eForms

A Project Report

Presented to

The Faculty of the Department of General Engineering

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Engineering

By

Shifali Gupta

Fall 2009

APPROVED FOR THE DEPARTMENT OF GENERAL ENGINEERING

Dr. Leonard P. Wesley, MSE Director, Associate Professor Computer Engineering Department

Dr. Simon Shim, Professor Computer Engineering Department, San Jose State University

Acknowledgement

I would like to thank **Dr. Simon Shim, Professor in Computer Engineering** for providing his guidance, reviewing my work and for providing valuable suggestions.

I would like to thank **Dr. Leonard P. Wesley, MSE Director, and Associate Professor Dept. of Computer Engineering** for his valuable insights to the project. I really appreciate Dr. Wesley for his patience, time and guidance.

Shifali Gupta

ABSTRACT

In this Internet age, where increasingly more information, data, and services are available online and can be accessed with blazingly fast speed, it makes perfect sense to make forms available online. Not only it is friendly to the environment as it saves tons of paper, gas and other resources, it is also very economical, robust and convenient for both the provider and the user. In our case both the university and the students will benefit from the new eForm system by having automatically fillable forms and by automatic tracking of student success. A better advising system can save months of student graduating time. In this document we discuss how the eForms system helps address these and related forms processing issues.

Table of Contents

List of Figures

List of Tables

1.0 Objective

Our eForms project has two objectives. The first objective is to convert manual form processing at Universities into online form processing. The second objective is to design an online student advising system which will track student success and will send an alert to both student and his or her advisor in case student does not progress properly.

2.0 Introduction

Eforms are the electronically fillable forms which can improve the efficiency of a system significantly. Eform is the obvious choice in comparison to paper form. Paper forms require manual handling which is time consuming, error prone and costly. On the other hand eforms can be submitted at the convenience of your home, office or anywhere else with internet connection. Submission status could be checked immediately thereafter. With new web2.0 technologies like AJAX and java script, any obvious errors in the form could be checked before even the form is submitted. Because eforms are convenient, economical, saves time and are environmental friendly; they are used almost everywhere. Our dependence on these omnipresent eforms is such that almost 80% of the processes in public and private businesses depend on eforms.

Above is the motivation behind this project where we are implementing a system to convert paper form processing in our SJSU University into eform based system? This is a very scalable system which can be scaled to multiple Universities with little changes and support. Our eForm system will fill the major and candidacy forms for the students automatically without student's intervention. This means all the submissions will be error-free and on time. Thus it will save considerable amount of time currently spent by students and advisors on fixing these manual mistakes. Later on many more forms can be added to this system.

An online student advising system will be a better means to identify if and when some intervention is required. It's a famous saying that a stitch in time saves nine. That is sooner we detect that students need advising, sooner appropriate intervention can be taken. It will alert students in sufficient time to make the appropriate academic changes so that the student has a chance to avoid going on probation. So based on some triggering points like if a student's GPA is falling down or how a student is progressing towards their degree or if a student is lacking in some area then our advising system will send automatic alerts to both the advisor and the students so that appropriate action can be taken to achieve their graduation goal on time.

Error! Bookmark not defined. **Scope**

A requirement of the eForms system is to have databases that contain a variety of information. Examples include, the student's department and degree program, courses required for the degree, the semester, year, and grade received for courses, and so forth. This data is voluminous and will need to be accessed in different ways and with varying frequency while maintaining timely performance.

To help meet this requirement, the scope of our project will focus on:

Reviewing the design of the existing eForms database, suggest and implement changes:

EForms database stores Student information, Student's course grade information, Student major and candidacy form information and department information. Our work involves reviewing this database schema such that resulting database is compact and without redundancy. Also to make sure that this schema is BCNF compliant.

Designing and implementing advising database schemas for the eForms project:

Advising databases will store both non-derived as well as derived data. Non-derived data includes Student id, Department id and the cumulative data like number of previous semesters taken by the student, current GPA trend of the student etc. whereas derived data is the estimated data based on student history. Derived data will be future semesters, future GPA trend and estimated semester when a student might finish his or her degree. This derived data is the key to figure out the appropriate conditions and the time when to advise a particular student. Our work involves finding out the various derived and non-derived data requirements and figuring out an efficient representation (schema) to store this data on disk. Our work will also involve finding out the most commonly used queries and benchmarking the system for performance.

Designing and implementing a report generation mechanism for the eForms project:

Report generation will involve the creation of queries satisfying various information needs. Information needs are different for different databases. For example, for the eForms database one need is to extract all of the approved electives for a particular department and program that was approved in a particular year and semester. For the advising database, there might be a need to identify all students whose GPA trend is negative. Our work will involve figuring out various queries which might be useful once this system is deployed. Also we will measure the performance of various queries and adjust the database schema if needed for fine tuning.

4.0 Design

4.1 Entity-Relationship Data Model

Our ultimate objective was to design an efficient database, for that we started with ER diagrams. ER diagrams are created such that design should follow certain principles. For example, First and foremost, the design should be faithful to the specifications. Entity sets and attributes should reflect reality. Relationship policies should be followed exactly. Secondly, it should avoid redundancy which means we should be careful to represent everything once. We need to remove attributes which don't provide anything new. Derivative information should not be kept. Third principle is to avoid introducing more elements into our design than is absolutely necessary. Fourth is to choose the right relationships. Entity sets can be connected in various ways by relationships, but adding every possible relationship is not always a good idea. Relationships should mimic the real world. Based on these design principles we created ER diagrams which are as follows:

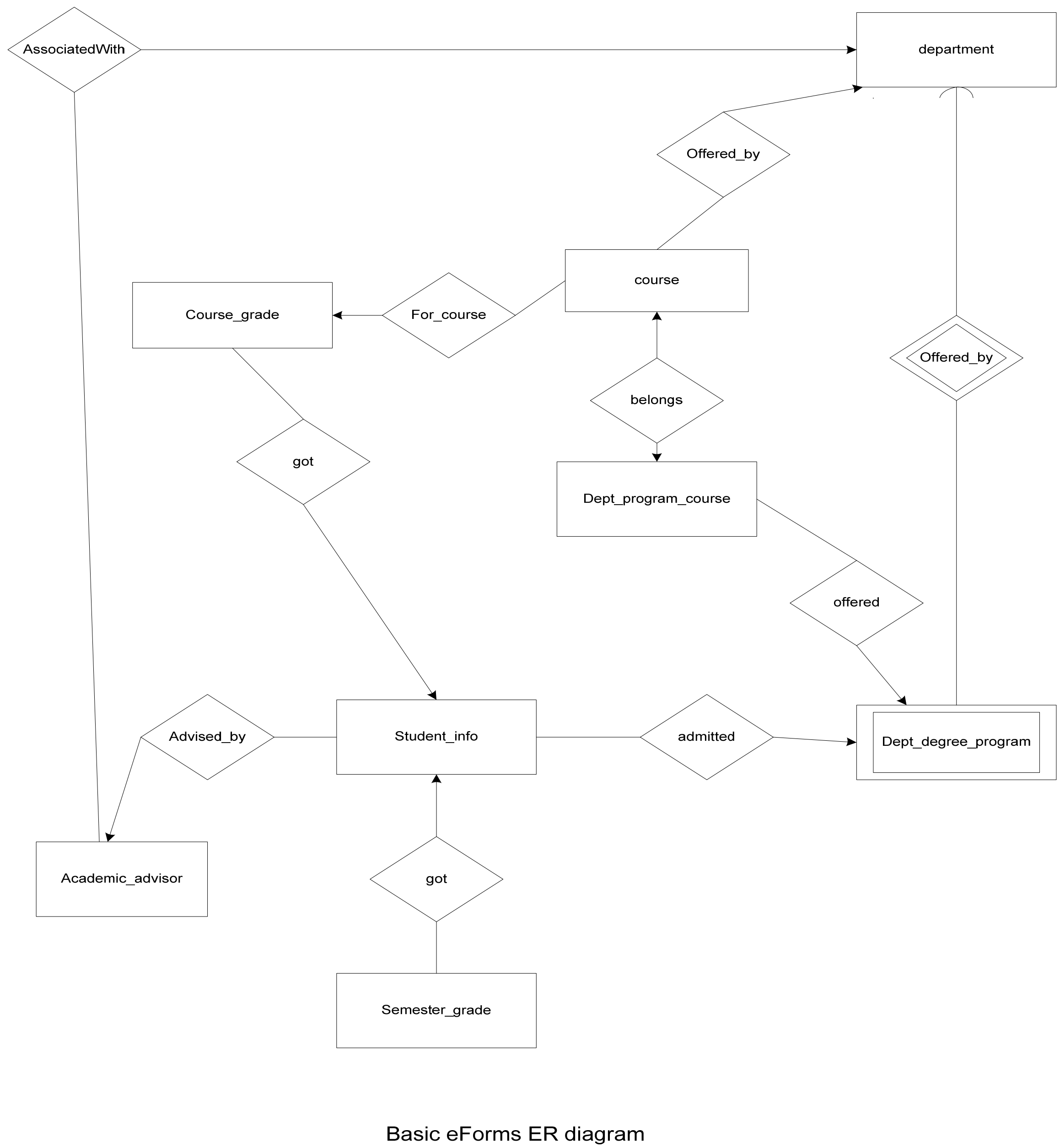


Figure 1: Basic eForms ER diagram

Major and Candidacy form ER diagram

Figure 2: Major and Candidacy form ER diagram

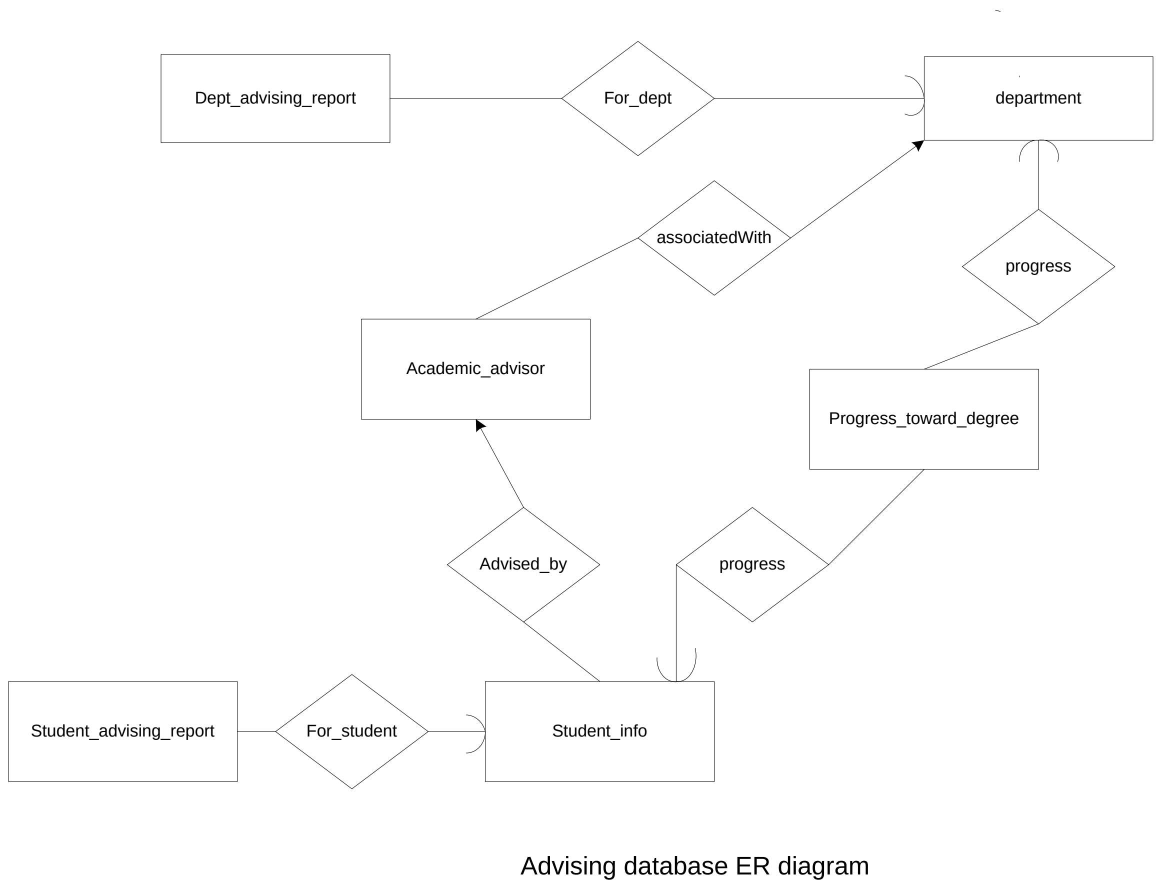


Figure 3: Advising database ER diagram

Please refer to appendix A for detailed fields in each table.

4.2 Relational Database Design

Entities are converted into relations also called tables and entity-attributes are converted into table fields. But this simple conversion could have many anomalies. For example many field values in multiple tuples could be same. This redundancy leads to wastage of storage space. Also there could be updation and deletion anomalies where changing information in one tuple still leaves unchanged information in other tuples. This could lead to inconsistent database. Relational tables are decomposed to eliminate these anomalies. Decomposition involves splitting the attributes of table into two or more tables. If the resulting decomposed tables satisfy normal form (BCNF) then it guarantees that there are no more such anomalies. We took greater care in making sure that our tables follow BCNF.

4.3 Design for Performance

Advising database is designed by keeping in mind the performance of typical queries which would be run more often. We decided to tune the database based on the following queries:

1. Give me GPA trend for specific student_id.
2. Give me list of all students whose GPA trend is negative.
3. Give me list of all students whose GPA trends are negative & will result in probation within (1, 2 …) semesters.
4. Give me list of students whose (common-courses or elective or major or GE or prep-for-major) GPA trend is negative.
5. Give me the GPA trend of students of a particular department.
6. Give me the average GPA trend of a particular department.
7. Give me list of all departments whose average (major or GE or overall) GPA trend is negative.

Based on above queries we decided to pre-compute and store three extra fields (Student's GPA trend, Student's overall GPA, Department's average GPA trend) into our advising database tables.

5.0 Implementation

5.1 Database Creation and Population

We created database tables based on our schema design. Advising database stores both non-derived as well as derived data. Non-derived data includes Student id, address, email, Department id, courses and student course grades etc. whereas derived data is the computed or estimated data. Derived data is student GPA trend and student semester grades which are derived from student course grades etc. This derived data is the key to figure out the appropriate conditions and the time when to advise a particular student. We wrote many programs to get non-derived data in memory, computed derived data from this in-memory data, also populated derived tables. We also wrote programs to fill dummy intelligent data which referred properly to all the ids in other tables. A short description of these java files is as follows:

Connect.java: to connect with the database.

Departments.java: reads departments data from the database. We need departmentId to be referred in other tables.

AcademicAdvisors.java: populates dummy data in academic_advisors table to be used later in triggers.

StudentCourseGrades.java: reads student course grade data from the database. This was the sanitized data which came from actual student records. Hence we needed to read this to get whatever student ids used here to be consistent with references from other tables.

StudentSemesterGrades.java: Computes and populates student semester grades data from student course grade data.

GenRAndom.java: to generate a random number or string between an input ranges. **StudentInfo.java:** to populate dummy data for student_info table. It uses in-memory structures in other java programs to get proper ids.

StudentAdvisingReport.java: to populate dummy data for student_advising_report table.

DeptAdvisingReport.java: to populate dummy data for dept_advising_report.

Queries.java: Runs each query 100 times and computes the average execution time of a query.

Please refer to appendix B for details and source code.

5.2 Report generation

Report generation mechanism involved figuring out various queries which might be useful once this system is deployed. After analyzing all the information, we figured out seven most frequent queries which one can run for the student advising system. We also measured the performance of each query. These queries are as follows:

Queries:

1. Give me GPA trend for specific student_id.

```
Select
        student_id, overall_gpa_trend
From
        student_advising_report
```

```
Where
        student_id = "' + studentId + "'";
```

2. Give me list of all students whose GPA trend is –ve.

```
Select
        Student_id, overall_gpa_trend
From
        student_advising_report
Where
        overall_gpa_trend < 0;
```

3. Give me list of all students whose GPA trend are –ve & will result in probation within (1, 2 …) semesters.

```
Select
        student_info.student_id, overall_gpa, overall_gpa_trend, overall_gpa_threshold,
        overall_gpa_intercept_threshold
from
        student_info, student_advising_report
where
        student_info.student_id = student_advising_report.student_id AND
        overall_gpa_trend < 0 AND
        overall_gpa + overall_gpa_trend * overall_gpa_intercept_threshold
                <= overall_gpa_threshold;
```

4. Give me list of students whose (common-courses or elective or major or GE or prep-for-major) GPA trend is –ve.

```
select
        student_id, overall_gpa_trend, ge_gpa_trend, major_course_trend
from
        student_advising_report
where overall_gpa_trend < 0 OR
        ge_gpa_trend < 0 OR
        major_course_trend < 0;
```

5. Give me the GPA trend of students of a particular department.

```
Select
        student_info.student_id, dept_id, overall_gpa_trend
from
        student_info, student_advising_report
where
```

```
        student_info.student_id = student_advising_report.student_id AND
        student_info.dept_id = '1355'
```

6. Give me the average GPA trend of a particular department.

```
Select
        name, dept_id, avg_overall_gpa_trend
from
        departments, dept_advising_report
where
        departments.id = dept_id AND dept_id = '1355';
```

7. Give me list of all departments whose average (major or GE or overall) GPA trend is –ve

```
select
        name, dept_id, avg_overall_gpa_trend, avg_major_gpa_trend, avg_ge_trend
from
        departments, dept_advising_report
where
        departments.id = dept_id AND
        (avg_overall_gpa_trend < 0 OR avg_major_gpa_trend < 0 OR
        avg_ge_trend < 0);
```

5.3 Performance Analysis of eForms Queries

We ran each query described above one hundred times to figure out the average execution time per query. Below are the results:

Table 1: Queries and their execution time

Query	Query Execution Time(ms)
Qry1	24
Qry2	95
Qry3	61
Qry4	188
Qry5	51
Qry6	16
Qry7	17

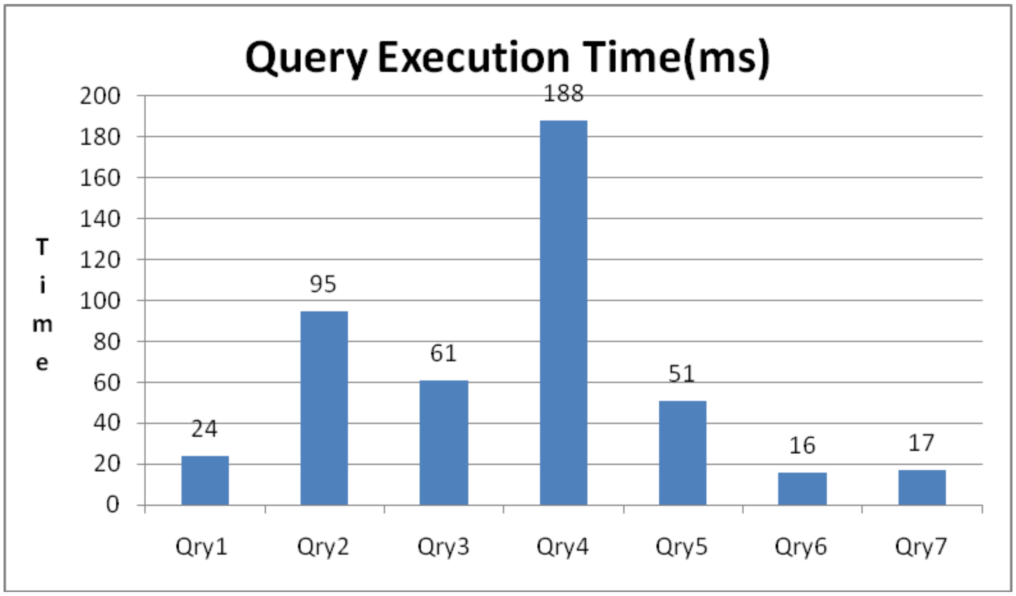


Figure 4: Query Execution Time

From above, query4 took the most time as number of records returned by this query is the most because of OR-ing of the conditions. Note that even the worst time which is taken by this query is less than One Second. Hence query performance is very reasonable.

5.4 Triggers and Automatic alerts

For the student advising system we need to alert student and his advisor whenever there is fear that student could go on probation. In such cases an automatic email will be sent to student and his advisor. We also created other triggers to automatically update student semester grades whenever course grade record was modified for better performance. A short description of these triggers is as follows:

1) **Trig_update_student_advising:** This trigger is triggered when student GPA trend is negative and will result in probation within intercept threshold semesters. Here when we detect such a student we send an email to both student as well as his or her advisor. We use

mysql's system command "\!" to call our script sendEmail.csh. sendEmail.csh script reads email parameters from /tmp/emails_eforms.txt file which is created by MySQL's SELECT INTO OUTFILE command.

2) **Trig_update_grade:** This trigger is triggered when student grade (gr) is updated for some course. We compute the gpa from the course grade, update gp of the course. And also update the corresponding semester's gpa and gp.

3) **Trig_insert_grade:** This trigger is triggered when a new student course record is inserted in the database. If this is the first course then we need to insert a new student semester record. Otherwise we update the existing student semester record. To detect whether this is first course record or not, we use MySQL's exception handler "NOT FOUND".

4) **Trig_delete_grade:** This trigger is triggered when a student course record is deleted. We update the corresponding student semester grade. If last course record is removed we delete the corresponding semester grade record also.

Please refer to appendix C for details and source code.

6.0 Economic Justification

Objective of this section is to justify project importance, identify potential customers, competitors, benefits to the customers, economic benefits and the cost analysis.

6.1 Executive Summary

UniEforms System Inc. will be established in January 2010 as a sole proprietorship organization. Main products of our company will be the software systems for Universities to convert paper form processing into online forms processing and online Student Advising system. To create such systems our company needs capital around $1.3 million. 60% of the capital investment will come from the investors. We will reach the break-even point in the 7th quarter after deployment i.e. third quarter of 2011. Our ROI will reach up to 50% within one year after break-even point. Our primary customer is San Jose State University and once system is deployed and tested, our customers will be any academic institution. The company will license eForms system to other universities for at least $240000 and targets to serve four universities within the next two years of its deployment. The price point is calculated from the expenses and the paper cost analysis. From our analysis, paper form cost is around half a million per year which is double the money we charge for our eForms system. Hence it’s a great advantage for the Universities to switch to our eForms system to save 50% of the paper cost. Presently we have one CTO (Chief Technical Officer) and five developers. But once the development is done we will need one CEO and one Marketing and Sales head along with our current employees.

According to our research, eForms automatic fillable major and candidacy forms and student advising system is new to the universities. Currently there are no direct competitors in this space. Note that overall eforms usage and market is growing, which means competition will increase rapidly. But as the early starters we will be well placed to take good advantage of this growing market.

6.2 Problem Statement

Currently, the student form handling process at SJSU is very inefficient which increases frustration among students and faculty. Students have to manually submit forms, and feedback about the form is provided when the student picks up the form, receives an email from the advisor, via postal service, or from the student Registrar's office. If there is some mistake, this whole process repeats. So this is a time consuming, inefficient, and error prone process. In addition, there is limited tracking of student success. In SJSU, the average six year undergraduate graduation rate is approximately 20 %(Emily, Allen.2009). This can definitely be improved if students and their advisors are informed about potential problems or delays in academic progress in a more timely and preemptive manner.

6.3 Solution and Value Proposition

The above stated problem can definitely be improved if students and their advisors are informed about potential problems or delays in academic progress in a more timely and preemptive manner. The eForms project intends to develop a system to automatically complete major and candidacy forms. It ensures that error-free forms are submitted to the university on time and eliminates the need for students to remember when the forms need to be submitted.

6.4 Market Size

Nowadays almost 80% of the processes in public and private businesses depend on eforms. But our scope is limited only to educational institutes. In initial few years, we'll target Universities in USA and after analyzing the results we can deploy our system in other parts of the world. "According to UNESCO the US has the second largest number of higher education institutions in the world, with a total of 5,758, an average of more than 115 per state. "

(http://en.wikipedia.org/wiki/Higher_education_in_the_United_States). So this implies the US market size is around $1.38 Billion.

6.5 Competitors

We have researched various publications about systems, designs, capabilities and commercial products that support the academic processing of forms. To date we have not found a system that provides the same functionality as our proposed eForms systems. Some of the vendors which provide eForms solutions are as follows:

Table 2: Representative eForms Vendors

Some Representative eForms Vendors	
Company	**Comments**
Cardiff, http://www.cardiff.com	Established provider of eForms solutions; maker of a compre-hensive eForms solution, LiquidOffice.
Adobe, http://www.adobe.com	A growing presence in the eForms space through Acrobat and acquisition of Accelio.
Microsoft, http://office.microsoft.com/infopath	A newly focused presence in the eForms space with the intro-duction of InfoPath
PureEdge Solutions, http://www.pureedge.com/	Established provider of eForms solutions. Along with Cardiff, has taken active role in development of Xforms specification.
Texcel Systems, http://www.texcel.com	Smaller but growing company that provides tools for convert-ing various legacy document formats into eForms.
Mosquito orgabit GmbH, http://mozquito.markuplanguage.net	mozquito is the maker of the DENG XBrowser. DENG is a modu-lar XML browser, that supports XForms, SVG, XHTML, arbitrary XML with CSS, XFrames and any other custom XML namespace.
ScanSoft, http://www.scansoft.com/	ScanSoft is a supplier of productivity software with concentra-tions in imaging, speech, and language solutions. Their Omni-Page and PDF Converter tools are used for converting documents to various eForm formats.

http://www.adobe.com/enterprise/pdfs/gilbane_eforms.pdf

However these eforms vendors just provide generalized eForm components, but don't provide the customized complete solution as we desire. Our solution requires automatically filling of forms without student intervention. We didn't even find any University using such a system. Also we didn't find any University providing online student advising system. None of these vendors provide such a solution either. Thus, our eForms system does not have any direct competitors.

6.6 Customers

Academic institutions are our main customers. Because of greater cooperation, investment and relationship with San Jose State University during development, we will be initially deploying our system at San Jose State University. Thereafter we will deploy the system at other universities. As the system is scalable, we don't foresee much effort in getting it deployed at other universities. In initial few years, we'll target Universities in USA and after analyzing the results we can deploy our system in other parts of the world.

6.7 Cost Analysis

6.7.1 Paper form cost

- Estimating that most students live within 5 miles of campus, average student spends around \$3-\$5 in gas, oil and wear and tear on the vehicle per form submission. (source: Dr. Leonard Wesley, SJSU)
- It takes around 1hr of student's time to prepare and drop off a form. Let student's time is \$5/hr. Also student has to do this on average, multiple times in order to correct errors in submitted forms. (source: Dr. Leonard Wesley, SJSU)

- An advisor takes 15 minutes on average to process a candidacy or major form, at an average burdened hourly rate of $50/hr. (source: Dr. Leonard Wesley, SJSU) Also advisors have to do this multiple times because students frequently need to submit forms multiple times(For calculation on an average for multiple times, we will use twice(2)).

Overall there will be following significant paper form cost factors:

Table 3: Paper Cost

Type of cost	**Amount per paper**
Oil, gas cost	$3
Student time cost	$5
Advisor time cost	$13
Paper	1c
file storage/cabinets	$500 for 5 drawer cabinet. Total around $20k for 40 cabinets. i.e. around $1k per year assuming cabinets are used for 20 years. This implies around $0.5 per paper assuming 2000 forms per year.
Printer/copy machines	Around 10c
Printer/copier toners	Around 3c
Filing and office supplies	Around 1c
Rent	Average office rent space is around $15-$20 per square foot. i.e. around $6k per month for 400sq feet. This implies per paper cost = $6k * 12/2000 = $36
Employee salary	2 employees to manage this at the rate of $50k i.e. $100k

	per year. This implies $50 per paper.

Thus a paper costs around $3 + $5 + $13 + $0.01 + $0.5 + $0.1 + $0.03 + $0.1 + $36 + $50 = 107.74. Assuming two papers per form considering there would be errors during filling of forms. Total cost per form = $215.48. For an average of 250 candidacy and major forms per department and for 8 departments, we need total of 2000 forms per year.

Thus total cost of 2000 paper forms per year = $430960.

6.7.2 eForms cost

6.7.2.1 Fixed Cost

Table 4: Fixed Cost

Computers/Laptops	$800 * number of units (8) = $6400
Server Cost	$3000 * number of servers (2) = $6000
Office Furniture	$4000
Software Cost	$0 as all open source code is used
Installation/Setup cost	$1000
Total	$17400

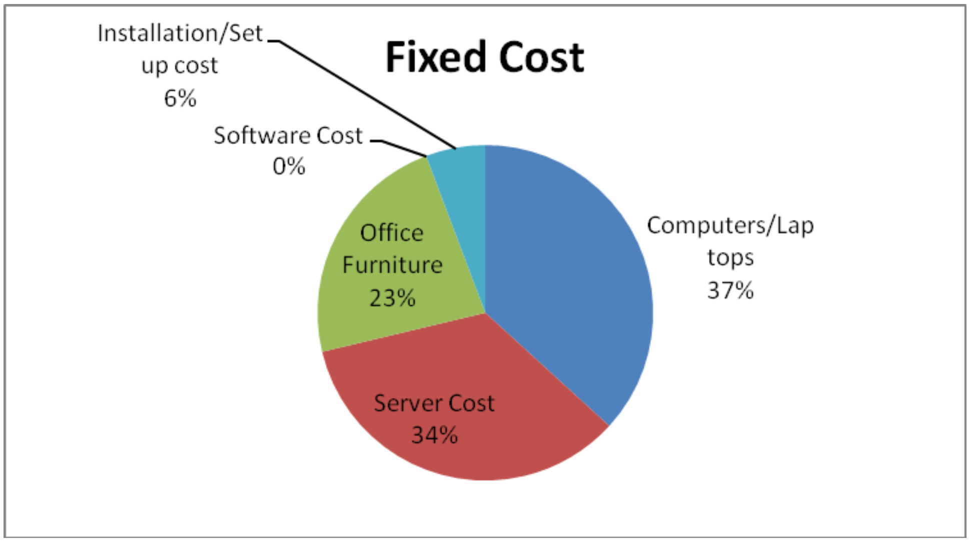


Figure 5: Fixed Cost

6.7.2.2 Variable cost

For the initial one year, we are in the development mode. During this time we won't spend money on advertisement and marketing. Also we do not need CEO till the very end of the year. We will need five developers at the cost of $40/hr per developer, each developer working for at least 20hrs/week. After the development is done, we will need a Marketing head and a CEO, but we won't spend much money on the development. To maintain and deploy our system at multiple universities, the number of developers will remain more or less the same.

Table 5: Variable Cost

	2009	2010	2011	2012
CEO	$20000	$100000	$105000	$120000
Marketing and Sales Head	$0	$80000	$90000	$100000
Advertisement	$0	$1000	$3000	$3000
CTO	$80000	$85000	$90000	$105000
Developers	$208000	$208000	$210000	$215000
Office Rent	$36000	$40000	$40000	$40000
Electricity	$1000	$1000	$1000	$1000
Server maintenance cost	$1000	$1000	$1000	$1000
System admin time	$15000	$15000	$15000	$15000
Internet/Telephone	$1200	$1200	$1200	$1200
Total	**$362200**	**$532200**	**$556200**	**$601200**

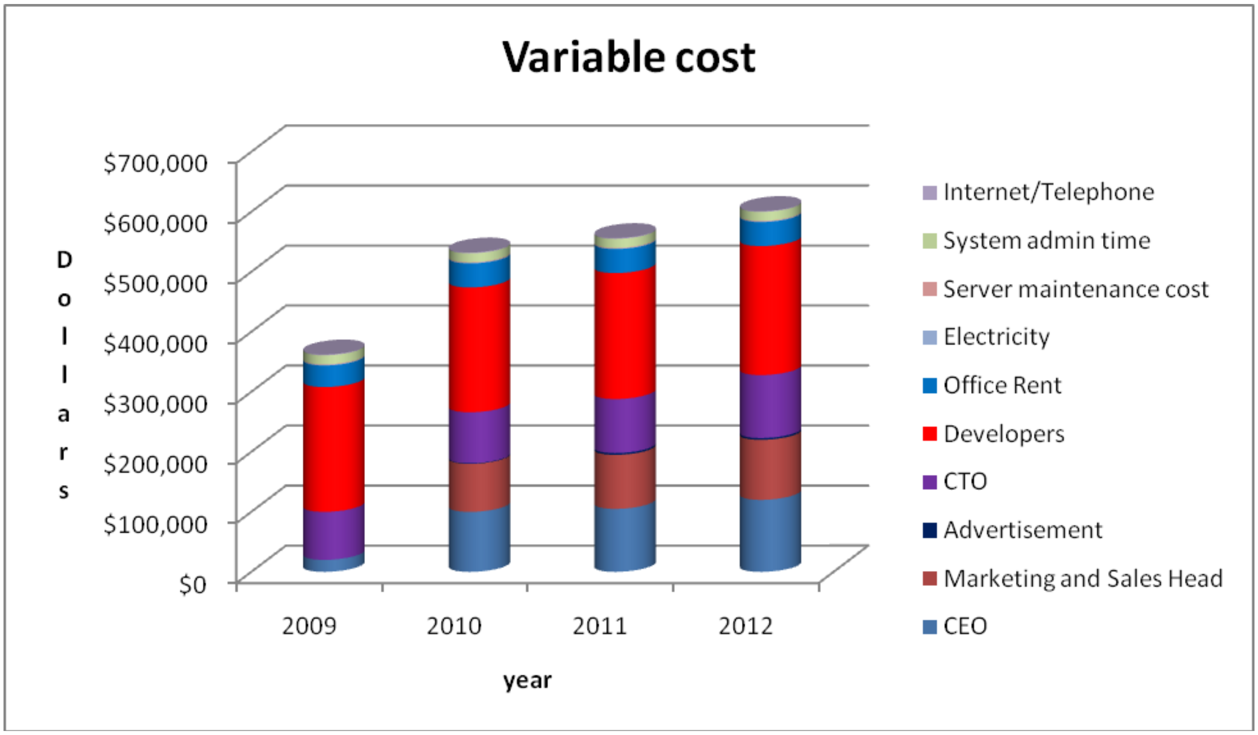


Figure 6: Variable Cost

Based on our paper form cost calculations we project to license our system for at least $240000 per year to a University after development is over. This is quite plausible as by switching to our system, University's form cost will reduce more than 50%. Also we project to grow faster in initial years by selling this system to 4 universities by the end of second year.

6.8 Price Point

The price point is calculated from the expenses and the paper cost analysis. Based on this analysis, the company will sell the license to other universities for at least $240000 which can be renewed every year.

6.9 SWOT Analysis

For better understanding of the market we need to identify its strengths, weaknesses, opportunities and threats. SWOT analysis of eForms system is as follows:

Table 6: SWOT Analysis

Strengths	Weaknesses
1. The advising database and major and candidacy forms to be filled automatically are new and unique concepts. 2. There are no direct competitors as of now.	1. It is difficult to accurately assess the market demand as it is new and unique. 2. During transition some of the paper forms might require manual conversion to eforms.

Opportunities	Threats
1. As taking into consideration of global warming and other environmental factors, eforms demand will increase. Also there could be government backing too. 2. As there is no direct competition, our system has the potential to grab 100% market share.	1. Oracle or some other provider may come up with better solution. 2. It could be risky to many investors to invest in a small startup when economy is under recession.

6.10 Investment Capital Requirements

The company will break even in the 7th quarter after deployment i.e. third quarter of 2011 and before that it requires $1.3M of capital investment.

6.11 Return on Investment

It is wise to invest in a business which provides high Returns. The ROI for the eForms project can be calculated using the following formula:

ROI = [(Total Revenue – Total Expense)/Total Expense]*100

Table 7: Return on Investment

Quarter	Total Revenue	Total Expense	ROI	% ROI
Q1’09	$0	$107950	-$107950	-100
Q2’09	$0	$198500	-$198500	-100
Q3’09	$0	$289050	-$289050	-100
Q4’09	$0	$379600	-$379600	-100
Q1’10	$120000	$512650	-$392650	-76.59
Q2’10	$240000	$645700	-$405700	-62.83
Q3’10	$360000	$778750	-$418750	-53.77
Q4’10	$480000	$911800	-$431800	-47.36
Q1’11	$780000	$1050850	-$270850	-25.77
Q2’11	$1080000	$1189900	-$109900	-9.23
Q3’11	$1380000	$1328950	$51050	3.84
Q4’11	$1680000	$1468000	$212000	14.44
Q1’12	$2160000	$1618300	$541700	33.47
Q2’12	$2640000	$1768600	$871400	49.27

This graph shows the %age ROI.

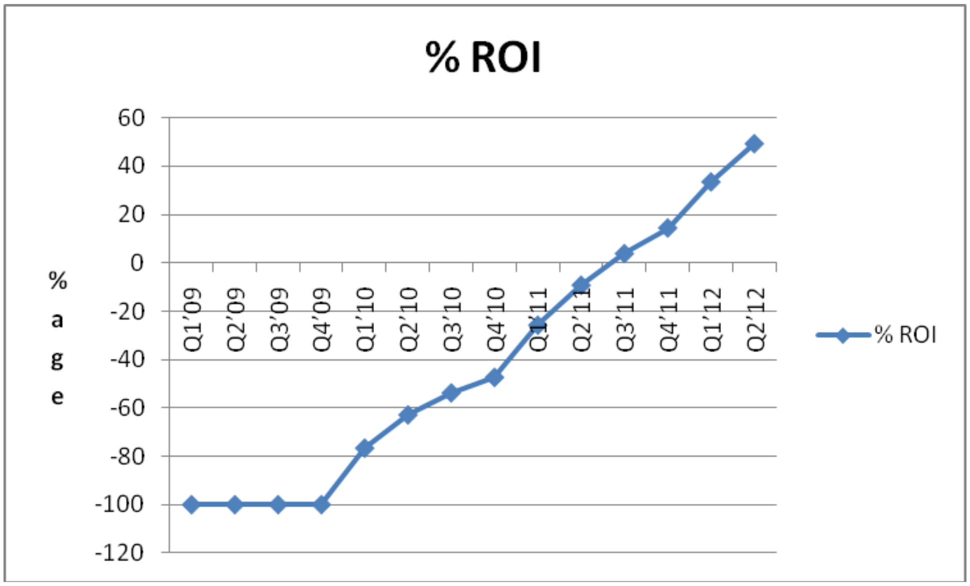


Figure 7: Return on Investment

This shows that ROI will reach upto 50% at the end of 14th quarter i.e. 2nd quarter of 2012 which is a positive sign for the benefit to investors and the customers.

6.12 Personnel

The company is started in the first quarter of 2009 with one CTO and five developers. CEO will be hired sometime in the last quarter of 2009. Once the system is deployed, that is in the first quarter of 2010 we'll also need Marketing and Sales head.

6.13 Business and Revenue Model

The main aim of any business is to make profit. As the company intends to sell the solution to other Universities, the aim of advertisements and the marketing head will be to convince other Universities about how our system can help their University and save them thousands of dollars.

The company will license its system to other universities for at least $240000 and our target is to serve four universities within two years of its deployment. License can be renewed every year.

6.14 Strategic Alliance/Partners

Currently we plan to target our solution only for the Universities. In future we will form strategic alliance with other players serving areas other than technical institutions. We can also form alliance to combine our solution with generic solutions from big companies like Adobe or Oracle to tap their marketing power.

6.15 Profit & Loss

To check the worth of investment in a project, a company needs to analyze and forecast its profit and loss. The eForms project will be deployed in the first quarter of 2010. Total cost of the project is the sum of fixed cost and variable cost. Total revenue of the project is the money we receive from universities when we license our software. Profit/Loss is calculated by subtracting total revenue from total expense. All the negative values show loss and the positive values show profit. The quarterly distribution of Income, Expense and the Profit/Loss is shown in the following graph:

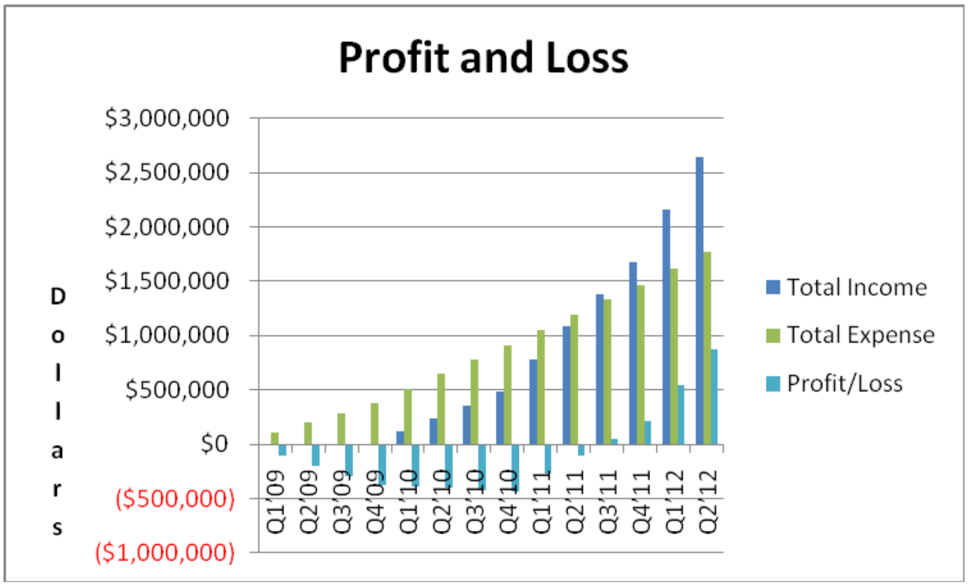


Figure 8: Quarterly Distribution of Profit/Loss

We will break even in third quarter of 2011 as shown in the following chart:

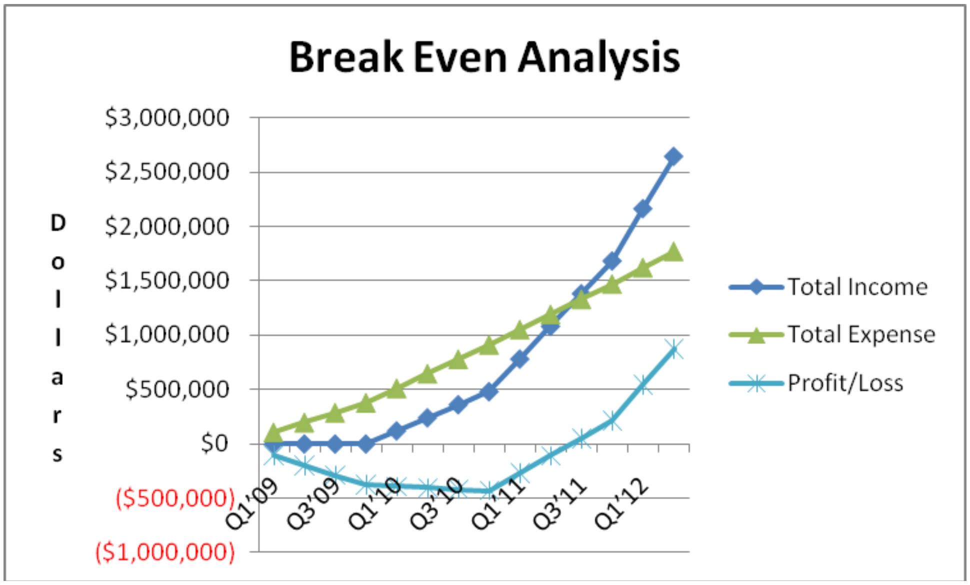


Figure 9: Break Even Analysis

From the above breakeven figure, you can see that our capital requirement is $1.3Million. We estimate low risk (a = 0.2) for our project as technical expertise and skills required for this project are easily available. We don't have any existing competitors in the same space. Also we don't have to pay any license fee for the software used to develop this system. Below is the Norden Rayleigh model for our system based on these parameters:

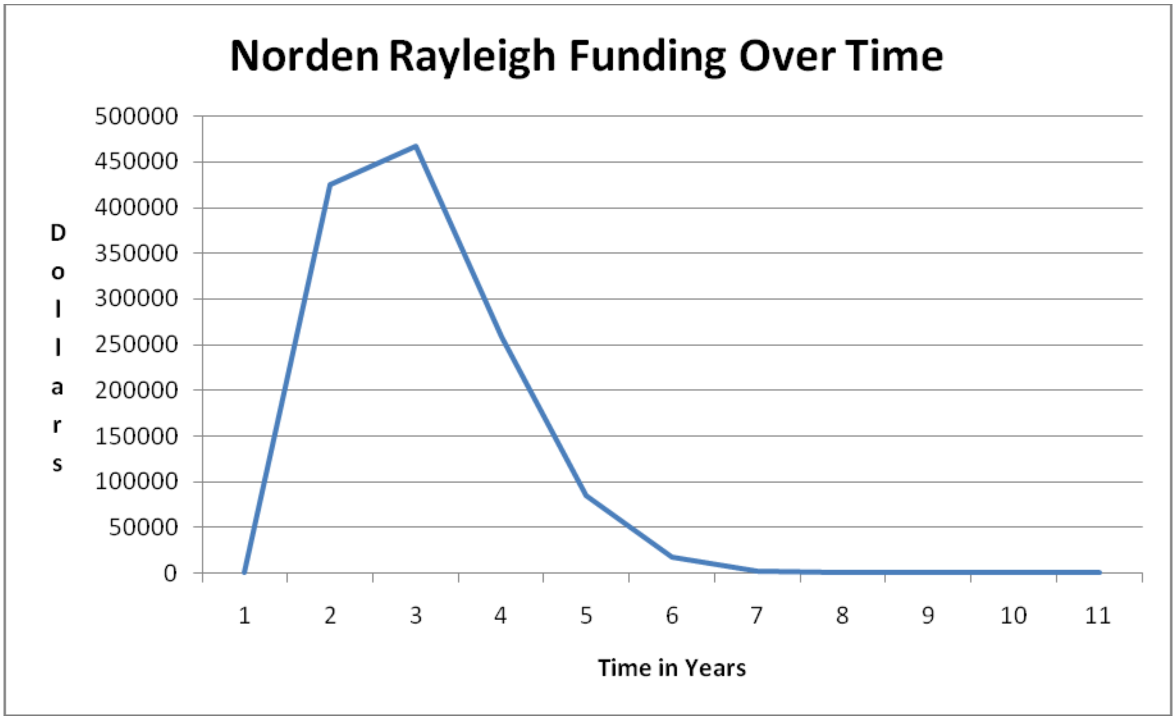


Figure 10: Norden Rayleigh Funding Profile Over Time

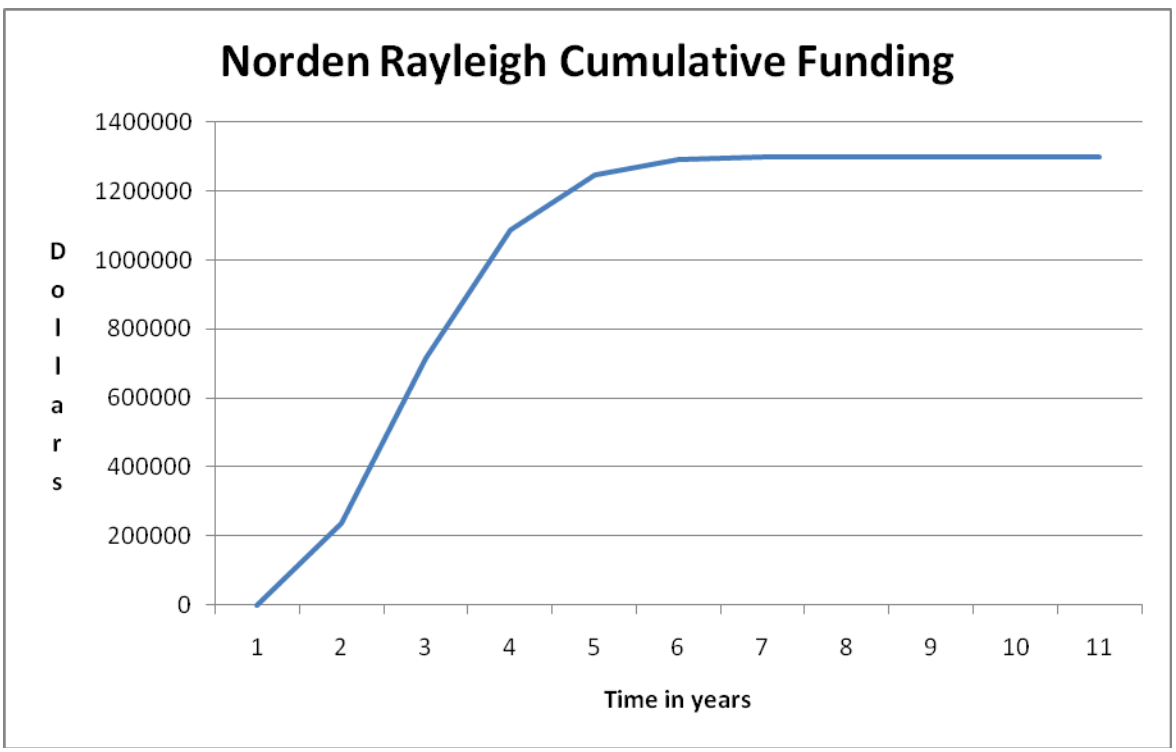


Figure 11: Norden Rayleigh Cumulative Funding Over Time

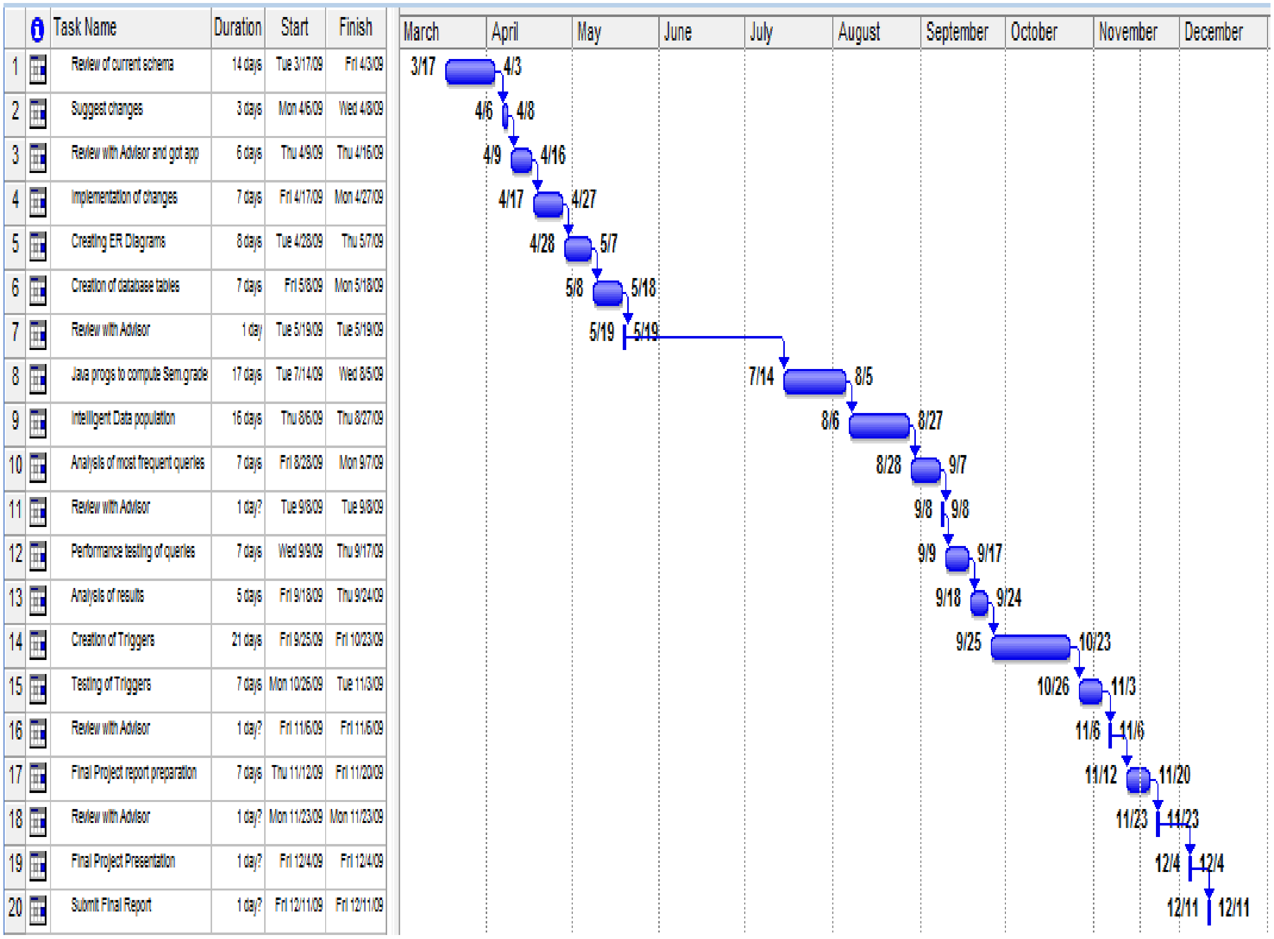
6.16 Exit Strategy

Our exit strategy is to get acquired by a big player in the eForms space. Our company is projected to grow faster in the initial years. We plan to sell this system to at least four Universities by the end of second year. We will have positive cash flow by then, which will make the company very attractive for acquisitions. Big companies like Adobe or Oracle would find it in their interest to acquire our company to expand their market reach.

6.17 Future Work

Poor planning is the reason why many students are not able to complete their graduation on time. One factor in poor planning is the non-availability of future course offerings. Due to course cancellations or full enrollments, students are sometimes left with several required courses for their last semester. Near the end of their academic career, they come to know that these courses are not offered together in one semester. Thus future work involves obtaining course offering data from various departments. Then design and implement database to store this information in order to help advice students with respect to when and which degree program courses to enroll in.

7.0 Project Schedule



		Task Name	Duration	Start	Finish
1		Review of current schema	14 days	Tue 3/17/09	Fri 4/3/09
2		Suggest changes	3 days	Mon 4/6/09	Wed 4/8/09
3		Review with Advisor and got app	6 days	Thu 4/9/09	Thu 4/16/09
4		Implementation of changes	7 days	Fri 4/17/09	Mon 4/27/09
5		Creating ER Diagrams	8 days	Tue 4/28/09	Thu 5/7/09
6		Creation of database tables	7 days	Fri 5/8/09	Mon 5/18/09
7		Review with Advisor	1 day	Tue 5/19/09	Tue 5/19/09
8		Java progs to compute Sem.grade	17 days	Tue 7/14/09	Wed 8/5/09
9		Intelligent Data population	16 days	Thu 8/6/09	Thu 8/27/09
10		Analysis of most frequent queries	7 days	Fri 8/28/09	Mon 9/7/09
11		Review with Advisor	1 day?	Tue 9/8/09	Tue 9/8/09
12		Performance testing of queries	7 days	Wed 9/9/09	Thu 9/17/09
13		Analysis of results	5 days	Fri 9/18/09	Thu 9/24/09
14		Creation of Triggers	21 days	Fri 9/25/09	Fri 10/23/09
15		Testing of Triggers	7 days	Mon 10/26/09	Tue 11/3/09
16		Review with Advisor	1 day?	Fri 11/6/09	Fri 11/6/09
17		Final Project report preparation	7 days	Thu 11/12/09	Fri 11/20/09
18		Review with Advisor	1 day?	Mon 11/23/09	Mon 11/23/09
19		Final Project Presentation	1 day?	Fri 12/4/09	Fri 12/4/09
20		Submit Final Report	1 day?	Fri 12/11/09	Fri 12/11/09

Figure 12: Project Schedule

8.0 Conclusion

In nutshell, eForms system has many advantages over paper form system. And technologically it's feasible to implement such a system. From our economic justification, we will reach breakeven point in third quarter of 2011. Within 2.5 years we can reach ROI of 50%. Having automatic form submission, advising database and report generation mechanism is a Win-Win situation for both the University and the students.

References

Dr. Leonard P. Wesley. (2007). Associate Professor Computer Engineering Department, MSE Director. San Jose State University.

The White House (2003). *Office of Management and budget*. Retrieved August 03, 2009, from web site:
http://www.whitehouse.gov/omb/fedreg_gpea2

The Gilbane Report. (2003). *Microsoft, Adobe & XForms to shake up electronic forms market.* Retrieved August 22, 2009, from web site:
http://www.adobe.com/enterprise/pdfs/gilbane_eforms.pdf

Wikipedia (2009). *Higher education in the United States.* Retrieved August 22, 2009, from web site:
http://en.wikipedia.org/wiki/Higher_education_in_the_United_States

Verity LiquidOffice (2004). *Business Process Automation*. Retrieved February 27, 2009, from web site:
http://www.syscomservices.com/mainmenu/product/cardiffliquidoffice/LiquidOfficeOverview.aspx

IBM FORUM (2006). *Results through Innovation*. Retrieved July 13, 2009, from web site:
http://www-07.ibm.com/events/nz/forum06/pressos/pdf/1430_Improving_business_controls_with_Workplace_technologies.pdf

Orfali, Robert; Harkey, Dan; Edwards, Jeri (1999). *Client/Server Survival Guide* (3rd ed). New York, Chichester, Weinheim, Brisbane, Singapore & Toronto.

Garcia-Molina, Hector; Ullman, Jeffrey D; Widom, Jennifer (2002). *Database Systems* (2nd ed).

DeFurio, Lori; Forrest, Steve (n.d.). *Adobe eForm Solutions*. Retrieved August 1, 2009, from web site:
http://www.planetpdf.com/planetpdf/pdfs/pdf2k/02E/ldefurio_pdfforms.pdf

Xigla Software (2008). *Absolute Form Processor XE*. Retrieved March 9, 2009, from web site:
http://www.xigla.com/absolutefp/

Whitmore, Joseph (2007). *E-Forms Technology: Past, Present and Future*. Retrieved March 1, 2009, from web site:
http://www.nysforum.org/documents/pdf/2007/et/mobiledigitalforms/eforms.pdf

Trippe, Bill (2004). *The State of the eForms Market*. Retrieved May 1, 2009, from web site:

http://www.nmpub.com/presentations/seyboldeuropeeformstrippe.pdf

Pacific University (2007). *Class Climate*. Retrieved March 19, 2009, from web site: http://www.scantron.com/downloads/Pacific%20Univ.%20Success%20Story.pdf

IBM (2005). *IBM Workplace: Strategy, Launch, Overview*. Retrieved March 19, 2009, from web site:
http://www.vidra.co.yu/pdf/IntroPR.pdf

Mandal, S.; Chowdhury, S.P.; Das, A.K.; Chanda, B. (2005). *Document Analysis and Recognition,* IEEE. A hierarchical method for automated identification and segmentation of forms, Volume 2, Pages 705-709.

Anders Tornqvist, Chris Nelson, Mats Johnsson. (1999). *XML and Objects-The Future for E-Forms on the Web, IEEE.* Pages 303-308.

Boehm, B. (2002). *Safe and simple software cost analysis*, IEEE. Volume 17, issue 5, Pages 14-17.

Dai, Zhoulin; Gu, Yi; Liu, Jun; Xu Yi Jie. (2007). *Computer Software and Applications Conference,* IEEE. Experiences in Accurately Estimating Electronic Forms Conversion Services with a Spiral Estimate Process, Volume 2, Pages 497-500.

Tornqvist, A; Nelson, C; Johnson, M. (1999). Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE. XML and objects-the future of the e-forms on the web, Pages 303-308.

Vassilakis, Costas; Lepouras, George; Rouvas, Stathis; Georgiadis, Panagiotis. (2005). I*nternational Journal of Web Engineering and Technology,* Inderscience. Exploiting form semantics and validation checks to improve e-form layout, Volume 2, Issue 1, Pages 81-96.

Kai, Luo; Shahram, Latifi; Kazem, Taghva; Emma, Regentova. (2004). *Document Analysis and Recognition*, IEEE. A hierarchical method for automated identification and segmentation of forms, Volume 2, Pages 705-709.

A primer on electronic document security (2007). Retrieved March 19, 2009, from web site:
http://www.adobe.com/security/pdfs/acrobat_livecycle_security_wp.pdf

Office of Institutional Research (OIR) report that was generated for the period 2002 to 2008

Emily, Allen. (2009). C.W.Davidson College of Engineering, Council of Chairs Meeting Presentation.

http://www.evenlogic.co.uk/PDF_files/eforms_brochure.pdf

Cardiff LiquidOffice (2006). *eForms*. Retrieved February 27, 2009, from web site:
http://www.alcom.se/liquidoffice/PDF/MK0756_LO_eforms_Alcom.pdf

Verity LiquidOffice (2004). *Business Process Automation*. Retrieved February 27, 2009, from web site:
http://www.syscomservices.com/mainmenu/product/cardiffliquidoffice/LiquidOfficeOverview.aspx

FileNet (2005). *Forms Manager*. Retrieved February 27, 2009, from web site:
http://www.aksis.com.tr/d/forms_manager_brochure.pdf

Acrobat (2009). *Adobe Acrobat eForms*. Retrieved February 28, 2009, from web site:
http://www.adobe.com/products/acrobat/eforms.html

Monmouth University (2009). *Advising*. Retrieved March 7, 2009, from web site:
http://www.monmouth.edu/academics/communication/advising.asp

DRS (2008). *Education Forms Processing*. Retrieved March 7, 2009, from web site:
http://www.drs.co.uk/education_forms_processing.html

University of California, Riverside (2008). *eForms.* Retrieved March 7, 2009, from web site:
http://cnc.ucr.edu/eforms/

University of California, Riverside (2008). *Currently Available eForms.* Retrieved March 7, 2009, from web site:
http://cnc.ucr.edu/eforms/access.html

Scantron (2007). *Class Climate*. Retrieved March 9, 2009, from web site:
http://www.scantron.com/classclimate/

California Department of Motor Vehicles (2009). *Forms Online*. Retrieved March 9, 2009, from web site:
http://dmv.ca.gov/forms/forms.htm

Internal Revenue Service (2009). *Forms and Publications*. Retrieved March 9, 2009, from web site:
http://www.irs.gov/formspubs/index.html

State of California (2009). *Franchise Tax Board*. Retrieved March 15, 2009, from web site:
http://ftb.ca. gov

Virtual Medical Worlds (2004). *Verity LiquidOffice at heart of eForms strategy implemented by Ottawa Hospital.* Retrieved March 9, 2009, from web site:
http://www.hoise.com/vmw/04/articles/vmw/LV-VM-06-04-19.html

Catherine, Edwin St. (2003). *The eForm Revolution: Implications for the United Nations Demographic Yearbook*. Retrieved March 1, 2009, from web site: http://unstats.un.org/unsd/demographic/meetings/egm/DYB_1103/docs/no_18.pdf

APPENDIX A

Database Tables

```
mysql> source /home/shifali/project/showTables.sql
+----------------------------------+
| Tables_in_eforms                 |
+----------------------------------+
| academic_advisors                |
| candidacy_form_templates         |
| courses                          |
| departments                      |
| dept_advising_report             |
| dept_degree_programs             |
| dept_program_courses             |
| login_privileges                 |
| major_allowed_courses            |
| major_form_templates             |
| progress_toward_degree           |
| semester_year                    |
| student_advising_report          |
| student_candidacy_form           |
| student_course_grade             |
| student_info                     |
| student_major_form               |
| student_major_form_courses       |
| student_semester_grade           |
| student_tech_elective_exceptions |
| user_login_info                  |
+----------------------------------+
21 rows in set (0.01 sec)

| academic_advisors                |
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| advisor_id    | bigint(20)  | NO   | PRI |         |       |
| dept_id       | varchar(10) | YES  |     | NULL    |       |
| first_name    | varchar(80) | YES  |     | NULL    |       |
| last_name     | varchar(80) | NO   |     |         |       |
| email         | varchar (80)| YES  |     | NULL    |       |
| phone         | int(20)     | YES  |     | NULL    |       |
| room_location | int(20)     | YES  |     | NULL    |       |
```

```
+---------------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

| candidacy_form_templates        |
+--------------+---------------------+------+-----+---------+----------------+
| Field        | Type                | Null | Key | Default | Extra          |
+--------------+---------------------+------+-----+---------+----------------+
| id           | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| sem_yr_id    | varchar(4)          | NO   | MUL |         |                |
| template     | text                | NO   |     |         |                |
| dept_id      | varchar(10)         | NO   |     |         |                |
| program_name | varchar(40)         | NO   |     |         |                |
| comment      | varchar(100)        | YES  |     | NULL    |                |
+--------------+---------------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

| courses                         |
+------------------------+--------------+------+-----+---------+-------+
| Field                  | Type         | Null | Key | Default | Extra |
+------------------------+--------------+------+-----+---------+-------+
| id                     | varchar(6)   | NO   | PRI |         |       |
| class_class_nbr        | int(19)      | NO   | PRI |         |       |
| abv                    | varchar(10)  | YES  |     | NULL    |       |
| number                 | varchar(10)  | YES  |     | NULL    |       |
| title                  | varchar(100) | YES  |     | NULL    |       |
| units                  | float        | YES  |     | NULL    |       |
| dept_id                | varchar(10)  | NO   |     |         |       |
| sem_yr_id              | varchar(4)   | YES  |     | NULL    |       |
| ge_equiv_units         | varchar(3)   | NO   |     |         |       |
| cross_listed_course_id | varchar(6)   | YES  |     | NULL    |       |
+------------------------+--------------+------+-----+---------+-------+
10 rows in set (0.01 sec)

| departments                     |
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | varchar(10) | NO   | PRI |         |       |
| name  | varchar(30) | YES  |     | NULL    |       |
| abv   | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

| dept_advising_report            |
+------------------------------------+---------------------+------+-----+---------+----------------+
```

```
| Field                              | Type                | Null | Key | Default | Extra          |
+------------------------------------+---------------------+------+-----+---------+----------------+
| id                                 | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| dept_id                            | varchar(10)         | YES  |     | NULL    |                |
| major_id                           | bigint(20) unsigned | NO   |     |         |                |
| avg_overall_gpa_trend              | float               | NO   |     |         |                |
| avg_overall_gpa_intercept_threshold | float              | NO   |     |         |                |
| report_period_start_semester       | bigint(20) unsigned | NO   |     |         |                |
| report_period_start_yr             | char(4)             | NO   |     |         |                |
| report_period_end_semester         | bigint(20) unsigned | NO   |     |         |                |
| report_period_end_yr               | char(4)             | NO   |     |         |                |
| avg_major_gpa_trend                | float               | NO   |     |         |                |
| avg_major_gpa_intercept_threshold  | float               | NO   |     |         |                |
| avg_ge_trend                       | float               | NO   |     |         |                |
| avg_major_course_trend             | float               | NO   |     |         |                |
| avg_prep_for_major_trend           | float               | NO   |     |         |                |
| avg_common_courses_trend           | float               | NO   |     |         |                |
+------------------------------------+---------------------+------+-----+---------+----------------+
15 rows in set (0.00 sec)

| dept_degree_programs          |
+-------------------------------+---------------------+------+-----+---------+----------------+
| Field                         | Type                | Null | Key | Default | Extra          |
+-------------------------------+---------------------+------+-----+---------+----------------+
| id                            | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| dept_id                       | varchar(10)         | YES  |     | NULL    |                |
| degree                        | varchar(10)         | NO   |     |         |                |
| program_name                  | varchar(40)         | NO   |     |         |                |
| degree_program_type           | varchar(40)         | YES  |     | NULL    |                |
| degree_program_name_cip_code  | varchar(13)         | YES  |     | NULL    |                |
| sem_yr_id                     | varchar(4)          | YES  |     | NULL    |                |
| min_major_gpa                 | float               | YES  |     | NULL    |                |
+-------------------------------+---------------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)

| dept_program_courses          |
+------------------------+----------------------------------------------------------------------------------------
---+------+-----+---------+----------------+
| Field                  | Type                                                                                   | Null | Key |
Default | Extra          |
+------------------------+----------------------------------------------------------------------------------------
---+------+-----+---------+----------------+
| id                     | bigint(20) unsigned                                                                    | NO   | PRI |
NULL    | auto_increment |
```

```
| dept_degree_program_id | bigint(20) unsigned | NO
| MUL | | |
| course_id | varchar(6) | YES | |
NULL | |
| course_type |
enum('MAJOR_COMMON_CORE','MAJOR_REQUIRED','MAJOR_ELECTIVE','PREP_FOR
_MAJOR','MAJOR_OTHER') | YES | | NULL | |
+------------------------+----------------------------------------------------------------------------------------
---+------+-----+---------+----------------+
4 rows in set (0.01 sec)

| login_privileges |
+--------------------+------------------------------------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------------------+------------------------------------------+------+-----+---------+-------+
| login_privilege_id | bigint(20) | NO | PRI | | |
| login_id | varchar(100) | NO | PRI | | |
| privilege_name | enum('ADMIN','ADVISOR','STAFF','OTHERS') | NO | | | |
+--------------------+------------------------------------------+------+-----+---------+-------+
3 rows in set (0.02 sec)

| major_allowed_courses |
+------------------------+-----------------------------------------------------------+------+-----+---------+--
--------------+
| Field | Type | Null | Key | Default | Extra |
+------------------------+-----------------------------------------------------------+------+-----+---------+--
--------------+
| id | bigint(20) unsigned | NO | PRI | NULL |
auto_increment |
| major_form_template_id | bigint(20) unsigned | NO | MUL | |
|
| course_id | varchar(6) | NO | | | |
| course_type |
enum('COMMON_CORE','REQUIRED','ELECTIVE','PREP_FOR_MAJOR') | YES | |
NULL | |
+------------------------+-----------------------------------------------------------+------+-----+---------+--
--------------+
4 rows in set (0.02 sec)

| major_form_templates |
+--------------+---------------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+--------------+---------------------+------+-----+---------+----------------+
| id | bigint(20) unsigned | NO | PRI | NULL | auto_increment |
| sem_yr_id | varchar(4) | NO | | | |
```

```
| template     | text                 | NO   |     |         |                |
| dept_id      | varchar(10)          | NO   |     |         |                |
| program_name | varchar(40)          | NO   |     |         |                |
| comment      | varchar(100)         | YES  |     | NULL    |                |
+--------------+---------------------+------+-----+---------+----------------+
6 rows in set (0.03 sec)

| progress_toward_degree           |
+------------------------------------+---------------------+------+-----+---------+----------------+
| Field                              | Type                | Null | Key | Default | Extra          |
+------------------------------------+---------------------+------+-----+---------+----------------+
| id                                 | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| student_id                         | varchar(11)         | NO   |     |         |                |
| dept_id                            | varchar(20)         | NO   |     |         |                |
| major                              | varchar(20)         | NO   |     |         |                |
| num_degree_units                   | int(11)             | NO   |     |         |                |
| num_prep_for_major_units           | int(11)             | NO   |     |         |                |
| num_major_units                    | int(11)             | NO   |     |         |                |
| num_common_core_units              | int(11)             | NO   |     |         |                |
| num_ge_units                       | int(11)             | NO   |     |         |                |
| num_degree_units_completed         | int(11)             | NO   |     |         |                |
| num_prep_major_units_completed     | int(11)             | NO   |     |         |                |
| num_minor_units_completed          | int(11)             | NO   |     |         |                |
| num_common_units_completed         | int(11)             | NO   |     |         |                |
| num_ge_units_completed             | int(11)             | NO   |     |         |                |
| num_tech_elect_units_completed     | int(11)             | NO   |     |         |                |
| num_years_for_degree               | int(11)             | NO   |     |         |                |
| matriculation_sem_id               | bigint(20) unsigned | NO   |     |         |                |
| matriculation_year                 | char(4)             | NO   |     |         |                |
| target_degree_completion_yr        | char(4)             | NO   |     |         |                |
| target_degree_completion_sem_id    | bigint(20) unsigned | NO   |     |         |                |
| target_prep_major_completion_year  | char(4)             | NO   |     |         |                |
| target_prep_major_completion_sem_id | bigint(20) unsigned | NO   |     |         |                |
| target_major_completion_year       | char(4)             | NO   |     |         |                |
| target_major_completion_sem_id     | bigint(20) unsigned | NO   |     |         |                |
| target_common_core_completion_year | char(4)             | NO   |     |         |                |
| target_common_core_completion_sem_id | bigint(20) unsigned | NO   |     |         |                |
+------------------------------------+---------------------+------+-----+---------+----------------+
26 rows in set (0.02 sec)

| semester_year              |
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| id       | varchar(4)  | NO   | PRI |         |       |
```

semester	varchar(10)	NO	PRI		
year	varchar(4)	NO			

3 rows in set (0.01 sec)

| student_advising_report |

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PRI	NULL	auto_increment
student_id	varchar(11)	NO			
num_prev_sem	int(11)	NO			
num_future_sem	int(11)	NO			
overall_gpa_threshold	float	NO			
major_gpa_threshold	float	NO			
overall_gpa_trend	float	NO			
overall_gpa_intercept_threshold	float	NO			
major_gpa_intercept_threshold	float	NO			
ge_gpa_trend	float	NO			
ge_intercept_threshold	float	NO			
major_course_trend	float	NO			
major_course_intercept_threshold	float	NO			
prep_for_major_trend	float	NO			
prep_for_major_intercept_threshold	float	NO			
common_courses_trend	float	NO			
elective_gpa_trend	float	NO			
elective_gpa_intercept_threshold	float	NO			

18 rows in set (0.02 sec)

| student_candidacy_form |

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PRI	NULL	auto_increment
candidacy_form_xml	text	NO			
candidacy_form_pdf	longblob	NO			
sem_yr_id	varchar(4)	NO			
student_id	varchar(11)	NO	MUL		
degree_prog_name	varchar(70)	NO			
comment	varchar(100)	YES		NULL	
dept_id	varchar(10)	NO	MUL		
proposed_grad_date	date	NO			

9 rows in set (0.01 sec)

| student_course_grade |

Field	Type	Null	Key	Default	Extra
id	bigint(20) unsigned	NO	PRI	NULL	auto_increment
ua	float	YES		NULL	
ug	float	YES		NULL	
ue	float	YES		NULL	
gr	varchar(3)	YES		NULL	
gp	float	NO			
course_id	varchar(6)	YES		NULL	
class_class_nbr	int(19)	YES		NULL	
student_id	varchar(11)	NO	MUL		
sem_yr_id	varchar(4)	YES		NULL	

10 rows in set (0.03 sec)

| student_info |

Field	Type	Null	Key	Default	Extra
student_id	varchar(11)	NO	PRI		
first_name	varchar(30)	YES		NULL	
middle_name	varchar(30)	YES		NULL	
last_name	varchar(30)	YES		NULL	
abv_degree_prog_name1	varchar(10)	YES		NULL	
degree_prog_name1	varchar(30)	YES		NULL	
abv_degree_prog_name2	varchar(10)	YES		NULL	
degree_prog_name2	varchar(30)	YES		NULL	
degree	varchar(4)	YES		NULL	
standing	varchar(10)	YES		NULL	
stem_standing	varchar(10)	YES		NULL	
status	varchar(4)	YES		NULL	
grad_date	date	YES		NULL	
dept_id	varchar(10)	NO	MUL		
address_line_1	varchar(50)	YES		NULL	
address_line_2	varchar(50)	YES		NULL	
phone	varchar(24)	YES		NULL	
state	varchar(6)	YES		NULL	
city	varchar(20)	YES		NULL	
zipcode	varchar(12)	YES		NULL	
email	varchar(70)	YES		NULL	
overall_gpa	float	YES		0	
univ_major_gpa	float	YES		0	

```
| computed_major_gpa     | float       | YES  |     | 0       |       |
| begin_sem_yr_id        | varchar(4)  | YES  |     | NULL    |       |
| dept_degree_program_id | bigint(20)  | YES  |     | NULL    |       |
| advisor_id             | bigint(20)  | YES  |     | NULL    |       |
+------------------------+-------------+------+-----+---------+-------+
27 rows in set (0.01 sec)

| student_major_form              |
+--------------------+---------------------+------+-----+---------+----------------+
| Field              | Type                | Null | Key | Default | Extra          |
+--------------------+---------------------+------+-----+---------+----------------+
| id                 | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| major_form_xml     | text                | NO   |     |         |                |
| major_form_pdf     | longblob            | NO   |     |         |                |
| sem_yr_id          | varchar(4)          | NO   |     |         |                |
| student_id         | varchar(11)         | NO   | MUL |         |                |
| degree_prog_name   | varchar(70)         | NO   |     |         |                |
| comment            | varchar(100)        | YES  |     | NULL    |                |
| dept_id            | varchar(10)         | NO   |     |         |                |
| proposed_grad_date | date                | NO   |     |         |                |
+--------------------+---------------------+------+-----+---------+----------------+
9 rows in set (0.03 sec)

| student_major_form_courses       |
+-------------------------+---------------------+------+-----+---------+----------------+
| Field                   | Type                | Null | Key | Default | Extra          |
+-------------------------+---------------------+------+-----+---------+----------------+
| id                      | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| student_major_form_id   | bigint(20) unsigned | NO   | MUL |         |                |
| major_allowed_courses_id | varchar(6)         | NO   |     |         |                |
+-------------------------+---------------------+------+-----+---------+----------------+
3 rows in set (0.02 sec)

| student_semester_grade          |
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| student_id   | varchar(11) | NO   | MUL |         |       |
| sem_yr_id    | varchar(4)  | YES  |     | NULL    |       |
| ua           | float       | YES  |     | NULL    |       |
| ug           | float       | NO   |     |         |       |
| ue           | float       | NO   |     |         |       |
| gp           | float       | NO   |     |         |       |
| semester_gpa | float       | NO   |     |         |       |
+--------------+-------------+------+-----+---------+-------+
```

7 rows in set (0.03 sec)

```
| student_tech_elective_exceptions |
+------------+---------------------+------+-----+---------+----------------+
| Field      | Type                | Null | Key | Default | Extra          |
+------------+---------------------+------+-----+---------+----------------+
| id         | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| student_id | varchar(11)         | NO   | MUL |         |                |
| course_id  | varchar(6)          | NO   | MUL |         |                |
+------------+---------------------+------+-----+---------+----------------+
3 rows in set (0.02 sec)

| user_login_info                  |
+------------+-----------------------------------------+------+-----+---------+-------+
| Field      | Type                                    | Null | Key | Default | Extra |
+------------+-----------------------------------------+------+-----+---------+-------+
| login_id   | varchar(100)                            | NO   | PRI |         |       |
| login_name | varchar(80)                             | NO   |     |         |       |
| first_name | varchar(80)                             | NO   |     |         |       |
| last_name  | varchar(80)                             | NO   |     |         |       |
| login_type | enum('ADMIN','ADVISOR','STAFF','OTHERS') | YES  |     | NULL    |       |
+------------+-----------------------------------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

APPENDIX B

Java Programs

B.1. Connect.java: to connect with the database.

```
import java.sql.*;
import java.util.Properties;
import java.util.Vector;
public class Connect {
  public static Connection makeConnection(String url) throws Exception {
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    String userName = "shifali";
    String pword = "database";
    Properties props = new Properties();
    props.put("user" , userName);
    props.put("password", pword);
    props.put("userJvmCharsetConverters", "true");
    Connection conn = DriverManager.getConnection (url, props);

    System.out.println("Connection to database " + url + " succeeded");

    return conn;
  }
  public static void main(String[] args) {
    System.out.println("Hello, World");
    try
    {
      //                    Connection                    connEforms                    =
Connect.makeConnection("jdbc:mysql://localhost/eforms_test");
      Connection     connEforms     =     Connect.makeConnection("jdbc:mysql://testing-
ldap.engr.sjsu.edu:3306/eforms_test");

      Statement stmt = connEforms.createStatement();
      stmt.setQueryTimeout(180);
      try
      {
        Vector<String> abvVec = new Vector<String>();
        Vector<String> nameVec = new Vector<String>();
        ResultSet rs = stmt.executeQuery( "select * from departments");
        while (rs.next())  {
          String id = rs.getString("id");
          String abv = rs.getString("abv");
          String name = rs.getString("name");
```

```
          abvVec.add(abv);
          nameVec.add(name);
        }
        System.out.println("String[] abvList = {");
        for (String abv : abvVec) {
          System.out.print("\"" + abv + "\", ");
        }
        System.out.println("}");
        System.out.println("String[] nameList = {");
        for (String name : nameVec) {
          System.out.print("\"" + name + "\", ");
        }
        System.out.println("}");
      }

      catch(SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage() );
        throw new Exception();
      }
    }
    catch( Exception e)
    {
      System.out.println("There has been an error " + e);
    }
  }
}
```

B.2. Departments.java: reads departments data from the database. We need departmentId to be referred in other tables.

```
import java.sql.*;
import java.util.Properties;
import java.util.Vector;
public class Departments {
  public Departments() {
    department_ = null;
  }
  public void readData(String url) {
    department_ = new Vector<Department>();
    try {
      Connection connEforms = Connect.makeConnection(url);
      Statement stmt = connEforms.createStatement();
      stmt.setQueryTimeout(180);
      try {
```

```
      ResultSet rs = stmt.executeQuery( "select * from departments");
      while (rs.next())  {
        String id = rs.getString("id");
        String abv = rs.getString("abv");
        String name = rs.getString("name");
        Department dept = new Department(id, name, abv);
        department_.add(dept);
      }
    } catch(SQLException ex) {
      System.out.println("SQLException: " + ex.getMessage() );
      throw new Exception();
    }
  } catch( Exception e) {
    System.out.println("There has been an error " + e);
  }
 }
 public void printMe() {
   System.out.println("Total Departments : " + department_.size());
   for (Department dept : department_) {
     dept.printMe();
   }
 }
 public Vector<Department> departments() { return department_; }

 public static void main(String[] args) {
   Departments depts = new Departments();
   depts.readData("jdbc:mysql://testing-ldap.engr.sjsu.edu:3306/eforms_test");
   depts.printMe();
 }
 private Vector<Department> department_;
}
class Department {
 public Department(String id, String name, String abv) {
   id_ = id;
   name_ = name;
   abv_ = abv;
 }
 public void printMe() {
   System.out.println("Id: " + id_ + " Name: " + name_ + " Abv: " + abv_);
 }
 String id_;
 String name_;
 String abv_;
}
```

B.3. AcademicAdvisors.java: populates dummy data in academic_advisors table to be used later in triggers.

```
import java.util.Date;
import java.sql.*;
import java.util.Properties;
import java.util.Vector;
import java.util.Map;
import java.util.HashMap;

class AcademicAdvisors
{
  public void populateDummyData(String url) {
    Departments depts = new Departments();
    depts.readData(url);

    try {
      Connection connEforms = Connect.makeConnection(url);
      Statement stmt = connEforms.createStatement();
      stmt.setQueryTimeout(180);

      stmt.executeUpdate("DELETE FROM academic_advisors");
      try {
        Vector<Department> deptVec = depts.departments();

        int i = 0;
        for (Department dept : deptVec) {
          i++;
          String deptId = dept.id_;
          String fName = "foo_" + i;
          String lName = "bar_" + i;
          String email = "foo_bar_" + i + "@sjsu.edu";

          String insertString = "insert academic_advisors values" +
                        "(" + i +
                        "," + "'" + deptId + "'" +
                        "," + "'" + fName + "'" +
                        "," + "'" + lName + "'" +
                        "," + "'" + email + "'" +
                        ", NULL, NULL" +
                        ")";
          System.out.println(insertString);
          stmt.executeUpdate(insertString);
        }
      } catch(SQLException ex) {
```

```
        System.out.println("SQLException: " + ex.getMessage() );
        throw new Exception();
      }
    } catch( Exception e) {
      System.out.println("There has been an error " + e);
    }
  }

  public static void main (String args[] ) {
    System.out.println("USE eforms_test;");
    AcademicAdvisors acAv = new AcademicAdvisors();
    acAv.populateDummyData("jdbc:mysql://testing-ldap.engr.sjsu.edu:3306/eforms_test");
  }
}
```

B.4. StudentCourseGrades.java: reads student course grade data from the database. This was the sanitized data which came from actual student records. Hence we needed to read this to get whatever student ids used here to be consistent with references from other tables.

```
import java.sql.*;
import java.util.Properties;
import java.util.Vector;
import java.util.Map;
import java.util.HashMap;

public class StudentCourseGrades {
  public StudentCourseGrades() {
    studentId_ = null;
    studentCourseGrade_ = null;
    studentSemId_ = null;
    studentCourseGradePerSem_ = null;
  }
  public void readData(String url) {
    studentId_ = new Vector<String>();
    studentCourseGrade_ = new HashMap<String, Vector<StudentCourseGrade>>();
    studentSemId_ = new Vector<StudentSemIdKey>();
    studentCourseGradePerSem_ = new HashMap<StudentSemIdKey,
Vector<StudentCourseGrade>>();
    try {
      Connection connEforms = Connect.makeConnection(url);
      Statement stmt = connEforms.createStatement();
      stmt.setQueryTimeout(180);
      try {
        ResultSet rs = stmt.executeQuery("select * from student_course_grade order by
student_id, sem_yr_id");
```

```
      while (rs.next())  {
        int id = rs.getInt("id");
        String studentId = new String(rs.getString("student_id"));
        String courseId = new String(rs.getString("course_id"));
        String semYrId = new String(rs.getString("sem_yr_id"));
        int classNum = rs.getInt("class_class_nbr");
        String gr = new String(rs.getString("gr"));
        float gp = rs.getFloat("gp");
        float ua = rs.getFloat("ua");
        float ug = rs.getFloat("ug");
        float ue = rs.getFloat("ue");
        if (ug == 0)
          continue;
        StudentCourseGrade sCG = new StudentCourseGrade(id, studentId, courseId,
semYrId,
                      classNum, gr, ua, ug, ue, gp);
        if (studentCourseGrade_.containsKey(studentId)) {
          Vector<StudentCourseGrade> sCGVec = studentCourseGrade_.get(studentId);
          sCGVec.add(sCG);
        } else {
          Vector<StudentCourseGrade> sCGVec = new Vector<StudentCourseGrade>();
          sCGVec.add(sCG);
          studentCourseGrade_.put(studentId, sCGVec);
          studentId_.add(studentId);
        }
        StudentSemIdKey semIdKey = new StudentSemIdKey(studentId, semYrId);
        if (studentCourseGradePerSem_.containsKey(semIdKey)) {
          Vector<StudentCourseGrade>                    sCGVec                    =
studentCourseGradePerSem_.get(semIdKey);
          sCGVec.add(sCG);
        } else {
          Vector<StudentCourseGrade> sCGVec = new Vector<StudentCourseGrade>();
          sCGVec.add(sCG);
          studentCourseGradePerSem_.put(semIdKey, sCGVec);
          studentSemId_.add(semIdKey);
        }
      }
    } catch(SQLException ex) {
      System.out.println("SQLException: " + ex.getMessage() );
      throw new Exception();
    }
  } catch( Exception e) {
    System.out.println("There has been an error " + e);
  }
}
```

```
public void updateStudentCourseGP(String url) {
  try {
    Connection connEforms = Connect.makeConnection(url);
    Statement stmt = connEforms.createStatement();
    stmt.setQueryTimeout(180);
    try {
      for (StudentSemIdKey studentSem : studentSemId_) {
        Vector<StudentCourseGrade>                    sCGVec                    =
studentCourseGradePerSem_.get(studentSem);
        for (StudentCourseGrade sCG : sCGVec) {
          if (java.lang.Math.abs(sCG.gpa_ * sCG.ue_ - sCG.gp_) > 0.0001) {
            String upQr = "UPDATE student_course_grade SET gp = '" + sCG.gpa_ * sCG.ue_
+ "'" +
                          " WHERE id = '" + sCG.id_ + "'";
            stmt.executeUpdate(upQr);
          }
        }
      }
    } catch(SQLException ex) {
      System.out.println("SQLException: " + ex.getMessage() );
      throw new Exception();
    }
  } catch( Exception e) {
    System.out.println("There has been an error " + e);
  }
}
public void printMe() {
  System.out.println("Total student_ids : " + studentId_.size());
  for (StudentSemIdKey studentSem : studentSemId_) {
    studentSem.printMe();
    Vector<StudentCourseGrade> sCGVec = studentCourseGradePerSem_.get(studentSem);
    for (StudentCourseGrade sCG : sCGVec) {
      sCG.printMe();
    }
  }
}
public Vector<String> studentIds() { return studentId_; }
public Vector<StudentSemIdKey> studentSemIds() { return studentSemId_; }
public Vector<StudentCourseGrade> studentSemesterCourseGrades(StudentSemIdKey sss)
{
  return studentCourseGradePerSem_.get(sss);
}
public static void main(String[] args) {
  StudentCourseGrades sCGs = new StudentCourseGrades();
  String url = "jdbc:mysql://testing-ldap.engr.sjsu.edu:3306/eforms_test";
```

```
        sCGs.readData(url);
        sCGs.updateStudentCourseGP(url);
        sCGs.printMe();
      }
      private Vector<String> studentId_;
      private Map<String, Vector<StudentCourseGrade>> studentCourseGrade_;
      private Vector<StudentSemIdKey> studentSemId_;
      private           Map<StudentSemIdKey,              Vector<StudentCourseGrade>>
studentCourseGradePerSem_;
    }

    class StudentSemIdKey {
      String studentId_;
      String semYrId_;
      public StudentSemIdKey(String studentId, String semYrId) {
        studentId_ = studentId;
        semYrId_ = semYrId;
      }
      public void printMe() {
        System.out.println("StudentId: " + studentId_ + " SemYrId: " + semYrId_);
      }
      @Override public boolean equals(Object that) {
        // System.out.println("equals called");
        if (this == that)
          return true;
        if ( !(that instanceof StudentSemIdKey) )
          return false;
        StudentSemIdKey k = (StudentSemIdKey)that;
        // System.out.print("First object: ");
        // this.printMe();
        // System.out.print("Second object: ");
        // k.printMe();
        boolean ret = (studentId_.equals(k.studentId_) && semYrId_.equals(k.semYrId_));
        // System.out.println("equals returned: " + ret);
        return ret;
        //return (studentId_.equals(k.studentId_) && semYrId_.equals(k.semYrId_));
      }
      @Override public int hashCode() {
        return studentId_.hashCode() + semYrId_.hashCode();
      }
    };
    class StudentCourseGrade {
      public StudentCourseGrade(int id, String studentId, String courseId, String semYrId,
                          int classNum, String gr, float ua, float ug, float ue, float gp) {
        id_ = id;
```

```
    studentId_ = studentId;
    courseId_ = courseId;
    semYrId_ = semYrId;
    classNum_ = classNum;
    gr_ = gr;
    if (gr.equals("A+"))
      gpa_ = 4.0f;
    else if (gr.equals("A"))
      gpa_ = 4.0f;
    else if (gr.equals("A-"))
      gpa_ = 3.7f;
    else if (gr.equals("B+"))
      gpa_ = 3.3f;
    else if (gr.equals("B"))
      gpa_ = 3.0f;
    else if (gr.equals("B-"))
      gpa_ = 2.7f;
    else if (gr.equals("C+"))
      gpa_ = 2.3f;
    else if (gr.equals("C"))
      gpa_ = 2.0f;
    else if (gr.equals("C-"))
      gpa_ = 1.7f;
    else if (gr.equals("D+"))
      gpa_ = 1.3f;
    else if (gr.equals("D"))
      gpa_ = 1.0f;
    else if (gr.equals("D-"))
      gpa_ = 0.7f;
    else
      gpa_ = 0.0f;
    ua_ = ua;
    ug_ = ug;
    ue_ = ue;
    gp_ = gp;
    if (java.lang.Math.abs(gpa_ * ue_ - gp_) > 0.0001) {
      System.out.println("Error: gpa(" + gpa_ + ") * ue(" + ue_ + ") != gp(" + gp_ + ")");
      printMe();
    }
  }
  public void printMe() {
    System.out.println("   StudentId: " + studentId_ + " CourseId: " + courseId_ +
                       " SemYrId: " + semYrId_ + " GR: " + gr_ + " GP: " + gp_ +
                       " UE: " + ue_ + " UG: " + ug_ + " UA: " + ua_);
  }
```

```
  int id_;
  String studentId_;
  String courseId_;
  String semYrId_;
  int classNum_;
  float gpa_; // gpa of this course
  float gp_; // grade point of this course = gpa_ * ue_
  float ua_; // units accumulated with this course
  float ug_; // units graded
  float ue_; // units earned with this course
  String gr_;
}
```

B.5. StudentSemesterGrades.java: Computes and populates student semester grades data from student course grade data.

```
import java.sql.*;
import java.util.Properties;
import java.util.Vector;
import java.util.Map;
import java.util.HashMap;

public class StudentSemesterGrades {
  public StudentSemesterGrades(StudentCourseGrades sCGs) {
    studentSemesterGrade_ = null;
    studentCourseGrades_ = sCGs;
  }
  public void computeData() {
    studentSemesterGrade_ = new HashMap<String, Vector<StudentSemesterGrade>>();
    for (StudentSemIdKey studentSem : studentCourseGrades_.studentSemIds()) {
      String studentId = studentSem.studentId_;
      String semYrId = studentSem.semYrId_;
      Vector<StudentCourseGrade> sCGVec =
          studentCourseGrades_.studentSemesterCourseGrades(studentSem);
      float ua = 0.0f; // units accumulated in this sem
      float ug = 0.0f; // units graded
      float ue = 0.0f; // units earned
      float semGpa = 0.0f; // gpa of this sem
      for (StudentCourseGrade sCG : sCGVec) {
        ua += sCG.ua_;
        ug += sCG.ug_;
        ue += sCG.ue_;
        semGpa += sCG.ue_ * sCG.gpa_;
      }
      semGpa = semGpa / ue;
```

```
    StudentSemesterGrade sSG = new StudentSemesterGrade(studentId, semYrId, ua, ug, ue,
                                                         semGpa * ue, semGpa);
    if (studentSemesterGrade_.containsKey(studentId)) {
      Vector<StudentSemesterGrade> sSGVec = studentSemesterGrade_.get(studentId);
      sSGVec.add(sSG);
    } else {
      Vector<StudentSemesterGrade> sSGVec = new Vector<StudentSemesterGrade>();
      sSGVec.add(sSG);
      studentSemesterGrade_.put(studentId, sSGVec);
    }
  }
  computeCGPA();
}
public void computeCGPA() {
  Vector<String> studentIds = studentCourseGrades_.studentIds();
  for (String studentId : studentIds) {
    Vector<StudentSemesterGrade> sSGVec = studentSemesterGrade_.get(studentId);
    float cgpa = 0.0f;
    float totUnits = 0.0f;
    for (StudentSemesterGrade sSG : sSGVec) {
      cgpa = (cgpa * totUnits + sSG.semGpa_ * sSG.ue_) / (totUnits + sSG.ue_);
      sSG.setCGPA(cgpa);
      totUnits += sSG.ue_;
    }
  }
}
public void populateData(String url) {
  System.out.println("Starting populateData");
  computeData();
  System.out.println("After computeData");
  try {
    Connection connEforms = Connect.makeConnection(url);
    Statement stmt = connEforms.createStatement();
    stmt.setQueryTimeout(180);
    stmt.executeUpdate("DELETE FROM student_semester_grade");
    try {
      Vector<String> studentIds = studentCourseGrades_.studentIds();
      for (String studentId : studentIds) {
        Vector<StudentSemesterGrade> sSGVec = studentSemesterGrade_.get(studentId);
        if (sSGVec == null) {
          System.out.println("Student: " + studentId + " has no sem data");
          continue;
        }
        for (StudentSemesterGrade sSG : sSGVec) {
          String insertString = "insert into student_semester_grade values(" +
```

```
                        "'" + sSG.studentId_ + "', '" + sSG.semYrId_ + "', " +
                        "'" + sSG.ua_ + "', '" + sSG.ug_ + "', '" + sSG.ue_ + "', " +
                        "'" + sSG.semGp_ + "', '" + sSG.semGpa_ + "')";
            System.out.println(insertString);
            stmt.executeUpdate(insertString);
          }
        }
        } catch(SQLException ex) {
          System.out.println("SQLException: " + ex.getMessage() );
          throw new Exception();
        }
      } catch( Exception e) {
        System.out.println("There has been an error " + e);
      }
    }
    public void printMe() {
      Vector<String> studentIds = studentCourseGrades_.studentIds();
      for (String studentId : studentIds) {
        Vector<StudentSemesterGrade> sSGVec = studentSemesterGrade_.get(studentId);
        if (sSGVec == null) {
          System.out.println("PrintMe Student: " + studentId + " has no sem data");
          continue;
        }
        for (StudentSemesterGrade sSG : sSGVec) {
          sSG.printMe();
        }
      }
    }
    public static void main(String[] args) {
      StudentCourseGrades sCGs = new StudentCourseGrades();
      String url = "jdbc:mysql://testing-ldap.engr.sjsu.edu:3306/eforms_test";
      sCGs.readData(url);
      // sCGs.printMe();
      StudentSemesterGrades sSGs = new StudentSemesterGrades(sCGs);
      sSGs.populateData(url);
      // sSGs.printMe();
    }
    private StudentCourseGrades studentCourseGrades_;
    private Map<String, Vector<StudentSemesterGrade>> studentSemesterGrade_;
  }
  class StudentSemesterGrade {
    public StudentSemesterGrade(String studentId, String semYrId, float ua, float ug, float ue,
                          float semGp, float semGpa) {
      studentId_ = new String(studentId);
      semYrId_ = new String(semYrId);
```

```
    ua_ = ua;
    ug_ = ug;
    ue_ = ue;
    semGp_ = semGp;
    semGpa_ = semGpa;
    cgpaComputed_ = false;
  }
  public void setCGPA(float cgpa) {
    cgpaComputed_ = true;
    cgpa_ = cgpa;
  }
  public float getCGPA() { return cgpa_; }
  public void printMe() {
    System.out.println("   StudentId: " + studentId_ + " SemYrId: " + semYrId_ +
                       " UE: " + ue_ + " UG: " + ug_ + " UA: " + ua_ +
                       " GP: " + semGp_ + " SemGPA: " + semGpa_);
  }
  String studentId_;
  String semYrId_;
  float ua_; // units accumulated in this sem
  float ug_; // units graded
  float ue_; // units earned
  private float cgpa_; // cgpa
  float semGpa_; // gpa of this sem
  float semGp_; // gp of this sem = gpa_ * ue_
  private boolean cgpaComputed_;
}
```

B.6. GenRAndom.java: to generate a random number or string between an input ranges.

```
import java.util.*;
public class GenRandom {
  private static Random rn = new Random();
  private GenRandom()
  {
  }
  public static int rand(int lo, int hi)
  {
    int n = hi - lo + 1;
    int i = rn.nextInt() % n;
    if (i < 0)
      i = -i;
    return lo + i;
  }
  public static String randomstring(int lo, int hi)
```

```
  {
    int n = rand(lo, hi);
    byte b[] = new byte[n];
    for (int i = 0; i < n; i++)
      b[i] = (byte)rand('a', 'z');
    return new String(b, 0);
  }
  public static String randomstring()
  {
    return randomstring(5, 25);
  }
}
```

B.7. StudentInfo.java: to populate dummy data for student_info table. It uses in-memory structures in other java programs to get proper ids.

```
import java.util.Date;
import java.sql.*;
import java.util.Properties;
import java.util.Vector;
import java.util.Map;
import java.util.HashMap;

class StudentInfo
{
  public void populateDummyData(String url) {
    StudentCourseGrades sCGs = new StudentCourseGrades();
    sCGs.readData(url);
    Departments depts = new Departments();
    depts.readData(url);
    try {
      Connection connEforms = Connect.makeConnection(url);
      Statement stmt = connEforms.createStatement();
      stmt.setQueryTimeout(180);
      stmt.executeUpdate("DELETE FROM student_info");
      try {
        Vector<String> studentIdVec = sCGs.studentIds();
        Vector<Department> deptVec = depts.departments();
        String[] degreeList = {"GRAD", "PBAC", "UGRD" };
        int[] semYrIdList = {2082, 2083, 2084, 2091, 2092, 2093, 2094, 2101, 2102, 2103};

        int i = 0;
        for (String studentId : studentIdVec) {
          i++;
          String first_name = "bob_" + i;
```

```
        String middle_name = "";
        String last_name = "patel_" + i;
        int abvProgIndex = GenRandom.rand(0, deptVec.size() - 1);
        String abv_degree_prog_name1 = deptVec.get(abvProgIndex).abv_;
        String degree_prog_name1 = deptVec.get(abvProgIndex).name_;
        String abv_degree_prog_name2 = "";
        String degree_prog_name2 = "";
        String degree = degreeList[GenRandom.rand(0, degreeList.length - 1)];
        String prev_college_info = "";
        String change_classification_sem_yr_id = "";
        String standing = "";
        String stem_standing = "";
        String status = "";
        int yr = 2010 + GenRandom.rand(0, 2);
        String dt = "" + yr + "-0" + GenRandom.rand(1, 9) + "-" + GenRandom.rand(10, 28);
        String grad_date = dt;
        String  dept_id = "" + deptVec.get(abvProgIndex).id_;
        String address_line_1 = "";
        String address_line_2 = "";
        String phone = "";
        String state = "";
        String city = "";
        String zipcode = "";
        String email = first_name + last_name + "@sjsu.edu";
        float overall_gpa = 4 - 2.0f * GenRandom.rand(0, 99) / 100.0f;
        float univ_major_gpa = 4 - 2.0f * GenRandom.rand(0, 99) / 100.0f;
        float computed_major_gpa = 4 - 2.0f * GenRandom.rand(0, 99) / 100.0f;
        int beginYear = 2007 + GenRandom.rand(0, 2);
        String beginDate = "'" + yr + "-0" + GenRandom.rand(1, 9) + "-" +
GenRandom.rand(10, 28) + "'";
        String begin_sem_yr_id = "" + semYrIdList[GenRandom.rand(0, semYrIdList.length -
1)];
        String dept_degree_program_id = "NULL";
        String advisor_id = "" + GenRandom.rand(1, 19);
        String insertString = "insert into student_info values" +
                "(" + "'" + studentId + "'" +
                "," + "'" + first_name + "'" +
                "," + "'" + middle_name + "'" +
                "," + "'" + last_name + "'" +
                "," + "'" + abv_degree_prog_name1 +  "'" +
                "," + "'" + degree_prog_name1 + "'" +
                "," + "'" + abv_degree_prog_name2 + "'" +
                "," + "'" + degree_prog_name2 + "'" +
                "," + "'" + degree + "'" +
                "," + "'" + prev_college_info + "'" +
```

```
                    "," + "'" + change_classification_sem_yr_id + "'" +
                    "," + "'" + standing + "'" +
                    "," + "'" + stem_standing + "'" +
                    "," + "'" + status + "'" +
                    "," + "'" + grad_date + "'" +
                    "," + "'" + dept_id + "'" +
                    "," + "'" + address_line_1 + "'" +
                    "," + "'" + address_line_2 + "'" +
                    "," + "'" + phone + "'" +
                    "," + "'" + state + "'" +
                    "," + "'" + city + "'" +
                    "," + "'" + zipcode + "'" +
                    "," + "'" + email + "'" +
                    "," + overall_gpa + "," + univ_major_gpa + "," + computed_major_gpa +
                    "," + "'" + begin_sem_yr_id + "'" +
                    "," + dept_degree_program_id + "," + advisor_id +
                    ")";
          System.out.println(insertString);
          stmt.executeUpdate(insertString);
        }
      } catch(SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage() );
        throw new Exception();
      }
    } catch( Exception e) {
      System.out.println("There has been an error " + e);
    }
  }
  public static void main (String args[] ) {
    System.out.println("USE eforms_test;");
    StudentInfo sI = new StudentInfo();
    sI.populateDummyData("jdbc:mysql://testing-ldap.engr.sjsu.edu:3306/eforms_test");
  }
}
```

B.8. StudentAdvisingReport.java: to populate dummy data for student_advising_report table.

```
import java.util.Date;
import java.sql.*;
import java.util.Properties;
import java.util.Vector;
import java.util.Map;
import java.util.HashMap;

class StudentAdvisingReport
```

```
{
  public void populateDummyData(String url) {
    StudentCourseGrades sCGs = new StudentCourseGrades();
    sCGs.readData(url);
    try {
      Connection connEforms = Connect.makeConnection(url);
      Statement stmt = connEforms.createStatement();
      stmt.setQueryTimeout(180);
      stmt.executeUpdate("DELETE FROM student_advising_report");
      try {
        Vector<String> studentIdVec = sCGs.studentIds();
        int i = 0;
        for (String studentId : studentIdVec) {
          i++;
          int numPrevSem = GenRandom.rand(1, 4);
          int numFutSem = 4 - GenRandom.rand(1, 4);
          float overallGPATh = 2.0f;
          float majorGPATh = 2.0f;
          float overallGPATrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f; // -0.5 to 0.5
          float overallGPAIntTh = 2.0f;
          float majorGPAIntTh = 2.0f;
          float geGPATrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
          float geIntTh = 2.0f;
          float majorCourseTrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
          float majorCourseIntTh = 2.0f;
          float prepForMajorTrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
          float prepForMajorIntTh = 2.0f;
          float commonCoursesTrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
          float electiveGPATrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
          float electiveGPAIntTh = 2.0f;
          String insertString = "insert into student_advising_report values" +
                "(" + i +
                "," + "'" + studentId + "'" +
                "," + numPrevSem +
                "," + numFutSem +
                "," + overallGPATh +
                "," + majorGPATh +
                "," + overallGPATrend +
                "," + overallGPAIntTh +
                "," + majorGPAIntTh +
                "," + geGPATrend +
                "," + geIntTh +
                "," + majorCourseTrend +
                "," + majorCourseIntTh +
                "," + prepForMajorTrend +
```

```
                    "," + prepForMajorIntTh +
                    "," + commonCoursesTrend +
                    "," + electiveGPATrend +
                    "," + electiveGPAIntTh +
                    ")";
          System.out.println(insertString);
          stmt.executeUpdate(insertString);
        }
      } catch(SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage() );
        throw new Exception();
      }
    } catch( Exception e) {
      System.out.println("There has been an error " + e);
    }
  }
  public static void main (String args[] ) {
    System.out.println("USE eforms_test;");
    StudentAdvisingReport sAR = new StudentAdvisingReport();
    sAR.populateDummyData("jdbc:mysql://testing-ldap.engr.sjsu.edu:3306/eforms_test");
  }
}
```

B.9. DeptAdvisingReport.java: to populate dummy data for dept_advising_report.

```
import java.util.Date;
import java.sql.*;
import java.util.Properties;
import java.util.Vector;
import java.util.Map;
import java.util.HashMap;

class DeptAdvisingReport
{
  public void populateDummyData(String url) {
    Departments depts = new Departments();
    depts.readData(url);
    try {
      Connection connEforms = Connect.makeConnection(url);
      Statement stmt = connEforms.createStatement();
      stmt.setQueryTimeout(180);
      stmt.executeUpdate("DELETE FROM dept_advising_report");
      try {
        Vector<Department> deptVec = depts.departments();
        int i = 0;
```

```
          for (Department dept : deptVec) {
            i++;
            String deptId = dept.id_;
            String majorId = deptId; //major_id same as dept_id
            float avgOverallGPATrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f; // -0.5 to
0.5
            float avgOverallGPAIntTh = 2.0f;
            String reportPeriodStartSemYrId = "";
            String reportPeriodEndSemYrId = "";
            float avgMajorGPATrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
            float avgMajorGPAIntTh = 2.0f;
            float avgGeTrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
            float avgMajorCourseTrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
            float avgPrepForMajorTrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
            float avgCommonCoursesTrend = 1.0f * GenRandom.rand(1, 99) / 100.0f - 0.5f;
            String insertString = "insert into dept_advising_report values" +
                        "(" + i +
                        "," + "'" + deptId + "'" +
                        "," + "'" + majorId + "'" +
                        "," + avgOverallGPATrend +
                        "," + avgOverallGPAIntTh +
                        "," + "'" + reportPeriodStartSemYrId + "'" +
                        "," + "'" + reportPeriodEndSemYrId + "'" +
                        "," + avgMajorGPATrend +
                        "," + avgMajorGPAIntTh +
                        "," + avgGeTrend +
                        "," + avgMajorCourseTrend +
                        "," + avgPrepForMajorTrend +
                        "," + avgCommonCoursesTrend +
                        ")";
            System.out.println(insertString);
            stmt.executeUpdate(insertString);
          }
        } catch(SQLException ex) {
          System.out.println("SQLException: " + ex.getMessage() );
          throw new Exception();
        }
      } catch( Exception e) {
        System.out.println("There has been an error " + e);
      }
    }
    public static void main (String args[] ) {
      System.out.println("USE eforms_test;");
      DeptAdvisingReport dAR = new DeptAdvisingReport();
      dAR.populateDummyData("jdbc:mysql://testing-ldap.engr.sjsu.edu:3306/eforms_test");
```

```
  }
}
```

B.10. Queries.java -- Runs each query 100 times and computes the average execution time of a query

```
import java.sql.*;
import java.util.Properties;
import java.util.Vector;
public class Queries {
  public Queries(String url) {
    url_ = url;
    StudentCourseGrades sCGs = new StudentCourseGrades();
    sCGs.readData(url);
    studentIdVec_ = sCGs.studentIds();
    depts_ = new Departments();
    depts_.readData(url);
  }
  public void runQuery1(int howManyTimes) {
    System.out.println("query1: Give me overall GPA trend for specific student id");
    long totalTime = 0;
    long totalStartTime = System.currentTimeMillis();
    for(int i = 0; i < howManyTimes; ++i) {
      long startTime = System.currentTimeMillis();
      Connection connEforms = null;
      try {
        connEforms = Connect.makeConnection(url_);
        Statement stmt = connEforms.createStatement();
        stmt.setQueryTimeout(180);
        try {
          String studentId = studentIdVec_.get(GenRandom.rand(0, studentIdVec_.size() - 1));
          String queryString = "select student_id, overall_gpa_trend from " +
                               "student_advising_report where " +
                               "student_id = '" + studentId + "'";
          ResultSet rs = stmt.executeQuery(queryString);
          System.out.println("------------- Query " + (i+1) + " ------------------");
          System.out.println("student_id    overall_gpa_trend");
          while (rs.next())  {
            String sid = rs.getString("student_id");
            float overallGPATrend = rs.getFloat("overall_gpa_trend");
            System.out.println(sid + "    " + overallGPATrend);
          }
        } catch(SQLException ex) {
          System.out.println("SQLException: " + ex.getMessage() );
          throw new Exception();
```

```
        }
      } catch( Exception e) {
        System.out.println("There has been an error " + e);
      }
      finally {
        if (connEforms != null) {
          try {
            connEforms.close ();
            System.out.println ("Database connection terminated");
          } catch (Exception e) { /* ignore close errors */ }
        }
      }
      long endTime = System.currentTimeMillis();
      System.out.println("Time consumed by this query(1): "+ (endTime-startTime) + "ms");
      totalTime += endTime-startTime;
    }
    long totalEndTime = System.currentTimeMillis();
    System.out.println("Total time consumed by " + howManyTimes + " queries(1): " +
                    (totalEndTime - totalStartTime) + "ms");
    System.out.println("Avg time consumed by one query(1): " +
                   (totalEndTime - totalStartTime)/howManyTimes + "ms");
    // System.out.println("Total time: " + totalTime + "ms");
  }
  public void runQuery2(int howManyTimes) {
    System.out.println("query2: Give me list of all students whose GPA trend is -ve");
    long totalTime = 0;
    long totalStartTime = System.currentTimeMillis();
    for(int i = 0; i < howManyTimes; ++i) {
      long startTime = System.currentTimeMillis();

      Connection connEforms = null;
      try {
        connEforms = Connect.makeConnection(url_);
        Statement stmt = connEforms.createStatement();
        stmt.setQueryTimeout(180);
        try {
          String studentId = studentIdVec_.get(GenRandom.rand(0, studentIdVec_.size() - 1));
          String queryString = "select student_id, overall_gpa_trend from " +
                           "student_advising_report where " +
                           "overall_gpa_trend < 0";
          ResultSet rs = stmt.executeQuery(queryString);
          System.out.println("------------- Query " + (i+1) + " -------------------");
          System.out.println("student_id    overall_gpa_trend");
          while (rs.next())  {
            String sid = rs.getString("student_id");
```

```
          float overallGPATrend = rs.getFloat("overall_gpa_trend");
          System.out.println(sid + "    " + overallGPATrend);
        }
      } catch(SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage() );
        throw new Exception();
      }
    } catch( Exception e) {
      System.out.println("There has been an error " + e);
    }
    finally {
      if (connEforms != null) {
        try {
          connEforms.close ();
          System.out.println ("Database connection terminated");
        } catch (Exception e) { /* ignore close errors */ }
      }
    }
    long endTime = System.currentTimeMillis();
    System.out.println("Time consumed by this query(2): "+ (endTime-startTime) + "ms");
    totalTime += endTime-startTime;
  }
  long totalEndTime = System.currentTimeMillis();
  System.out.println("Total time consumed by " + howManyTimes + " queries(2): " +
                    (totalEndTime - totalStartTime) + "ms");
  System.out.println("Avg time consumed by one query(2): " +
                    (totalEndTime - totalStartTime)/howManyTimes + "ms");
  // System.out.println("Total time: " + totalTime + "ms");
}
public void runQuery3(int howManyTimes) {
  System.out.println("query3: Give me list of all students whose GPA trend is -ve & " +
                    "will result in probation within (1, 2 ) semesters");
  long totalTime = 0;
  long totalStartTime = System.currentTimeMillis();
  for(int i = 0; i < howManyTimes; ++i) {
    long startTime = System.currentTimeMillis();
    Connection connEforms = null;
    try {
      connEforms = Connect.makeConnection(url_);
      Statement stmt = connEforms.createStatement();
      stmt.setQueryTimeout(180);
      try {
        String studentId = studentIdVec_.get(GenRandom.rand(0, studentIdVec_.size() - 1));
        String queryString = "Select student_info.student_id, overall_gpa, " +
                            "overall_gpa_trend, overall_gpa_threshold, " +
```

```
                    "overall_gpa_intercept_threshold " +
                    "from student_info, student_advising_report " +
                    "where student_info.student_id = " +
                      "student_advising_report.student_id AND " +
                    "overall_gpa_trend < 0 AND " +
                    "overall_gpa + overall_gpa_trend * " +
                      "overall_gpa_intercept_threshold <= overall_gpa_threshold";
        ResultSet rs = stmt.executeQuery(queryString);
        System.out.println("------------- Query " + (i+1) + " ------------------");
        System.out.println("student_id    overall_gpa_trend");
        while (rs.next())  {
          String sid = rs.getString("student_id");
          float overallGPA = rs.getFloat("overall_gpa");
          float overallGPATrend = rs.getFloat("overall_gpa_trend");
          float overallGPATh = rs.getFloat("overall_gpa_threshold");
          float overallGPAIntTh = rs.getFloat("overall_gpa_intercept_threshold");
          System.out.println(sid + "    " + overallGPA + "    " + overallGPATrend +
                    "    " + overallGPATh + "    " + overallGPAIntTh);
        }
      } catch(SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage() );
        throw new Exception();
      }
    } catch( Exception e) {
      System.out.println("There has been an error " + e);
    }
    finally {
      if (connEforms != null) {
        try {
          connEforms.close ();
          System.out.println ("Database connection terminated");
        } catch (Exception e) { /* ignore close errors */ }
      }
    }
    long endTime = System.currentTimeMillis();
    System.out.println("Time consumed by this query(3): "+ (endTime-startTime) + "ms");
    totalTime += endTime-startTime;
  }
  long totalEndTime = System.currentTimeMillis();
  System.out.println("Total time consumed by " + howManyTimes + " queries(3): " +
              (totalEndTime - totalStartTime) + "ms");
  System.out.println("Avg time consumed by one query(3): " +
              (totalEndTime - totalStartTime)/howManyTimes + "ms");
  // System.out.println("Total time: " + totalTime + "ms");
}
```

```
    public void runQuery4(int howManyTimes) {
      System.out.println("query4: Give me list of students whose (common-courses or elective or major or GE or prep-for-major) GPA trend is -ve");
      long totalTime = 0;
      long totalStartTime = System.currentTimeMillis();
      for(int i = 0; i < howManyTimes; ++i) {
        long startTime = System.currentTimeMillis();
        Connection connEforms = null;
        try {
          connEforms = Connect.makeConnection(url_);
          Statement stmt = connEforms.createStatement();
          stmt.setQueryTimeout(180);
          try {
            String studentId = studentIdVec_.get(GenRandom.rand(0, studentIdVec_.size() - 1));
            String queryString = "select student_id, overall_gpa_trend, ge_gpa_trend, " +
                          "major_course_trend " +
                          "from student_advising_report " +
                          "where overall_gpa_trend < 0 OR ge_gpa_trend < 0 or " +
                          "major_course_trend < 0";
            ResultSet rs = stmt.executeQuery(queryString);
            System.out.println("------------- Query " + (i+1) + " ------------------");
            System.out.println("student_id          overall_gpa_trend          ge_gpa_trend major_course_trend");
            while (rs.next())  {
              String sid = rs.getString("student_id");
              float overallGPATrend = rs.getFloat("overall_gpa_trend");
              float geGPATrend = rs.getFloat("ge_gpa_trend");
              float majorGPATrend = rs.getFloat("major_course_trend");
              System.out.println(sid + "     " + overallGPATrend + "     " + geGPATrend + "     " + majorGPATrend);
            }
          } catch(SQLException ex) {
            System.out.println("SQLException: " + ex.getMessage() );
            throw new Exception();
          }
        } catch( Exception e) {
          System.out.println("There has been an error " + e);
        }
        finally {
          if (connEforms != null) {
            try {
              connEforms.close ();
              System.out.println ("Database connection terminated");
            } catch (Exception e) { /* ignore close errors */ }
          }
```

```
      }
      long endTime = System.currentTimeMillis();
      System.out.println("Time consumed by this query(4): "+ (endTime-startTime) + "ms");
      totalTime += endTime-startTime;
    }
    long totalEndTime = System.currentTimeMillis();
    System.out.println("Total time consumed by " + howManyTimes + " queries(4): " +
                (totalEndTime - totalStartTime) + "ms");
    System.out.println("Avg time consumed by one query(4): " +
                (totalEndTime - totalStartTime)/howManyTimes + "ms");
    // System.out.println("Total time: " + totalTime + "ms");
  }
  public void runQuery5(int howManyTimes) {
    System.out.println("query5: Give me the GPA trend of students of a particular
department");
    long totalTime = 0;
    long totalStartTime = System.currentTimeMillis();
    for(int i = 0; i < howManyTimes; ++i) {
      long startTime = System.currentTimeMillis();
      Connection connEforms = null;
      try {
        connEforms = Connect.makeConnection(url_);
        Statement stmt = connEforms.createStatement();
        stmt.setQueryTimeout(180);
        try {
          Vector<Department> deptVec = depts_.departments();
          String deptId = deptVec.get(GenRandom.rand(0, deptVec.size()-1)).id_;
          String queryString = "select student_info.student_id, dept_id, overall_gpa_trend " +
                      "from student_info, student_advising_report " +
                      "where student_info.student_id = " +
                         "student_advising_report.student_id AND " +
                         "student_info.dept_id = '" + deptId + "'";
          ResultSet rs = stmt.executeQuery(queryString);
          System.out.println("------------- Query " + (i+1) + " -------------------");
          System.out.println("dept_id    student_id    overall_gpa_trend");
          while (rs.next())  {
            String did = rs.getString("dept_id");
            String sid = rs.getString("student_info.student_id");
            float overallGPATrend = rs.getFloat("overall_gpa_trend");
            System.out.println(did + "    " + sid + "    " + overallGPATrend);
          }
        } catch(SQLException ex) {
          System.out.println("SQLException: " + ex.getMessage() );
          throw new Exception();
        }
```

```
      } catch( Exception e) {
        System.out.println("There has been an error " + e);
      }
      finally {
        if (connEforms != null) {
          try {
            connEforms.close ();
            System.out.println ("Database connection terminated");
          } catch (Exception e) { /* ignore close errors */ }
        }
      }
      long endTime = System.currentTimeMillis();
      System.out.println("Time consumed by this query(5): "+ (endTime-startTime) + "ms");
      totalTime += endTime-startTime;
    }
    long totalEndTime = System.currentTimeMillis();
    System.out.println("Total time consumed by " + howManyTimes + " queries(5): " +
                    (totalEndTime - totalStartTime) + "ms");
    System.out.println("Avg time consumed by one query(5): " +
                   (totalEndTime - totalStartTime)/howManyTimes + "ms");
    // System.out.println("Total time: " + totalTime + "ms");
  }
  public void runQuery6(int howManyTimes) {
    System.out.println("query6: Give me the average GPA trend of a particular department");
    long totalTime = 0;
    long totalStartTime = System.currentTimeMillis();
    for(int i = 0; i < howManyTimes; ++i) {
      long startTime = System.currentTimeMillis();
      Connection connEforms = null;
      try {
        connEforms = Connect.makeConnection(url_);
        Statement stmt = connEforms.createStatement();
        stmt.setQueryTimeout(180);
        try {
          Vector<Department> deptVec = depts_.departments();
          String deptId = deptVec.get(GenRandom.rand(0, deptVec.size()-1)).id_;
          String queryString = "select name, dept_id, avg_overall_gpa_trend " +
                        "from departments, dept_advising_report " +
                        "where departments.id = dept_id AND dept_id = '" + deptId + "'";
          ResultSet rs = stmt.executeQuery(queryString);
          System.out.println("------------- Query " + (i+1) + " -------------------");
          System.out.println("dept name    dept_id    avg_overall_gpa_trend");
          while (rs.next())  {
            String deptName = rs.getString("name");
            String did = rs.getString("dept_id");
```

```
          float avgOverallGPATrend = rs.getFloat("avg_overall_gpa_trend");
          System.out.println(deptName + "    " + did + "    " + avgOverallGPATrend);
        }
      } catch(SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage() );
        throw new Exception();
      }
    } catch( Exception e) {
      System.out.println("There has been an error " + e);
    }
    finally {
      if (connEforms != null) {
        try {
          connEforms.close ();
          System.out.println ("Database connection terminated");
        } catch (Exception e) { /* ignore close errors */ }
      }
    }
    long endTime = System.currentTimeMillis();
    System.out.println("Time consumed by this query(6): "+ (endTime-startTime) + "ms");
    totalTime += endTime-startTime;
  }
  long totalEndTime = System.currentTimeMillis();
  System.out.println("Total time consumed by " + howManyTimes + " queries(6): " +
                    (totalEndTime - totalStartTime) + "ms");
  System.out.println("Avg time consumed by one query(6): " +
                   (totalEndTime - totalStartTime)/howManyTimes + "ms");
  // System.out.println("Total time: " + totalTime + "ms");
}
public void runQuery7(int howManyTimes) {
  System.out.println("query7: Give me list of all departments whose average (major or GE
or overall) GPA trend is -ve");
  long totalTime = 0;
  long totalStartTime = System.currentTimeMillis();
  for(int i = 0; i < howManyTimes; ++i) {
    long startTime = System.currentTimeMillis();
    Connection connEforms = null;
    try {
      connEforms = Connect.makeConnection(url_);
      Statement stmt = connEforms.createStatement();
      stmt.setQueryTimeout(180);
      try {
        Vector<Department> deptVec = depts_.departments();
        String deptId = deptVec.get(GenRandom.rand(0, deptVec.size()-1)).id_;
        String queryString = "select name, dept_id, avg_overall_gpa_trend, " +
```

```
                    "avg_major_gpa_trend, avg_ge_trend " +
                    "from departments, dept_advising_report " +
                    "where departments.id = dept_id AND " +
                    "(avg_overall_gpa_trend < 0 OR avg_major_gpa_trend < 0 OR " +
                     "avg_ge_trend < 0)";
        ResultSet rs = stmt.executeQuery(queryString);
        System.out.println("------------- Query " + (i+1) + " ------------------");
        System.out.println("dept     name                    dept_id                  avg_overall_gpa_trend
avg_major_gpa_trend    avg_ge_trend");
        while (rs.next())  {
          String deptName = rs.getString("name");
          String did = rs.getString("dept_id");
          float avgOverallGPATrend = rs.getFloat("avg_overall_gpa_trend");
          float avgMajorGPATrend = rs.getFloat("avg_major_gpa_trend");
          float avgGeGPATrend = rs.getFloat("avg_ge_trend");
          System.out.println(deptName + "    " + did + "    " + avgOverallGPATrend +
                    "    " + avgMajorGPATrend + "    " + avgGeGPATrend);
        }
      } catch(SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage() );
        throw new Exception();
      }
    } catch( Exception e) {
      System.out.println("There has been an error " + e);
    }
    finally {
      if (connEforms != null) {
        try {
          connEforms.close ();
          System.out.println ("Database connection terminated");
        } catch (Exception e) { /* ignore close errors */ }
      }
    }
    long endTime = System.currentTimeMillis();
    System.out.println("Time consumed by this query(7): "+ (endTime-startTime) + "ms");
    totalTime += endTime-startTime;
  }
  long totalEndTime = System.currentTimeMillis();
  System.out.println("Total time consumed by " + howManyTimes + " queries(7): " +
            (totalEndTime - totalStartTime) + "ms");
  System.out.println("Avg time consumed by one query(7): " +
            (totalEndTime - totalStartTime)/howManyTimes + "ms");
  // System.out.println("Total time: " + totalTime + "ms");
}
public static void main(String[] args) {
```

```
    String url = "jdbc:mysql://testing-ldap.engr.sjsu.edu:3306/eforms_test";
    Queries queries = new Queries(url);
    queries.runQuery1(100);
    queries.runQuery2(100);
    queries.runQuery3(100);
    queries.runQuery4(100);
    queries.runQuery5(100);
    queries.runQuery6(100);
    queries.runQuery7(100);
  }
  private Vector<String> studentIdVec_;
  private Departments depts_;
  private String url_;
}
```

APPENDIX C

Triggers

General Testing flow: We used following flow for the testing:

Step1: Show initial data
Step2: Update/insert/delete to initiate our trigger
Step3: Show results to check if trigger updated the data
Step4: Update/delete back to original data
Step5: Show results to check if last data same as initial data

DETAILS:

C.1 Trigger: Trig_update_grade

This trigger is triggered when student grade(gr) is updated for any course

```
use eforms_test;
drop trigger trig_update_grade;
delimiter //
create trigger trig_update_grade
before update on student_course_grade
for each row
begin
  DECLARE oldCourseGpa FLOAT;
  DECLARE newCourseGpa FLOAT;
  DECLARE unit_acc FLOAT;
  DECLARE sem_gpa FLOAT;
  DECLARE sem_gp FLOAT;
  DECLARE x FLOAT;

  if ((old.gr != new.gr) && (old.ue = new.ue))
  then
    set oldCourseGpa = old.gp / old.ue;

    if (STRCMP(new.gr, 'A+') = 0) then
      set newCourseGpa = 4.0;
    elseif (STRCMP(new.gr, 'A') = 0) then
      set newCourseGpa = 4.0;
    elseif (STRCMP(new.gr, 'A-') = 0) then
      set newCourseGpa = 3.7;
    elseif (STRCMP(new.gr, 'B+') = 0) then
      set newCourseGpa = 3.3;
    elseif (STRCMP(new.gr, 'B') = 0) then
      set newCourseGpa = 3.0;
    elseif (STRCMP(new.gr, 'B-') = 0) then
```

```
      set newCourseGpa = 2.7;
    elseif (STRCMP(new.gr, 'C+') = 0) then
      set newCourseGpa = 2.3;
    elseif (STRCMP(new.gr, 'C') = 0) then
      set newCourseGpa = 2.0;
    elseif (STRCMP(new.gr, 'C-') = 0) then
      set newCourseGpa = 1.7;
    elseif (STRCMP(new.gr, 'D+') = 0) then
      set newCourseGpa = 1.3;
    elseif (STRCMP(new.gr, 'D') = 0) then
      set newCourseGpa = 1.0;
    elseif (STRCMP(new.gr, 'D-') = 0) then
      set newCourseGpa = 0.7;
    else
      set newCourseGpa = 0;
    end if;

    select ue, semester_gpa into unit_acc, sem_gpa
    from student_semester_grade
    where student_id = new.student_id and
          sem_yr_id = new.sem_yr_id limit 1;

    set x = (unit_acc * sem_gpa) - (old.ue * oldCourseGpa);
    set sem_gp = (old.ue * newCourseGpa + x) ;
    set sem_gpa = sem_gp / unit_acc;

    set new.gp = newCourseGpa * old.ue;

    update student_semester_grade
    set semester_gpa = sem_gpa, gp = sem_gp
    where student_id = new.student_id and
          sem_yr_id = new.sem_yr_id;
  end if;
end;//
delimiter ;
```

TESTING OF THIS TRIGGER: We ran following queries after creating this trigger:

Step1: Show initial data

*select * from student_course_grade where student_id = 005484609;*

Course grade is B+

```
| 1771 |    3 |    3 |    3 | B+   |  9.9 | 001803    |          43718 | 005484609  | 2084      |
```

*select * from student_semester_grade where student_id = 005484609;*

Semester gpa is 3.65
| 005484609 | 2084 | 6 | 6 | 6 | 21.9 | 3.65 |

Step2: Update course grade from B+ to A. Our trigger should get executed automatically to update course gp and semester gpa, gp
update student_course_grade set gr = 'A' where id = 1771;

Step3: Show data after the update to test the trigger
*select * from student_course_grade where student_id = 005484609;*

| 1771 | 3 | 3 | 3 | A | 12 | 001803 | 43718 | 005484609 | 2084 |

If you see above Grade has been changed from B+ to A, gp has been changed to 12 from 9.9.

*select * from student_semester_grade where student_id = 005484609;*

| 005484609 | 2084 | 6 | 6 | 6 | 24 | 4 |

If you see above semester gpa has been changed to 4, gp has been changed to 24.

Step4: Update back grade from A to B+
update student_course_grade set gr = 'B+' where id = 1771;

Step5: Show data after update back
*select * from student_course_grade where student_id = 005484609;*
| 1771 | 3 | 3 | 3 | B+ | 9.9 | 001803 | 43718 | 005484609 | 2084 |

*select * from student_semester_grade where student_id = 005484609;*
| 005484609 | 2084 | 6 | 6 | 6 | 21.9 | 3.65 |

As you can see now the data is same as our initial data.

This tests the correctness of this trigger.
Below is the output captured from mysql prompt:

```
mysql> source /home/shifali/project/eforms_test/triggerGradeUpdate.sql
Database changed
Query OK, 0 rows affected (0.15 sec)
Query OK, 0 rows affected (0.01 sec)

+-------+------+------+------+------+------+-----------+-----------------+------------+-----------+
| id    | ua   | ug   | ue   | gr   | gp   | course_id | class_class_nbr | student_id | sem_yr_id |
+-------+------+------+------+------+------+-----------+-----------------+------------+-----------+
|     6 |    3 |    3 |    3 | B+   |  9.9 | 010685    |           24531 | 005484609  | 2082      |
```

```
|  1771 |    3 |    3 |    3 | B+   |  9.9 | 001803    |           43718 | 005484609  | 2084      |
|  4647 |    3 |    3 |    3 | A-   | 11.1 | 005262    |           25882 | 005484609  | 2082      |
|  5789 |    0 |    0 |    0 |      |    0 | 005262    |           25883 | 005484609  | 2082      |
|  7526 |    0 |    0 |    0 |      |    0 | 010688    |           24522 | 005484609  | 2082      |
| 10969 |    3 |    3 |    3 | A    |   12 | 010684    |           43724 | 005484609  | 2084      |
| 13009 |    3 |    3 |    3 | A    |   12 | 010688    |           24521 | 005484609  | 2082      |
+-------+------+------+------+------+------+-----------+-----------------+------------+-----------+
7 rows in set (0.06 sec)

+------------+-----------+------+----+----+------+--------------+
| student_id | sem_yr_id | ua   | ug | ue | gp   | semester_gpa |
+------------+-----------+------+----+----+------+--------------+
| 005484609  | 2082      |    9 |  9 |  9 |   33 |      3.66667 |
| 005484609  | 2084      |    6 |  6 |  6 | 21.9 |         3.65 |
+------------+-----------+------+----+----+------+--------------+
2 rows in set (0.02 sec)

Query OK, 1 row affected (0.07 sec)
Rows matched: 1  Changed: 1  Warnings: 0

+-------+------+------+------+------+------+-----------+-----------------+------------+-----------+
| id    | ua   | ug   | ue   | gr   | gp   | course_id | class_class_nbr | student_id | sem_yr_id |
+-------+------+------+------+------+------+-----------+-----------------+------------+-----------+
|     6 |    3 |    3 |    3 | B+   |  9.9 | 010685    |           24531 | 005484609  | 2082      |
|  1771 |    3 |    3 |    3 | A    |   12 | 001803    |           43718 | 005484609  | 2084      |
|  4647 |    3 |    3 |    3 | A-   | 11.1 | 005262    |           25882 | 005484609  | 2082      |
|  5789 |    0 |    0 |    0 |      |    0 | 005262    |           25883 | 005484609  | 2082      |
|  7526 |    0 |    0 |    0 |      |    0 | 010688    |           24522 | 005484609  | 2082      |
| 10969 |    3 |    3 |    3 | A    |   12 | 010684    |           43724 | 005484609  | 2084      |
| 13009 |    3 |    3 |    3 | A    |   12 | 010688    |           24521 | 005484609  | 2082      |
+-------+------+------+------+------+------+-----------+-----------------+------------+-----------+
7 rows in set (0.07 sec)

+------------+-----------+------+----+----+----+--------------+
| student_id | sem_yr_id | ua   | ug | ue | gp | semester_gpa |
+------------+-----------+------+----+----+----+--------------+
| 005484609  | 2082      |    9 |  9 |  9 | 33 |      3.66667 |
| 005484609  | 2084      |    6 |  6 |  6 | 24 |            4 |
+------------+-----------+------+----+----+----+--------------+
2 rows in set (0.02 sec)

Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

+-------+------+------+------+------+------+-----------+-----------------+------------+-----------+
```

id	ua	ug	ue	gr	gp	course_id	class_class_nbr	student_id	sem_yr_id
6	3	3	3	B+	9.9	010685	24531	005484609	2082
1771	3	3	3	B+	9.9	001803	43718	005484609	2084
4647	3	3	3	A-	11.1	005262	25882	005484609	2082
5789	0	0	0		0	005262	25883	005484609	2082
7526	0	0	0		0	010688	24522	005484609	2082
10969	3	3	3	A	12	010684	43724	005484609	2084
13009	3	3	3	A	12	010688	24521	005484609	2082

7 rows in set (0.07 sec)

student_id	sem_yr_id	ua	ug	ue	gp	semester_gpa
005484609	2082	9	9	9	33	3.66667
005484609	2084	6	6	6	21.9	3.65

2 rows in set (0.02 sec)

C.2 Trigger: Trig_insert_grade
This trigger is triggered when student course gr is inserted for any course

```
use eforms_test;
drop trigger trig_insert_grade;
delimiter //
create trigger trig_insert_grade
before insert on student_course_grade
for each row
begin
  DECLARE sem_ua FLOAT;
  DECLARE sem_ug FLOAT;
  DECLARE sem_ue FLOAT;
  DECLARE sem_gp FLOAT;
  DECLARE sem_gpa FLOAT;
  DECLARE notFound INT DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR NOT FOUND set notFound = 1;

  select ua, ug, ue, gp, semester_gpa into sem_ua, sem_ug, sem_ue, sem_gp, sem_gpa
  from student_semester_grade
  where student_id = new.student_id and
        sem_yr_id = new.sem_yr_id limit 1;

  if (notFound = 1) then
    insert into student_semester_grade values
```

```
    (new.student_id, new.sem_yr_id, new.ua, new.ug, new.ue, new.gp, new.gp/new.ue);
  else
    set sem_ua = sem_ua + new.ua;
    set sem_ug = sem_ug + new.ug;
    set sem_ue = sem_ue + new.ue;
    set sem_gp = sem_gp + new.gp;
    set sem_gpa = sem_gp / sem_ue;

    update student_semester_grade
    set ua = sem_ua, ug = sem_ug, ue = sem_ue, gp = sem_gp, semester_gpa = sem_gpa
    where student_id = new.student_id and
            sem_yr_id = new.sem_yr_id;
  end if;
end;//
delimiter ;
```

TESTING OF THIS TRIGGER: We ran following queries after creating this trigger:

Step1: Show initial data

```
select * from student_course_grade where student_id = '009876543';
Empty set (0.03 sec)

select * from student_semester_grade where student_id = '009876543';
Empty set (0.00 sec)
```

Step2: Insert one course grade record

```
insert into student_course_grade values (15000, 3, 3, 3, 'B', 9, '004922', 49859, '009876543',
2084);

select * from student_course_grade where student_id = '009876543';
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| 15000 |    3 |    3 |    3 | B    |  9 | 004922    |           49859 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+

select * from student_semester_grade where student_id = '009876543';
+------------+-----------+------+----+----+----+--------------+
| 009876543  | 2084      |    3 |  3 |  3 |  9 |            3 |
+------------+-----------+------+----+----+----+--------------+
```

Step3: Insert one more course grade record

```
insert into student_course_grade values (15001, 3, 3, 3, 'A', 12, '004892', 49821, '009876543',
2084);

select * from student_course_grade where student_id = '009876543';
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
```

```
| 15000 |    3 |    3 |    3 | B    |  9 | 004922    |          49859 | 009876543  | 2084      |
| 15001 |    3 |    3 |    3 | A    | 12 | 004892    |          49821 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
```

select * from student_semester_grade where student_id = '009876543';

```
+------------+-----------+------+----+----+----+--------------+
| student_id | sem_yr_id | ua   | ug | ue | gp | semester_gpa |
+------------+-----------+------+----+----+----+--------------+
| 009876543  | 2084      |    6 |  6 |  6 | 21 |          3.5 |
+------------+-----------+------+----+----+----+--------------+
```

Step4: Delete course grade records
delete from student_course_grade where student_id = '009876543';
delete from student_semester_grade where student_id = '009876543';

Step5: Show data after update back
select * from student_course_grade where student_id = '009876543';
Empty set (0.00 sec)

select * from student_semester_grade where student_id = '009876543';
Empty set (0.00 sec)

As you can see now the data is same as our initial data.

This tests the correctness of Trigger: Trig_insert_grade
Below is the output captured from mysql prompt:

```
mysql> source /home/shifali/project/eforms_test/triggerGradeInsert.sql
Database changed
Query OK, 0 rows affected (0.01 sec)
Query OK, 0 rows affected (0.00 sec)
Empty set (0.03 sec)
Empty set (0.00 sec)
Query OK, 1 row affected (0.02 sec)

+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| id    | ua   | ug   | ue   | gr   | gp | course_id | class_class_nbr | student_id | sem_yr_id |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| 15000 |    3 |    3 |    3 | B    |  9 | 004922    |          49859 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
1 row in set (0.00 sec)

+------------+-----------+------+----+----+----+--------------+
| student_id | sem_yr_id | ua   | ug | ue | gp | semester_gpa |
+------------+-----------+------+----+----+----+--------------+
```

```
| 009876543  | 2084      |    3 |  3 |  3 |  9 |            3 |
+------------+-----------+------+----+----+----+--------------+
1 row in set (0.00 sec)

Query OK, 1 row affected (0.02 sec)

+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| id    | ua   | ug   | ue   | gr   | gp | course_id | class_class_nbr | student_id | sem_yr_id |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| 15000 |    3 |    3 |    3 | B    |  9 | 004922    |           49859 | 009876543  | 2084      |
| 15001 |    3 |    3 |    3 | A    | 12 | 004892    |           49821 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
2 rows in set (0.00 sec)

+------------+-----------+------+----+----+----+--------------+
| student_id | sem_yr_id | ua   | ug | ue | gp | semester_gpa |
+------------+-----------+------+----+----+----+--------------+
| 009876543  | 2084      |    6 |  6 |  6 | 21 |          3.5 |
+------------+-----------+------+----+----+----+--------------+
1 row in set (0.00 sec)

Query OK, 2 rows affected (0.01 sec)
Query OK, 0 rows affected (0.00 sec)
Empty set (0.00 sec)
Empty set (0.00 sec)
```

C.3 Trigger: Trig_delete_grade
This trigger is triggered when student course gr is deleted for any course

```
use eforms_test;
drop trigger trig_delete_grade;
delimiter //
create trigger trig_delete_grade
before delete on student_course_grade
for each row
begin
  DECLARE sem_ua FLOAT;
  DECLARE sem_ug FLOAT;
  DECLARE sem_ue FLOAT;
  DECLARE sem_gp FLOAT;
  DECLARE sem_gpa FLOAT;
  DECLARE notFound INT DEFAULT 0;
  DECLARE EXIT HANDLER FOR NOT FOUND set notFound = 1;

  select ua, ug, ue, gp, semester_gpa into sem_ua, sem_ug, sem_ue, sem_gp, sem_gpa
```

```
  from student_semester_grade
  where student_id = old.student_id and
        sem_yr_id = old.sem_yr_id limit 1;

  if (notFound = 1) then
    set notFound = 1;
  else
    set sem_ua = sem_ua - old.ua;
    set sem_ug = sem_ug - old.ug;
    set sem_ue = sem_ue - old.ue;
    set sem_gp = sem_gp - old.gp;
    set sem_gpa = sem_gp / sem_ue;

    /* If its last course which is delted then delete semester record */
    if ((sem_gp = 0)) then
      delete from student_semester_grade
      where student_id = old.student_id and
            sem_yr_id = old.sem_yr_id;
    else
      update student_semester_grade
      set ua = sem_ua, ug = sem_ug, ue = sem_ue, gp = sem_gp, semester_gpa = sem_gpa
      where student_id = old.student_id and
            sem_yr_id = old.sem_yr_id;
    end if;
  end if;
end;//
delimiter ;
```

TESTING OF THIS TRIGGER: We ran following queries after creating this trigger:

Step1: Show initial data

select * from student_course_grade where student_id = '009876543';
Empty set (0.02 sec)

select * from student_semester_grade where student_id = '009876543';
Empty set (0.00 sec)

Step2: Insert one course grade record

insert into student_course_grade values (15000, 3, 3, 3, 'B', 9, '004922', 49859, '009876543', 2084);

select * from student_course_grade where student_id = '009876543';

```
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| 15000 |    3 |    3 |    3 | B    |  9 | 004922    |           49859 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
```

```
select * from student_semester_grade where student_id = '009876543';
+------------+-----------+------+----+----+----+--------------+
| 009876543  | 2084      |    3 |  3 |  3 |  9 |            3 |
+------------+-----------+------+----+----+----+--------------+
```

Step3: Insert one more course grade record

```
insert into student_course_grade values (15001, 3, 3, 3, 'A', 12, '004892', 49821, '009876543', 2084);

select * from student_course_grade where student_id = '009876543';
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| id    | ua   | ug   | ue   | gr   | gp | course_id | class_class_nbr | student_id | sem_yr_id |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| 15000 |    3 |    3 |    3 | B    |  9 | 004922    |           49859 | 009876543  | 2084      |
| 15001 |    3 |    3 |    3 | A    | 12 | 004892    |           49821 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
select * from student_semester_grade where student_id = '009876543';
+------------+-----------+------+----+----+----+--------------+
| 009876543  | 2084      |    6 |  6 |  6 | 21 |          3.5 |
+------------+-----------+------+----+----+----+--------------+
```

Step4: Delete first course grade record

```
delete from student_course_grade where id = '15000' ;

select * from student_course_grade where student_id = '009876543';
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| 15001 |    3 |    3 |    3 | A    | 12 | 004892    |           49821 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
select * from student_semester_grade where student_id = '009876543';
+------------+-----------+------+----+----+----+--------------+
| 009876543  | 2084      |    3 |  3 |  3 | 12 |            4 |
+------------+-----------+------+----+----+----+--------------+
```

From above you can see that semester grade record is automatically updated.

Step5: Delete one more course grade record

```
delete from student_course_grade where id = '15001' ;
```

Step6: Show data after deletion

```
select * from student_course_grade where student_id = '009876543';
Empty set (0.00 sec)

select * from student_semester_grade where student_id = '009876543';
Empty set (0.00 sec)
```

As you can see now the data is same as our initial data.

This tests the correctness of Trigger: Trig_delete_grade
Below is the output captured from mysql prompt:

```
mysql> source /home/shifali/project/eforms_test/triggerGradeDelete.sql
Database changed
Query OK, 0 rows affected (0.02 sec)
Query OK, 0 rows affected (0.00 sec)
Empty set (0.02 sec)
Empty set (0.00 sec)
Query OK, 1 row affected (0.05 sec)

+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| id    | ua   | ug   | ue   | gr   | gp | course_id | class_class_nbr | student_id | sem_yr_id |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| 15000 |    3 |    3 |    3 | B    |  9 | 004922    |           49859 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
1 row in set (0.00 sec)

+------------+-----------+------+----+----+----+--------------+
| student_id | sem_yr_id | ua   | ug | ue | gp | semester_gpa |
+------------+-----------+------+----+----+----+--------------+
| 009876543  | 2084      |    3 |  3 |  3 |  9 |            3 |
+------------+-----------+------+----+----+----+--------------+
1 row in set (0.00 sec)

Query OK, 1 row affected (0.00 sec)

+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| id    | ua   | ug   | ue   | gr   | gp | course_id | class_class_nbr | student_id | sem_yr_id |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| 15000 |    3 |    3 |    3 | B    |  9 | 004922    |           49859 | 009876543  | 2084      |
| 15001 |    3 |    3 |    3 | A    | 12 | 004892    |           49821 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
2 rows in set (0.00 sec)

+------------+-----------+------+----+----+----+--------------+
| student_id | sem_yr_id | ua   | ug | ue | gp | semester_gpa |
+------------+-----------+------+----+----+----+--------------+
| 009876543  | 2084      |    6 |  6 |  6 | 21 |          3.5 |
+------------+-----------+------+----+----+----+--------------+
1 row in set (0.00 sec)

Query OK, 1 row affected (0.11 sec)
```

```
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| id    | ua   | ug   | ue   | gr   | gp | course_id | class_class_nbr | student_id | sem_yr_id |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
| 15001 |    3 |    3 |    3 | A    | 12 | 004892    |           49821 | 009876543  | 2084      |
+-------+------+------+------+------+----+-----------+-----------------+------------+-----------+
1 row in set (0.00 sec)

+------------+-----------+------+----+----+----+--------------+
| student_id | sem_yr_id | ua   | ug | ue | gp | semester_gpa |
+------------+-----------+------+----+----+----+--------------+
| 009876543  | 2084      |    3 |  3 |  3 | 12 |            4 |
+------------+-----------+------+----+----+----+--------------+
1 row in set (0.00 sec)

Query OK, 1 row affected (0.01 sec)
Empty set (0.00 sec)
Empty set (0.00 sec)
```

C.4 Trigger: Trig_update_student_advising
This trigger is triggered when student gpa trend is negative and will result in probation within intercept threshold semesters

```
use eforms_test;
drop trigger trig_update_student_advising;
delimiter //
create trigger trig_update_student_advising
before update on student_advising_report
for each row
begin
  DECLARE done INT DEFAULT 0;
  DECLARE sid CHAR(11);
  DECLARE ovall_gpa FLOAT;
  DECLARE sems_to_go FLOAT;
  DECLARE ovall_gpa_trend FLOAT;
  DECLARE ovall_gpa_threshold FLOAT;
  DECLARE ovall_gpa_int_threshold FLOAT;
  DECLARE mailSent INT DEFAULT 0;
  DECLARE cur1 CURSOR FOR SELECT student_id, overall_gpa FROM student_info;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  set ovall_gpa_trend = new.overall_gpa_trend;
  set ovall_gpa_threshold = new.overall_gpa_threshold;
  set ovall_gpa_int_threshold = new.overall_gpa_intercept_threshold;
  if (ovall_gpa_trend < 0) then
    OPEN cur1;
```

```
    REPEAT
      FETCH cur1 INTO sid, ovall_gpa;
      if NOT done then
        if (sid = new.student_id) then
          set sems_to_go = ovall_gpa + ovall_gpa_trend * ovall_gpa_int_threshold;
          if (sems_to_go <= ovall_gpa_threshold) then
            set mailSent = 1;
            /*  \! rm -f /tmp/emails.txt; */
            select academic_advisors.email, student_info.email
            from academic_advisors, student_info
            where student_info.student_id = '006529601' and
                  student_info.advisor_id = academic_advisors.advisor_id
            into outfile '/tmp/emails_eforms.txt';

            /* \! echo "MsgBody: Student could go on probation" | mail -s "Urgent: student going on
probation" shifugupta@yahoo.com;  */
            \! /home/shifali/project/sendEmail.csh
          end if;
        end if;
      end if;
    UNTIL done END REPEAT;

    CLOSE cur1;

  end if;
end;
//
delimiter ;
```

Script: sendEmail.csh

```
#!/bin/csh -f

foreach em (`cat /tmp/emails.txt`)
 # echo "MsgBody: Student could go on probation" | mail -s "Urgent: student going on
probation" shifugupta@yahoo.com;
 cat /tmp/emails.txt | mail -s "Urgent: student going on probation" $em shifugupta@yahoo.com;
end
```

TESTING OF THIS TRIGGER: We ran following queries after creating this trigger:

Step1: Show initial data

```
select student_info.student_id, overall_gpa, overall_gpa_trend, overall_gpa_threshold,
       overall_gpa_intercept_threshold
from student_info, student_advising_report
```

```
where student_info.student_id = student_advising_report.student_id AND
      student_info.student_id = '006529601' AND overall_gpa_trend < 0 AND
      overall_gpa + overall_gpa_trend * overall_gpa_intercept_threshold
                                <= overall_gpa_threshold;
```

Empty set (0.01 sec)

This student is not going on probation hence this query returned empty set.

Step2: Update student's gpa trend to such a number that student will go on probation within intercept threshold.

```
update student_advising_report
set overall_gpa_trend = -0.3
where student_id = '006529601';
```

```
select student_info.student_id, overall_gpa, overall_gpa_trend, overall_gpa_threshold,
       overall_gpa_intercept_threshold
from student_info, student_advising_report
where student_info.student_id = student_advising_report.student_id AND
      student_info.student_id = '006529601' AND overall_gpa_trend < 0 AND
      overall_gpa + overall_gpa_trend * overall_gpa_intercept_threshold
                                <= overall_gpa_threshold;
+------------+-------------+-------------------+----------------------+---------------------------------+
| 006529601  |        2.44 |              -0.3 |                    2 |                               2 |
+------------+-------------+-------------------+----------------------+---------------------------------+
```

From above you can see that this student is now part of the query. This means that a mail must be sent to the student and his advisor.

Step3: Update student's gpa trend back to original value.

```
update student_advising_report
set overall_gpa_trend = -0.2
where student_id = '006529601';
```

Currently I send the mail to myself also to check the correctness. After step2, I saw following email which tests the correctness of this trigger.

--- On Fri, 11/13/09, shifali@testing-ldap.engr.sjsu.edu <shifali@testing-ldap.engr.sjsu.edu> wrote: > From: shifali@testing-ldap.engr.sjsu.edu <shifali@testing-ldap.engr.sjsu.edu> > Subject: Urgent: student going on probation > To: bob_3238patel_3238@sjsu.edu, shifugupta@yahoo.com > Date: Friday, November 13, 2009, 9:11 PM > foo_bar_14@sjsu.edu > bob_3238patel_3238@sjsu.edu >

Below is the output captured from mysql prompt:

```
mysql> source /home/shifali/project/eforms_test/triggerStudentAdvising.sql
Database changed
Query OK, 0 rows affected (0.12 sec)
Query OK, 0 rows affected (0.01 sec)
Empty set (0.01 sec)
Query OK, 1 row affected (0.33 sec)
Rows matched: 1  Changed: 1  Warnings: 0

+------------+-------------+-------------------+----------------------+--------------------------------+
| student_id | overall_gpa | overall_gpa_trend | overall_gpa_threshold |
overall_gpa_intercept_threshold |
+------------+-------------+-------------------+----------------------+--------------------------------
-+
| 006529601  |        2.44 |              -0.3 |                    2 |                               2 |
+------------+-------------+-------------------+----------------------+--------------------------------+
1 row in set (0.00 sec)

Query OK, 1 row affected (0.12 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```