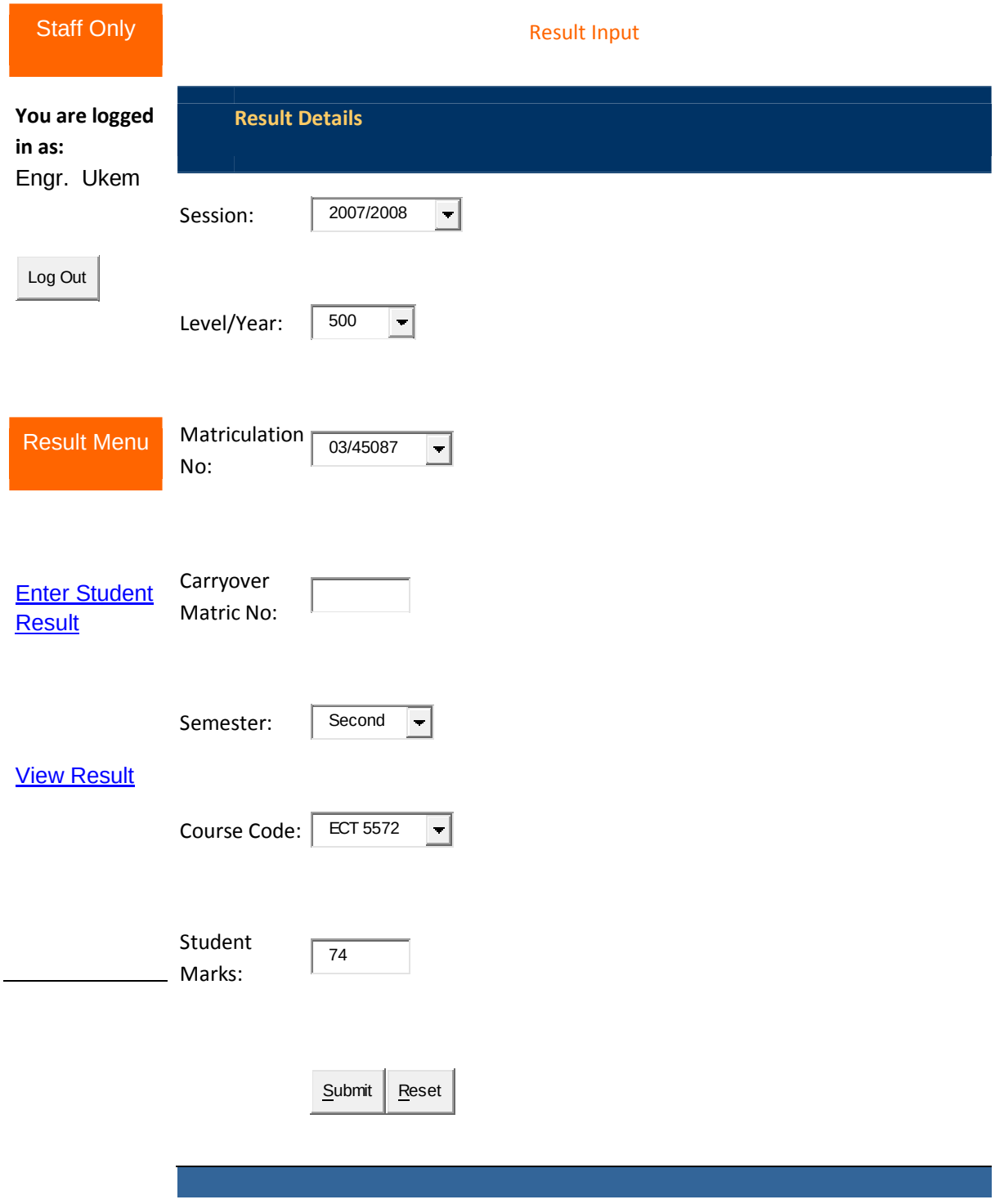

Fig. 6: A Course Coordinator's screen (environment)

Staff Only

User Name:

Password

Log In

View Result

Enter Your Information — Akpan Sule Bayo

Matriculation No: 03/45087

Level/Year: 200

Fetch Result

First Semester

Course Name	Course Code	Marks	Points	Grade
Computer Programming 1	GSS 2111	75	10	A
Advanced Maths 1	MTH 2011	33	0	F
Classical Mechanics	PHY 2211	33	0	F
Thermal Physics 1	PHY 2231	35	0	F
Statistics For Physicist	PHY 2241	64	8	B
Electromagnetism 1	PHY 2251	64	12	B

Second Semester

Course Name	Course Code	Marks	Points	Grade
Advanced Maths 2	MTH 2022	87	15	A
Waves and Sound	PHY 2242	89	15	A
Electric circuit and Electronics	PHY 2262	55	9	C
Introductory Atomic and Nuclear Physics	PHY 2272	76	15	A
Laboratory Physics 2	PHY 2280	63	8	B

Grade Points Statistics

Level/Year	Session	GPA
100	2003/2004	4.24242
200	2004/2005	3.06667
CGPA		**3.68253968254**

Fig. 7: Year Two Resul

Staff Only

User Name:

Password

Log In

View Result

Enter Your Information — Akpan Sule Bayo

Matriculation No: 03/45087

Level/Year: 500

Fetch Result

First Semester

Course Name	Course Code	Marks	Points	Grade
Practicals	ECT 5580	66	8	B
Computer Technology 2	ECT 5531	72	15	A
Electronic Engineering 2	ECT 5541	56	9	C
Material Science	ECT 5551	54	9	C
Environmental Physics	PHY 4901	76	10	A
The Engineer in Society	ECT 5591	62	4	B

Second Semester

Course Name	Course Code	Marks	Points	Grade
Project Work	ECT 5590	72	30	A
Electroacoustics Engineering	ECT 5532	56	9	C
Telecommunication 2	ECT 5542	44	3	E
Electronic Networking	ECT 5572	64	4	B
The Engineer in Society 2	ECT 5592	54	3	C
Element of Cooperate Law	LAW	65	12	B
Management Concept	MSS	59	9	C

Grade Points Statistics

Level/Year	Session	GPA
100	2003/2004	4.24242
200	2004/2005	3.06667
300	2005/2006	3.06667
400	2006/2007	4.75
500	2007/2008	3.67647
CGPA		**3.64028776978**

Fig. 8: Final Year Results

CONCLUSION

The application software was successfully developed, tested, and found to be working as expected. It is a well-designed software that is capable of storing and processing students' results with high speed and accuracy. It has adequate security that enforces data integrity. This results from the use of a relational database management system. It is elegant, convenient, and easy to use due to the use of a GUI rather than command-line approach. It also has a large database that can automatically increase in size (though this needs to be monitored so that it does not become

unmanageable). With this application, the processing of students' results can be automated to a large extent, thereby reducing processing time and increasing accuracy. Further measures can be taken to improve the security of the system, as security is of paramount importance. Such measures could include the use of data encryption, audit trail, and multi-layer backup.

REFERENCE

Curtis, G. and Longworth, G., 1998. Guide to Software Engineering Paradigm

Ekpenyong, Moses E., 2008. "A Real-Time IKBS for Students Results Computation". Journal of Ultra Scientist of Physical Sciences: International Journal of Physical Sciences 20 (3): 627-632.

Goodman, D., 2003. JavaScript Bible, Gold Edition, Hungry Minds Inc, New York

Jay, B. and Brad, P., 2007. Introduction to PHPMYSQL. White Scientific Production, Toronto.

Mazza, C. and Depablo, D., 1994. Software Engineering Standards. McGraw-Hill Inc., New York

Onoyom-Ita, E. O., 2008. "A Computer Software for the Processing of Students' Results in the Physics Department". A final year project in the Department of Physics, University of Calabar.

Pressmam, E., 2001. Software Engineering Techniques. The Conolis Group, New York.

Stephen, E., 1992. Introduction to Software Engineering. Los Altos, California.

Tata, 1995. Database Management Systems. Tata Consultancy Services, Nariman Point, Bombay, India.

Williams, Brian K. and Sawyer, Stacey C., 2003. Using Information Technology: A Practical Introduction to Computers & Communications. McGraw-Hill Irwin, Montreal.

APPENDIX
A LISTING OF SOME CODE SNIPPETS

The first code snippet that is shown here is the portion that creates the database with its tables, and populates some of the tables as desired.

```
File name ResultSystemDB.sql
DROP DATABASE IF EXISTS `ResultSystemDB`;

CREATE DATABASE `ResultSystemDB`;

USE `ResultSystemDB`;

CREATE TABLE `TBLStaff`(
    `staff_id` varchar(10) NOT NULL,
    `staff_name` varchar (30) NOT NULL,
    `unit` varchar (20) NULL,
    `role` varchar (50) NOT NULL,
    `user_name` varchar (30) NOT NULL,
    `password` varchar (30) NOT NULL,
    PRIMARY KEY(`staff_id`)
);

CREATE TABLE `TBLStudent`(
    `matric_no` varchar(10) NOT NULL,
    `full_name` varchar (30) NOT NULL,
    `level_resumed` varchar (10) NULL,
    `session_resumed` varchar (30) NOT NULL,
    `unit` varchar (30) NOT NULL,
    `current_level` varchar (10) NOT NULL,
    `registered_staff` varchar (10) NOT NULL,
    PRIMARY KEY(`matric_no`)
);

CREATE TABLE `TBLCourse`(
    `course_code` varchar(10) NOT NULL,
    `course_name` varchar (80) NOT NULL,
    `credit_unit` int(4) NOT NULL,
    `level_year` varchar (10) NOT NULL,
    `semester` varchar (10) NOT NULL,
    `registered_staff` varchar (10) NOT NULL,
    PRIMARY KEY(`course_code`)
);

CREATE TABLE `TBLStaffCourses`(
    `staff_id` varchar(10) NOT NULL,
    `course_code` varchar (10) NOT NULL
);

CREATE TABLE `TBLCalculatedGPAs`(
    `matric_no` varchar(10) NOT NULL,
    `level_year` varchar (80) NOT NULL,
    `session` varchar(10) NOT NULL,
    `gpa` float(4) NULL,
    `promotion_status` varchar (15) NULL
);

CREATE TABLE `TBLStudentLevels`(
    `matric_no` varchar(10) NOT NULL,
    `level_year` varchar (80) NOT NULL,
    `session` varchar(10) NOT NULL
);

CREATE TABLE `TBLResult`(
    `matric_no` varchar(10) NOT NULL,
    `course_code` varchar (10) NOT NULL,
```

```
    `marks` int(4) NOT NULL default '0',
    `staff_id` varchar (10) NOT NULL,
    `grade` varchar (2) NULL,
    `points` float (10) NULL,
    `session` varchar (10) NOT NULL,
    `level_year` varchar(10) NULL,
    `number_of_times` int (2) NULL,
    `unit` varchar (25) NULL
);

INSERT INTO `TBLStaff` VALUES('admin', 'Super Aministrator', 'Pure Physics', 'HOD', 'admin', 'admin');

GRANT INSERT, SELECT, DELETE, UPDATE ON ResultSystemDB.* TO 'result'@localhost identified by
'software';
```

The next snippet is from the section that handles students' registration. Among other things, it ensures that there are no multiple registrations by any individual.

File name: Student_reg.php

```
<?PHP
    session_start();
    include 'connect.inc';
    if(empty($_SESSION['staff_id']))
        header("Location: index.php");

    if($_SESSION['role'] != "HOD")
        header("Location: index.php");

    if(isset($_POST['submit']))
    {
        $matric_no = $_POST['matricNoTxt'];
        $full_name = $_POST['fullNameTxt'];
        $level_resumed = $_POST['levelResumedDown'];
        $session_resumed = $_POST['sessionResumedDown'];
        $unit = $_POST['unitDown'];

        $message = '';
        $error_header = '<tr>
            <td align="center"><table width="100%" border="0" cellspacing="3" cellpadding="2"
class="loginTable">
              <tr>
                <td>
                <strong>Please note this message:</strong><br>
                <ul>';

        $error_footer = '</td>
                          </tr>
                        </table></td>
                  </tr>
                  <tr>
                        <td align="center"> </td>
                  </tr>';

        if(empty($matric_no))
        {
            $message .= '<li>Student matriculation number is required</li>';
        }
        if(empty($full_name))
        {
            $message .= '<li>Student name is required</li>';
        }

        $sql = "SELECT * FROM TBLStudent WHERE matric_no = '$matric_no'";
        $result = mysql_query($sql, $conn) or die(mysql_error());

        if(mysql_num_rows($result) > 0)
        {
```

```
                        $message .= "<li><strong>".mysql_result($result, 0, 'full_name')."</strong> has been
    registered already with this matriculation number</li>";
                }

                if(empty($message))
                {
                        $sql = "INSERT INTO TBLStudent VALUES('$matric_no', '$full_name',
    '$level_resumed', '$session_resumed', '$unit', '$level_resumed', '$_SESSION[staff_id]')";
                        $result = mysql_query($sql, $conn) or die(mysql_error());

                        $sql = "INSERT INTO TBLCalculatedGPAs VALUES('$matric_no', '$level_resumed',
    '$session_resumed', '0', ')";
                        $result = mysql_query($sql, $conn) or die(mysql_error());

                        $sql = "INSERT INTO TBLStudentLevels VALUES('$matric_no', '$level_resumed',
    '$session_resumed')";
                        $result = mysql_query($sql, $conn) or die(mysql_error());

                        $message .= '<li>Student registration was successful!</li>';

                        $matric_no = $full_name = "";
                }
        }
```

The next code snippet shown below assigns a particular letter grade to a score and also computes the credit points of each course.

```
    function compute_grade($marks)
        {
            $grade = "";
            if($marks < 40)
            {
                    $grade = "F";
            }
            else if(($marks >= 40) && ($marks <= 44))
            {
                    $grade = "E";
            }
            else if(($marks >= 45) && ($marks <= 49))
            {
                    $grade = "D";
            }
            else if(($marks > 49) && ($marks <= 59))
            {
                    $grade = "C";
            }
            else if(($marks > 59) && ($marks <= 69))
            {
                    $grade = "B";
            }
            else
            {
                    $grade = "A";
            }
            return $grade;
     }
    function compute_points($marks, $course_code)
    {
            global $conn;
            $sql = "SELECT credit_unit FROM TBLCourse WHERE
     course_code='$course_code'";
            $result = mysql_query($sql, $conn) or die(mysql_error());
            if(mysql_num_rows($result) > 0)
            {
                    $credit_unit = mysql_result($result, 0, 'credit_unit');
            }
            $points = 0.0;
```

```
        if($marks < 40)
        {
                $points = 0*$credit_unit;
        }
        else if(($marks >= 40) && ($marks <= 44))
        {
                $points = 1*$credit_unit;
        }
        else if(($marks >= 45) && ($marks <= 49))
        {
                $points = 2*$credit_unit;
        }
        else if(($marks > 49) && ($marks <= 59))
        {
                $points = 3*$credit_unit;
        }
        else if(($marks > 59) && ($marks <= 69))
        {
                $points = 4*$credit_unit;
        }
        else
        {
                $points = 5*$credit_unit;
        }
        return $points;
}
```

The code snippet that follows is part of the actual result processing logic, with some of the built in checks. It tracks the number of times a course is taken, and ensures that no course, except the GSS courses, can be taken more than three times.

File name: Student_result.php

```
<?PHP
    session_start();
    include 'connect.inc';
    include 'validate_user.php';

    if(empty($_SESSION['staff_id']))
            header("Location: index.php");

    if($_POST['submit'] == "Submit")
    {
            $session_value = $_POST['sessionDown'];
            $course_code =  $_POST['courseCodeDown'];
            $matric_no = (empty($_POST['carryoverTxt'])) ? $_POST['matricNoDown']  :
$_POST['carryoverTxt'];
            $level = $_POST['levelDown'];
            $semester = $_POST['semesterDown'];
            $student_marks = $_POST['marksTxt'];

            $message = '';
            $error_header = '<tr>
                    <td align="center"><table width="100%" border="0" cellspacing="3" cellpadding="2"
class="loginTable">
                      <tr>
                            <td>
                            <strong>Please note this message:</strong><br>
                            <ul>';

            $error_footer = '</td>
                                    </tr>
                                </table></td>
                              </tr>
                              <tr>
                                <td align="center"> </td>
                              </tr>';
            if(empty($course_code))
```

```
		{
			$message .= '<li>Course code is required</li>';
		}
		if(empty($matric_no))
		{
			$message .= '<li>Student matriculation number is required</li>';
		}
		if(empty($student_marks))
		{
			$message .= '<li>Student marks are required</li>';
		}

		settype($student_marks, 'Integer');

		if(empty($message))
		{
			$str_course = explode(" ", $course_code);
			$flag = 0;
			if(strtoupper($str_course[0]) != "GSS")
			{
				$sql = "SELECT COUNT(number_of_times) AS times FROM TBLResult
WHERE matric_no = '$matric_no' AND  course_code = '$course_code'";
				$result = mysql_query($sql, $conn) or die(mysql_error());

				if(mysql_num_rows($result) > 0)
				{
					$num_times = mysql_result($result, 0, 'times');
					if($num_times <= 2)
					{
						$flag = 1;
					}
					else
					{
			$message = '<li>This student has attempted this course up to three times!</li>';
			$message .= '<li>Maximum allowed times 3 (three)</li>';
					}
				}
				else
				{
					$flag = 1;
				}
			}
			else
			{
				$flag = 1;
			}
				if($flag == 1)
			{
				$sql = "DELETE FROM TBLResult WHERE matric_no = '$matric_no' AND
level_year = '$level' AND course_code = '$course_code' AND session = '$session_value'";
				$result = mysql_query($sql, $conn) or die(mysql_error());

				$grade = compute_grade($student_marks);
				$points = compute_points($student_marks, $course_code);

				$sql = "INSERT INTO TBLResult VALUES('$matric_no', '$course_code',
'$student_marks', '$_SESSION[staff_id]', '$grade', '$points', '$session_value', '$level', '1',
'$_SESSION[unit]')";
				$result = mysql_query($sql, $conn) or die(mysql_error());

				$message = '<li>Student result was processed successfully!</li>';

				$student_marks = "";
			}
		}
	}
```