Xpernal 1.0 – Student Grade Performance Analyzer

D. D. M. M. Arachchi
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka
dilinimma.12@cse.mrt.ac.lk

***Abstract*— Making decisions by analyzing data is very common in all aspects. Similarly, in universities also, students and lecturers want to analyze results and track performance. Accomplishing this task manually will be very cumbersome, time consuming and error prone. The most feasible solution would be a software system that includes functionalities to overcome those problems. As web based applications are more trending, the applicability of the solution will increase if it is also a web based solution. Therefore, this paper introduces a web based software application that can achieve those goals reliably and efficiently. The system analyzes, compares and predicts student's grades and manages results and modules. The application was implemented using JavaScript with backend implementation using Java, while adhering to the Spring Web MVC framework. The system provides meaningful and informative reports to the students and the lecturers. In addition, it provides easy and user friendly interfaces for the lecturers to achieve the goal of managing results and modules.**

Keywords—student grade performance analyzer; correlation; predict results

I. INTRODUCTION

Analyzing students' grades is an important task in an educational institution such as a university. It helps to maintain the academic quality of the institution. With the growth of information technology, having an automated system to analyze students' academic records is useful in many ways. This paper describes such system which was implemented as a web based solution.

Xpernal 1.0 is a web based student grade performance analyzer. The domain of this system is the Department of Computer Science and Engineering (CSE) at University of Moratuwa. The users of this system are constrained to be the CSE students and the lecturers and the system manages only the compulsory modules.

A grade performance analyzer system was considered for this project in order to cater the challenges in existing manual system such as time and effort consumption and to prevent inaccuracy due to human errors. In a university system an undergraduate may need to know their performance in the selected modules or semesters. For example, in the bidding session for internship, students have to know their GPA (Grade Point Average) in order to claim for bonus points and to be prioritized in the selection process. Also, lecturers may want to analyze how the students perform in their modules. These facts increase the necessity of a system like Xpernal.

The objective of this system is to provide an automated tool to manage and analyze students' grades. With such kind of system, a student does not need to recall the grades he/she has received and able to compare the grade with other students' grades. Moreover, the system predicts the likely grade a student may obtain for a module he/she is going to take in the future. Furthermore, a lecturer does not need to concern about entering results one by one for each student, as the system provides an easy way to achieve it. Lecturers can keep track of the performance of each student and provide useful feedback for him/her if required.

The rest of the paper is organized as follows. Section II describes the related literature and the related systems. Section III discusses the requirements and the design of the system. Section IV delineates how the system was implemented and section V includes the analysis of system testing. Finally, section VI conveys the conclusion and the future work.

II. LITERATURE REVIEW

The theoretical aspect of this system falls majorly in the results analysis subsystem. When comparing and predicting students' grades, the main mathematical concept used was the correlation. According to A. M. Tuttle, *"Correlation is an analysis of the covariation between two or more variables."* [1].

$$Correlation(X,Y) = \frac{Covariance(X,Y)}{\sqrt{Variance(X) * Variance(Y)}}$$

Hence, when comparing two students' performance, correlation gives a reasonable linear relationship between those students. One advantage is, using correlation we can even find out whether there is no relationship at all among the students' grades. If the calculated correlation of two students is positive it means there is a significance relationship i.e. when one student's grades increase the other student's grades also increase. If it is negative, that means there is a negative relationship i.e. when one student's grades increase, the other student's grades decrease. On the other hand, if the correlation is zero that means the two students' grades are independent of each other. In addition to correlation, the average performances of the students are calculated based on the grades they have obtained. As the actual marks of the students were not obtainable due to legal issues, each grade has to be mapped to its corresponding grade point value. Hence, it was assumed that the mapping of grades does not affect the accuracy of the calculated results.

The results predicting functionality employs a simple algorithm based on correlation. This algorithm was implemented in this manner because of the unavailability of actual marks. When a student requires a likely grade of a future module, the algorithm calculates the result if it can, and if it cannot predict due to any reason, the system will say that no result can be predicted. The accuracy of this algorithm depends solely on the number of grades of previous batches that are available in the database. The more the number of grades of previous batches available, the more accurate the predicted

result will be. It should be noted that the grade is predicted purely based on the correlation. More on this algorithm will be discussed in Section IV.

Analyzing students' performance using their grades is not a new concept in software applications. There are existing systems which analyze students' grades and generate informative reports that help track the students over the past academic periods. Schoolspeak.com [2] and GradeXpert.com [3] are some examples for such systems. Schoolspeak.com provides its services for school systems where the teachers and the parents involve in analyzing data. It is more popular in the schools in USA. It provides a rich collaboration between the teachers and the parents via the analysis of student's grades. Moreover, GradeXpert.com is also a grade performance analyzer that provides its services for the Australian schools. It provides graphical analysis of students' grades with respect to some selected benchmarks. However, most of the existing systems constrain their scope for school systems with the users often being teachers and parents. That is a major difference between Xpernal and the existing systems.

Xpernal provides its services for the lecturers and the students of the university. Hence, the domain slightly changes from that of the existing systems. None of the above systems offers to predict the likely grades of a student for a module he is going to take in future. However, there has been researches conducted [4, 5, 6] to see how the Decision Tree Algorithms (DTA) can be used to predict students' grades using existing data. In their paper [7] A. Acharya et al. elaborate how Machine Learning Algorithms (MLA) can be used to predict student performance and a comparison of 5 classes of MLAs that were employed for the prediction. P. Cortez et al. describe the possibility of student performance prediction and the factors affect it in their paper [8]. They specifically focus on Data Mining techniques in their paper. How the MLA called Neural Networks can be used to predict student performance has been analyzed in their paper [9] by H. Agrawal et al. These are some interesting applications of statistical models in predicting student performance. However, the project had to be carefully designed so as to match with the given limited time period and the grade prediction functionality was only a sub functionality of the whole system. Hence, a statistical model like Pearson correlation that identifies a linear relationship in the grades when predicting results was used.

To sum up, researching more on these concepts and exporting them to Xpernal 1.0, can be proposed as a future enhancement of the system.

III. System Models

A. System Requirements

The major functional requirements of the system are; it should graphically analyze trends in one student's grades over a number of semesters, compare a student's performance against another student over a course module or many, predict the expected grade of a student in future module, provide features to view/update student profile and view feedback given by the lecturers. These requirements are depicted in Fig. 1 via the use cases linked to the student. Also using the system a lecturer should be able to, update/enter results, update/enter course modules, update/view own profile, analyze trends in individual student's performance and provide feedback for students. As shown in Fig. 1, the lecturer account should be verified before proceeding with updating course module and results.

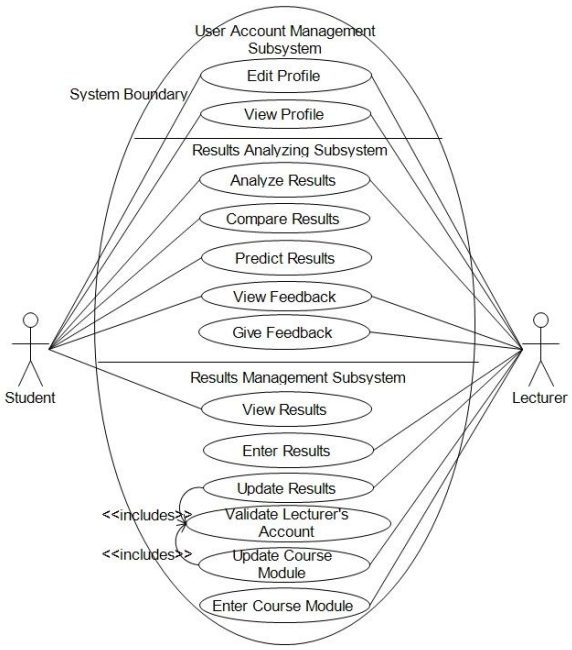

Fig. 1. Use-case Diagram of the System

The non-functional requirements can be classified into 3 categories. Of them, one usability requirement is; it should take maximum of 2-3 hours for a normal user to get a complete training regarding how to handle the system. The system should provide user friendly and easy interfaces for the lecturers to enter/update results and modules. The system should provide feedback for each user activity and meaningful error messages when an error occurs. The reliability requirements are; the system should be available 24/7 with a low Mean Time to Repair (MTTR) and a high Mean Time to Failure (MTTF) as well as the Mean Time between Failures (MTBF) of the system should be considerably high. Another important requirement is, the system should provide higher accuracy when analyzing students' grades. The performance requirements are; the response time for analyzing a student's grades and delivering it for the user should be less than 10 seconds (maximum). When comparing and predicting student's grades, the maximum time system should take for each task should be 5-10 seconds (maximum).

B. System Design

The system follows three-layer architecture with the standards provided by the Spring MVC Framework (Fig. 2). The presentation layer contains all the user interfaces (UIs) and controllers. The data access layer accesses the database, stores and retrieves data on requests. The business logic layer contains the functionalities and the service implementation of the system including user accounts managing, results analyzing and results managing.

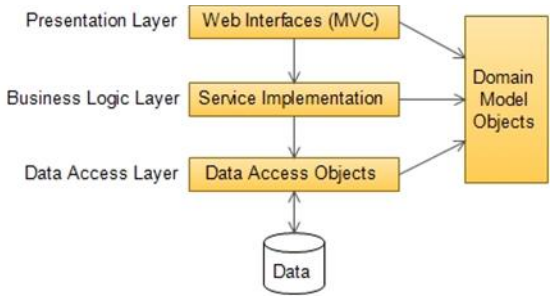

Fig. 2. The Layered Architecture

The logical view of the system is represented by the UML class diagrams which gives the interaction between each design class. As described earlier, the system has two types of users; student and lecturer. Both users share common attributes, hence they were inherited from a base class (User) to increase the reusability. According to the identified requirements, one result sheet contains a single batch, a single module and a set of grades whereas a module can have zero or more results sheets corresponding to different batches. A batch can have zero or more results sheets. A single result object is a collection of student index number, a grade object and a module object. One module can have more than one lecturer. These associations are shown in Fig. 3.

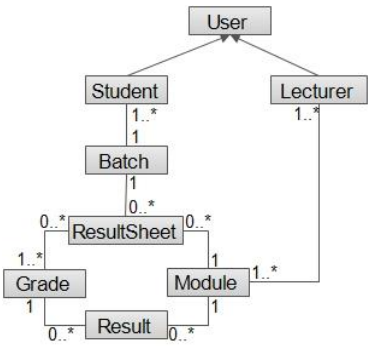

Fig. 3. Design Class Diagram

The process view of the system is illustrated using the activity diagrams. Fig. 4 shows the main activity diagram of the system regarding a student. A student has to login to the system in order to analyze his/her results. Before analyzing, he should specify the required data to the system (Fig. 4). For example, he might select the semester he wants to see his/her performance. The same process will be carried out when student is using the system to compare with a friend and predict future results.

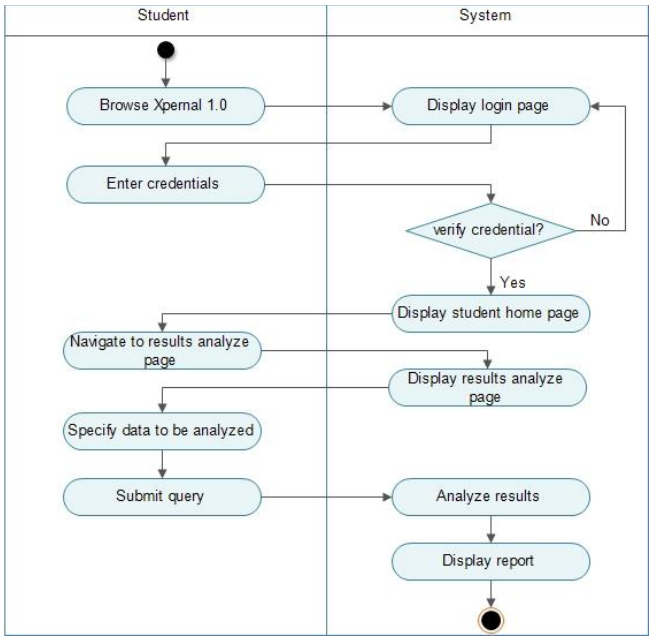

Fig. 4. Activity Diagram for, Student Analyzing His/hers Results

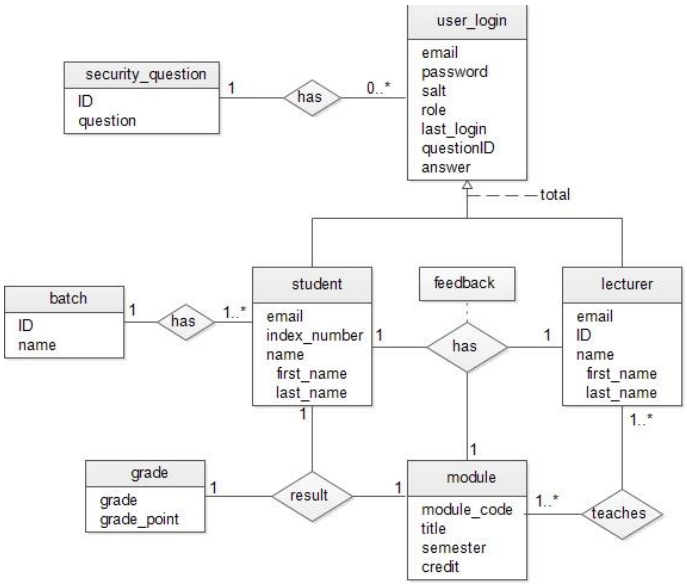

Fig. 5. Entity Relationship Diagram of the System

The Entity Relationship Diagram (ERD) shown in Fig. 5, depicts the data and their associations. Here, *student* and *lecturer* have a total generalized relationship with *user login*. We assumed a single module can have multiple lecturers.

The ERD was converted and normalized to the Relational Schema (Fig. 6) to avoid data redundancy and improve efficiency. Moreover, passwords were encrypted using Secure Hash Algorithm and a salt value prior to storing, thereby avoiding brute-force attacks on them.

$user\_login(email, password, salt, role, last\_login, questionID, answer$
$student(email, index\_number, first\_name, last\_name, street, city,$
$mobile\_number, gender, dob, image)$
$lecturer(email, ID, first\_name, last\_name, street, city, department,$
$gender, image)$
$grade(grade, grade\_point)$
$batch(ID, name)$
$module(module_{code}, title, continuous, written, semester, gpa, credit)$
$security\_question(ID, question)$
$result(student\_ID, module\_code, grade)$
$sudent\_batch(batch\_ID, student\_ID)$
$feedback(module\_code, student\_ID, lecturer\_ID, feedback)$
$teaches(lecturer\_ID, module\_code, year)$

Fig. 6. Relational Schema of the ERD

IV. System Implementation

A. Implementation Procedure

The tool Xpernal 1.0 is developed as a web based application, where JavaScript is used for the client side scripting language, and Java is used for the backend development. The implementation was adhered to Spring MVC (Model-View-Control) framework. For the UI designing, Bootstrap and CSS were used. A dashboard theme was reused as the main theme of the system. Bootstrap Form Validation was used to validate forms. Database was implemented using MySQL. The system was initially deployed in the Apache Tomcat Server 8.0.3.0. The implementation was done using the NetBeans IDE 8.0. The Rational Unified Process (RUP) software development methodology [10] is used to achieve a considerably exhaustive testing phase at the end of each iteration as each component was implemented iteratively.

This tool has used reusable software libraries such as Apache Commons Math 2.2 [11] to calculate the correlation and variance among the students' grades. User Interfaces were implemented using "startbootstrap-sb-admin-1.0.3" theme [12] in order to improve the aesthetic aspects of the UIs. The results analysis subsystem uses graphs to plot relationships, analyzed data etc. with the use of an external library called "Plotly.js" [13]. Further, Bootstrap Form Validation [14] was used to validate the forms.

The grades of two CSE batches, 2011 and 2012, were used for the prediction sub system. The grades were stored in the format "A+, A, A-, B+, B… etc." Each grade was mapped to its grade point value when analyzing them. The system manages only the modules which are compulsory for all the CSE stream students of the department. No other elective module was considered. This restriction was created because, when analyzing data a large sample would give better and accurate results. Other than the grades of the students, and the set of modules, a profile was created for all the students of batches '11, '12 and '13. Later these students can update their profiles using the system.

The pseudo code of the algorithm for predicting a future result using correlation is given in Fig. 7.

```
Function PredictResult(index, module_code)
    user_student ← getStudent(index)
    i ← 0
    if(! isVarying(user_student) //if there is no variation in user_student's grades
        average ← getAverage(user_student)
        return getGrade(average)
    students ← list of all students in previous batch
    for each student in students
        modules ← list of all modules user_student and student has obtained results
        studentMarks[] ← resultsOf(modules, user_student)
        otherStudentMarks[] ← resultsOf(modules, student)
        correlation[i++] ← calculateCorrelation(studentMarks, otherStudentMarks)
    max _index ← getIndexOfMaximumCorrelation(correlation)
    grade ← getGrade(students[max _index], module_code)
    if(grade is null)
        return error
    else
        return grade
```

Fig. 7. The Algorithm that Predicts the Future Grade of a Student

First, a student (Y) selects a module code (X) for which he wants to have a result predicted. He can choose modules which he has not received a grade yet, and for which results of previous batches are available in the database. Then, the algorithm checks whether there is a variation in Y's grades. If there is no variation that means the student is likely to get the same result for X also. Moreover, the correlation of Y's grades and others grades will be infinity as the variance of Y's grades is zero. Hence the grade corresponding to the average grade point of student Y is returned as the predicted result.

If there is a variation in Y's grades, a list of students of previous batch is retrieved. Then student Y is paired with each student (Z) of that list. Next the algorithm gets the set of modules for which both students (Y and Z) have received grades. Using the two arrays of grades for those modules, the algorithm calculates the correlation of each pair. Using the calculated correlations, the student (Z) having the maximum correlation with Y's grades is retrieved from the list.

The result Z got for the module X, if one exist, will be returned as the likely result of the requested student (Y). Following assumptions were made when predicting the future grade of a student using this algorithm. Predicting depends only on the correlation between the students, not any other factors such as, the time one spends studying, the methods used to study, attendance to the lectures, the difficulty level of the exam papers and other activities of the selected module.

B. Main Interfaces

Apart from the home pages of the students and the lecturers, one of the main interfaces is the individual results analysis page. Using this (Fig. 8), a student can see his/her trends for 3 categories of modules; Mathematics, CSE and Other. Also, a student can see how he has performed in each semester using this interface. In order to compare with a friend (Fig. 9), the student has to select a friend first. Then he has two options. He can compare either semester-wise or module-wise. Using the interface shown in Fig. 10, a student can see the likely grade he will receive for a module he is going to take in the future. He has to select a module either by the module code or title from a list of modules he has not taken yet.

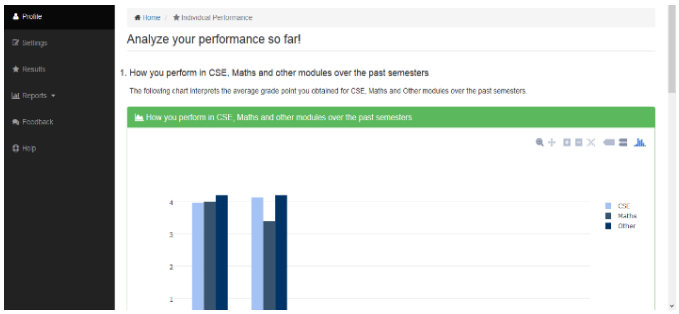

Fig. 8. Individual Results Analysis Page

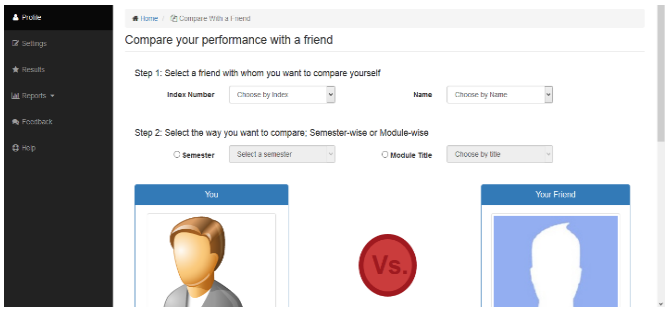

Fig. 9. Compare with a Friend

A lecturer can use the interface shown in Fig. 11 to enter results of a whole batch. First he should select the module for which he is going to insert results. Then he should give the year in which this module was taught. He should select the corresponding batch. Finally, he should upload a text file containing lines of the format, 123456Z A+.

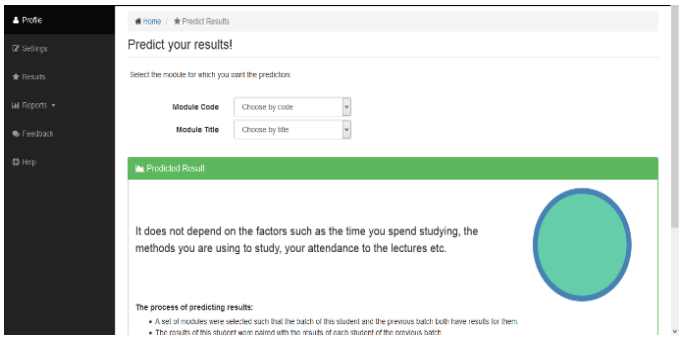

Fig. 10. Predict a Future Result

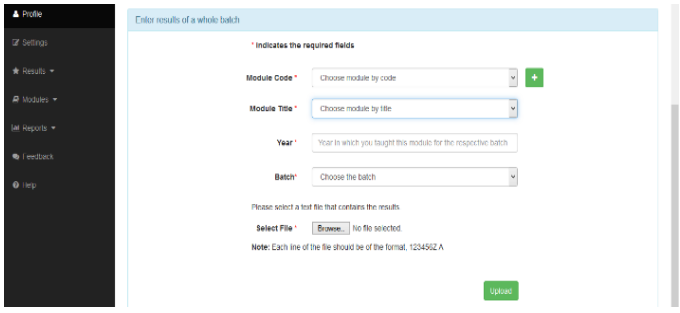

Fig. 11. Results Entering Page

V. System Testing and Analysis

The tool Xpernal was tested prior to deployment to see whether the system satisfies the requirements as intended. The testing approach was iterative and each component was tested before and after integrating to the system. Testing covered four major aspects of the system; functions, security, performance and Graphical User Interfaces (GUI).

Function testing was carried out involving unit testing. All the data access objects and the algorithms were tested. Unit testing was performed using JUnit 4.0. Each method was tested with test data and the actual outcome was assessed with the expected outcome. Almost every function test case passed successfully thereby consolidating the functionalities. Also, each and every functionality of Xpernal was tested via "black-box testing" approach in which the results of executing the use case scenarios are verified by interacting with the GUIs of the system. All functionalities were tested giving both valid and invalid data to see whether the intended feedback is received.

The eventual goal of performance testing is to verify whether the performance requirements have been satisfied. This was carried out using JUnit and JUnitPerf libraries. Most critical functionalities such as results analysis and results prediction etc. were tested to see whether they have acceptable response time. To achieve that, a maximum elapsed time (t_m) was set for each tested method. Then, the test-target was executed and the execution time (t_e) was measured. The test was considered to be succeeded if $t_e \leq t_m$ and to be failed if $t_e > t_m$.

UI testing was carried out to verify that the system navigation happens as expected, and the elements in the pages function as desired. Selenium and JUnit libraries were used for this. Page navigation was tested in all the UIs. Moreover, the text fields and buttons were tested to see whether they provide correct tab key navigations. User acceptance testing was done by engaging a set of students of CSE and slight changes were made based on their feedback.

Application-level security which considers about the access to data and functionalities and, system-level security which considers about logging into the system, were focused in security and access control testing. Testing was carried out using Selenium and JUnit libraries. Two different user accounts were created for lecturer and student, and then tested whether each user gets access only to functionalities and data for which the role is given permission. For system-level security, the reactions of the system for both valid and invalid login and sign-up were tested.

While testing the system some errors and bugs were found out. There were broken hyper-links which were not working properly and some submissions threw exceptions for null data. They were handled successfully. The test cases were designed to cover almost every significant aspect of the system as described above. However there were some difficulties when designing the test cases. One significant difficulty in carrying out performance testing was to estimate the maximum elapsed time (t_m) as it varies with the input size. Initially the functionalities were tested for a group of inputs (I_n) and then t_m was calculated as a multiplication of time taken for I_n.

Almost all the test cases passed successfully while some failed due to problems in the functionalities. They were corrected and retested to assure that the system delivers expected requirements as intended.

VI. Conclusion and Future Work

This paper presents a web based system, Xpernal 1.0, to analyze student grade performance in department of CSE, University of Moratuwa. The system mainly provides functionalities to analyze student grades over the past semesters and calculate the GPA, which is time and effort consuming task, otherwise. Using this system, a student can compare his performance with the performance of another friend or the entire batch. Further, the system allows a student to predict his/her expected grade for a future module by analyzing the grades of previous batches. Lecturers are provided with extra functionalities such as manage modules and results. Xpernal 1.0 consists of user friendly GUIs that both students and lectures can use.

The system can be further enhanced in many ways. The accuracy of the predicted result can be improved by considering the actual results of the students, instead of their grades. As the Continuous Assessment (CA) marks affect the final grade, it is suggested to consider CA as well in the results prediction process. Including the functionality to enter results to the system using MS Excel sheets rather than text files, can be proposed as another future enhancement. Also, the accuracy of the results prediction sub system can be enhanced using machine learning and decision tree algorithms. They can also be used to identify various patterns in students' grades. Further, this system can be developed as a mobile application in order to increase its usability. Introducing a separate Admin account to manage all the administrative functionalities can also be proposed as another possible future work. User acceptance and performance testing results evaluations can be included to produce a sophisticated version of Xpernal.

References

[1] A. K. Sharma, "The Text Book of Correlations and Regression", New Delhi: Discovery Publishing House, 2005.

[2] "SchoolSpeak," [Online]. Available: http://www.schoolspeak.com/CompanyV4/performance-analysis.aspx.

[3] "GradeXpert," [Online]. Available: http://www.gradexpert.com.au/.

[4] M. Pandey and V. K. Sharma, "A Decision Tree Algorithm Pertaining to the Student Performance Analysis and Prediction," *International Journal of Computer Applications,* vol. 61, no. 13, January, 2013.

[5] T. M. Lakshmi et al., "An Analysis on Performance of Decision Tree Algorithms using Student's Qualitative Data," *I.J.Modern Education and Computer Science,* pp. 18-27, June, 2013.

[6] A. Mashael et al., "Predicting Students Final GPA Using Decision Trees: A Case Study," *International Journal of Information and Education Technology,* vol. 6, no. 7, July, 2016.

[7] D. Sinha and A. Acharya, "Early Prediction of Students Performance using Machine Learning Techniques," *International Journal of Computer Applications,* vol. 107, no. 1, December 2014. [Online]. Available: http://research.ijcaonline.org/volume107/number1/pxc3899939.pdf

[8] P. Cortez and A. Silva, "Using Data Mining to Predict". [Online]. Available: http://www3.dsi.uminho.pt/pcortez/student.pdf

[9] H. Agrawal and H. Mavani, "Student Performance Prediction using Machine," *International Journal of Engineering Research & Technology (IJERT),* vol. 4, no. 3, March 2015. [Online]. Available: http://www.ijert.org/view-pdf/12553/student-performance-prediction-using-machine-learning

[10] P. Kruchten, "The Rational Unified Process: An Introduction", 3 ed., Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003

[11] "Commons Math," [Online]. Available: http://commons.apache.org/proper/commons-math/download_math.cgi.

[12] "wuxueying/bootstrap-templates, GitHub," [Online]. Available: https://github.com/wuxueying/bootstrap-templates/tree/master/sb-admin-2-1.0.3.

[13] "Plotly.js," [Online]. Available: https://plot.ly/.

[14] "Validators, FormValidation," [Online]. Available: http://formvalidation.io/validators/.