|  |
| --- |
| Final Report |
| Gradebook Assistant |
| Rose-Hulman Institute of Technology – CSSE 333 |

|  |
| --- |
| Vince Anderson, Kiana Caston, and Linde Schaffer  Spring 2017 |

Contents

[Executive Summary 2](#_Toc482713048)

[Introduction 2](#_Toc482713049)

[Problem Description 2](#_Toc482713050)

[Solution Description 2](#_Toc482713051)

[Front-End Discussion 2](#_Toc482713052)

[Back-End Discussion 2](#_Toc482713053)

[Key Challenges 2](#_Toc482713054)

[Database Design 2](#_Toc482713055)

[Security Measures 2](#_Toc482713056)

[Integrity Constraints 2](#_Toc482713057)

[Stored Procedures 2](#_Toc482713058)

[Views 2](#_Toc482713059)

[Indexes 2](#_Toc482713060)

[Triggers 2](#_Toc482713061)

[Design Analysis 2](#_Toc482713062)

[Strengths 2](#_Toc482713063)

[Weakness 2](#_Toc482713064)

[Appendix A 2](#_Toc482713065)

[Relational Schema 2](#_Toc482713066)

[Entity Relationship Diagram 2](#_Toc482713067)

[Explanation of Entity Relationship Diagram 2](#_Toc482713068)

[Glossary 2](#_Toc482713069)

[References 2](#_Toc482713070)

# Executive Summary

This final report document contains an overview of the many pieces considered and developed during the Introduction to Databases final project. This document will describe the initial problem in which this project is meant to solve as well as how the team went about solving it. The document will include discussion about the front-end, back-end, key challenges, and details about the database design. An analysis regarding the solution’s strengths and weaknesses will be performed. An appendix, glossary, and references list will be included at the end of the document.

# Introduction

This document is the final report on the Gradebook Assistant Database project. The team that was the driving force behind this project consists of Vince Anderson, Kiana Caston, and Linde Schaffer. The purpose of this document is to review the problem the team had to solve, and discuss the solution and how effective their approach was. In this document, items from the Security Analysis (1) and Final Problem Statement (2) will be revisited. In addition, the strengths and weaknesses of the team’s solution will be discussed.

# Problem Description

Many times, within the classroom, it is difficult for a professor to receive anything close to real time feedback in order to know how a student is currently doing. Some schools have adapted to use online databases to give their students access to their grades. Often times that system is only accurate at midterms, or end of terms, or it only includes major assessments such as midterm tests, finals, or big projects. In addition, there are no tools currently available that allow for real-time quiz creation, where a quiz can be sent out electronically and automatically processed and inserted into the gradebook while at the same time, giving the teacher valuable information as to whether more time and examples are needed on a particular concept.

The goal of the development team was to create a software that allows a professor to create an online assessment that could be completed and then automatically graded, in the framework of a class, to allow for follow-up lessons on the problem areas. Also included in this solution will be the capability of a student to check on his or her grades. This will give all involved parties real time feedback in order to allocate lesson time and resources to where they would be most useful.

Below is the list of features Gradebook Assistant contains:

* Ability for students and professors to login
* Ability to display the appropriate data for each user
* Ability to track the courses and sections a professor teaches
* Ability to track the courses and sections a student is enrolled in
* Ability for a student to check their grades
* Ability for a professor to check their students’ grades
* Ability for a professor to create assignments and assign them to a section
  + A professor can also create questions for an assignment
* Ability for students to view and take a quiz
  + Ability to earn a score on a quiz
  + Ability to view results of the quiz
* Ability for a professor to acquire instant results from the assignment

# Solution Description

The solution decided upon by our team was to develop a Java application using Eclipse and use Microsoft SQL Server Management Studio to maintain the database created for the application. A JDBC Driver was utilized to connect the database to the front-end.

## Front-End Discussion

The team wanted the application to be as basic and simplistic as possible, so they decided on using a Java Swing graphical-user-interface (GUI) to create the front-end. The user is able to click buttons, choose items from drop-down menus, and enter information into the application. The graphics are limited, but simple, and the color scheme adheres to primaries. Users can access the java application at any time, and do not have to wait for any information to be updated. The purpose behind the simplicity is to have the focus on the education experience, not on the java application.

## Back-End Discussion

The Java application and GUI use the information from the Gradebook Assistant database to run. The information from the database, such as the tables and attributes, interact with the Java application to create results for the user. In an attempt to provide extra security, accessibility to the database from the Java application is limited. The Gradebook Assistant Database currently exists on a Rose-Hulman Institute of Technology server, Golem, used for the spring quarter CSSE Introduction to Database courses. The database will likely be removed after the end of the spring quarter.

# Key Challenges

* **Challenge**: Inserting data into the database using Microsoft Excel spreadsheets

**Solution**: We ended up having to remove auto-incrementing for many of the IDs because they were making it difficult to insert data that would keep the data consistent among tables. When inserting data, it had to be uploaded in a specific order, which was time consuming, but ultimately the right approach.

**Lesson Learned**: Be organized with the data you want to insert and avoid inserting random data. Also, pay attention to foreign key constraints.

* **Challenge**: Adding attributes after the tables were already established

**Solution**: There was no real solution, other than just to have to add the attributes to the table. This caused the team to have to add columns to the various excel spreadsheets and rethink about the data being entered**.**

**Lesson** **Learned**: Plan ahead and try your best to account for all attributes you will be needing. However, you sometimes think of attributes to add that are necessary but not part of the original plan which is hard to avoid. In this case, be aware of how the addition will affect other tables.

* **Challenge**: Creating the Java application from scratch

**Solution**: It has been awhile since our team has used Java to create an application in Eclipse so we had to refresh on our skills. This just took a little time to become familiar with again which did not prove too difficult.

**Lesson** **Learned**: Choose a front-end approach that you are comfortable with using.

* **Challenge**: Hashing the password for the user

**Solution**: It took a while to find a good way to encrypt the passwords and put them in the database. Once we found a way to encrypt, we had to change a lot of the front-end and back-end to match this.

**Lesson** **Learned**: Figure out how you want information that needs to be encrypted sooner. Have a plan for how the front-end and back-end will handle the encrypted information.

# Database Design

An ER Diagram and Relation Schema are included in Appendix A.

## Security Measures

The Java application accesses the database through a special user account with limited access. All direct interaction between the GUI and the database is done by executing stored procedures. This provides added security to the system by eliminating the need for the Java application to create its own queries. This way, there will not be a way for a command to be executed that can harm the database. Since the Java application can only execute, if someone had access to the front-end, they would be unable to execute malicious queries such as DELETE or DROP. All data passed to the stored procedures are handled as parameters which limits query concatenation.

## Integrity Constraints

Within the database, a multitude of referential integrity constraints exist. Foreign keys are represented as arrows on the Relational Scheme diagram included in Appendix A. In general, deletions and updates are handled by cascading, unless there is a constraint violation.

Some of the main integrity constraints that exist in the system are as follows:

* Question.Points must be greater than 0
* [Student Has Assignments].Grade must be one of ‘A’, ‘B’, C’, ‘D’, ‘F’, or ‘N’
* [Student Has Assignments].Score must be an int greater than or equal to 0
* [Question Has Choices].Choice, Question.CorrectAnswer, and [Student Answers Question].Answer must be an int from 1, 2, 3, or 4

## Stored Procedures

|  |  |
| --- | --- |
| Stored Procedure | Function |
| AnswerQuestion | Allows a user to answer a question belonging to an assignment and inserts the result into the database |
| AssignSectionAssignment | Allows a professor to assign a particular assignment to a section of his or her course and inserts the desired results into the database |
| CompleteAssignment | Allows a student to complete an assignment and assigns and inserts a grade to that student if the assignment has been fully completed |
| CreateAssignment | Allows a professor to create an assignment and insert |
| CreatePassword | Inserts the PIN number for a given user into the GBUser table that acts as a password and ensures that the user PIN number is valid |
| CreateQuestion | Allows a professor to create a question, with 4 answer choices, and updates the database |
| GetAssignments | Selects assignments that have yet to be completed |
| GetChoices | Selects the choices for a given question |
| GetPrompt | Selects the prompt for a given question |
| GetQuestions | Selects the unanswered questions for an assignment based off of a specific student |
| ProfessorGradeReport | Selects the grades of students the professor teaches |
| ProfessorSchedule | Selects the schedule of a professor, allowing the user to view all of their courses or just their sections |
| StudentGradeReport | Selects the grades of a particular student |
| StudentSchedule | Selects the schedule of a student, allowing the user to view all of their courses they are enrolled in |
| UnansweredQuestions | Selects the unanswered question in assignment for a particular student |
| UserLogin | Allows a user to move to the home screen if the UserID and Password are correct, determines the user type, and determines if the user needs to create a password |
| ViewResultsOfAssignment | Selects the results of an assignment |

## Views

The Gradebook Assistant contains three views:

* Assignment Points
* Student Assignment Grade
* Student Unfinished Assignments

They are not accessible through the Java application itself. They are mainly used to make creating the stored procedures easier. In general, the views are not completely necessary since everything is accessed on the GUI through stored procedures.

## Indexes

Additional indexes, other than the ones created automatically, do not exist.

## Triggers

There are five triggers that exist in the database:

* TriggerAnswerQuestion
  + Exists on the Student Answers Question Table
  + See if it answers all the questions in the assignment, if it is, it will grade it
* TriggerCreateAssignment
  + Exists on the Assignment table
  + Either makes the assignment or say that the assignment already exists
* TriggerAssignSection
  + Exists on the
  + After you insert the assignment into the section, it will give it to all students in that section
* TriggerDeleteAssignment
  + Exists on the Assignment table
  + Will handle cascading the appropriate tables when an assignment is deleted
* TriggerDeleteUser
  + Exists on the GBUser table
  + Will handle cascading the appropriate tables when a user is deleted

# Design Analysis

## Strengths

* The system is only able to execute stored procedures, not directly modify data on the database.
* Attribute names are clear and descriptive
* Passwords are encrypted in the database, therefore, it is illegible to anyone simply looking at the GBUser table in the database

## Weaknesses

* We have limited views which are not used very much in the database.
* Because our solution relies on Stored Procedures, every time data needs to be retrieved, inserted, etc., a new Stored Procedure must be created.
* The structure of the Java code for the application is not very elegant. There are many different methods and classes which may be unnecessary.

## Relational Schema

Assignment

AssignmentID ProfessorID

Question

QuestionID Points Prompt CorrectAnswer

GBUser

UserID Username Name PIN

Student

StudentID GradYear

StudentMajor

StudentID Major

Professor

ProfessorID Department

Course

CourseID Course CourseName

Section

SectionID CourseID ProfID

Assignment has questions

AssignmentID QuestionID

Student has assignments

StudentID AID Grade Score

Student is in section

StudentID SectionID

Assignments for section

AssignmentID SectionID

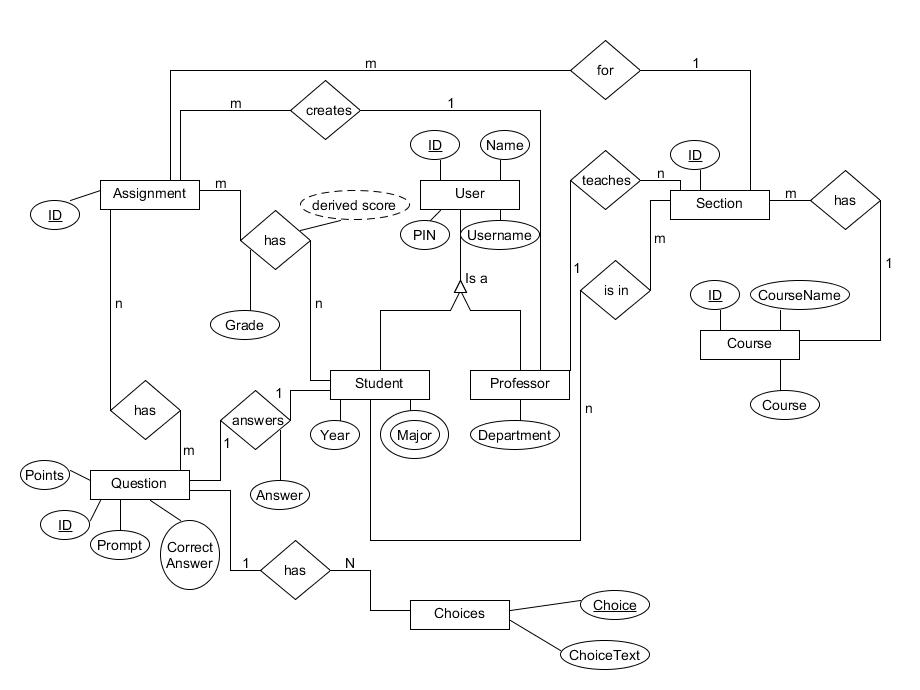
Student answers question

QuestionID StudentID Answer

Question has Choices

QuestionID Choice ChoiceText

## Entity Relationship Diagram



## Explanation of Entity Relationship Diagram

The project solution is built from the creation and existence of Users. A user can either be a student or professor. Professors can create and control Assignments. An assignment has questions, which in turn have choices. An assignment can have many questions, just as a question can have many choices. The professor teaches particular sections, and each section is a part of a course. A course can have many sections, taught by multiple professors. An assignment is to be assigned to a particular section by the professor whom it was created by. A student has assignments, which have questions. A derived score and grade is assigned based off of the student’s results of the assignment. The student is responsible for answering the questions in an assignment. Students are in multiple sections.

# Glossary

# References