

# e Grade Book

David A. Scharton Sr.

March 13, 2009



CAPELLA UNIVERSITY

**TS5356 Advance Application Development**

Vincent Tran **Instructor**

## Table of Contents

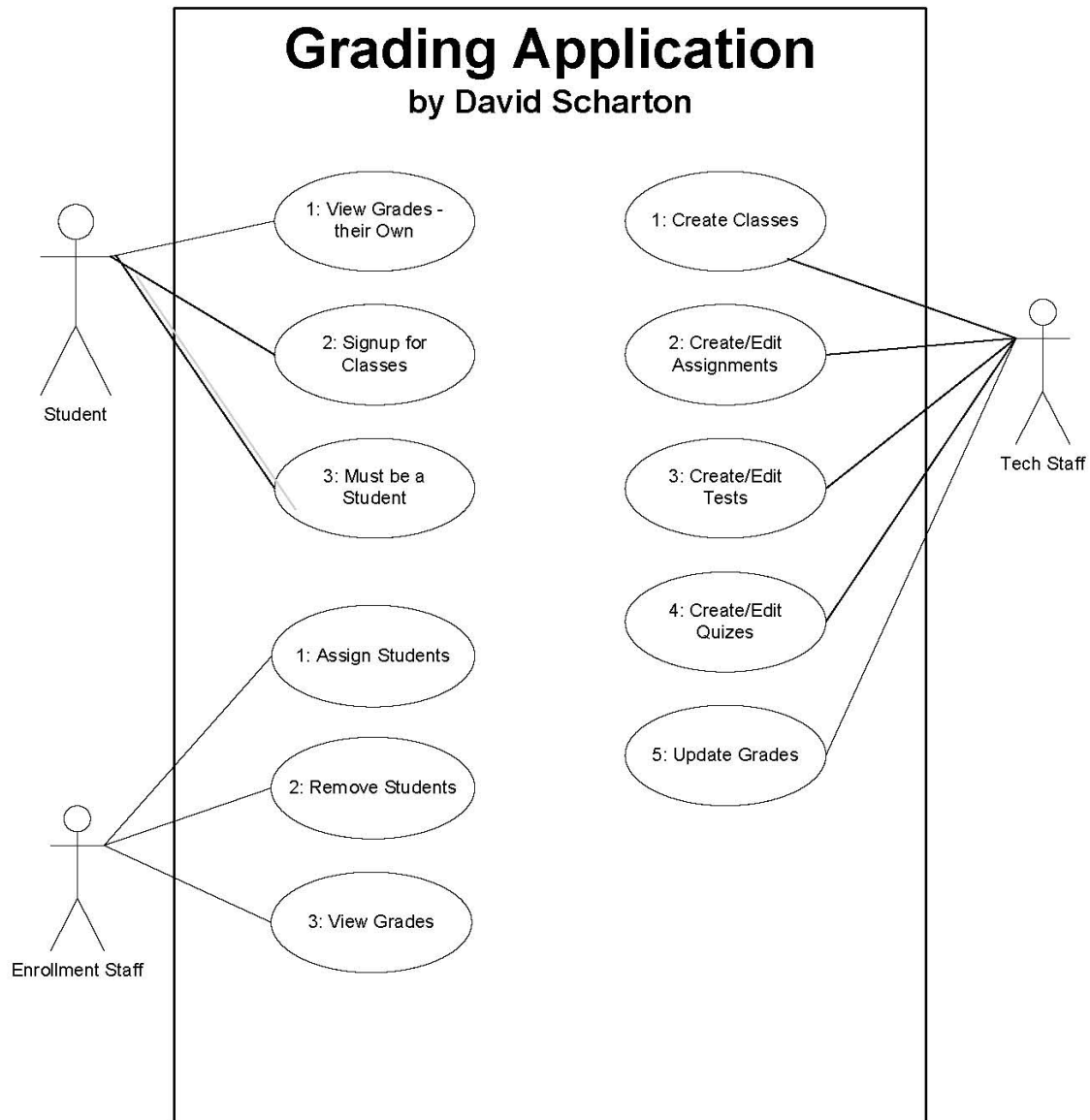
<b>PURPOSE OF DATABASE .....</b>	<b>3</b>
<b>USE CASE MODEL DIAGRAM .....</b>	<b>4</b>
<b>CLASS DIAGRAM.....</b>	<b>8</b>
<b>SEQUENCE DIAGRAM.....</b>	<b>9</b>
<b>ER DIAGRAM AND SCHEME .....</b>	<b>10</b>
ENTITIES DESCRIPTION .....	10
ER DIAGRAMS .....	11
ENTITY RELATIONAL DIAGRAM ( ERD ).....	12
CROW FOOT NOTATION .....	13
<b>SCRIPT FOR POPULATING THE DATABASE.....</b>	<b>14</b>
CREATE TABLES .....	18
<b>REFLECTIONS ON YOUR LEARNING EXPERIENCE .....</b>	<b>21</b>
<b>REFERENCE .....</b>	<b>22</b>

## Purpose of Database

The purpose of this application was to provide tools for Students, Teachers, and Administration to provide a electronic Grade Book to the participants. The Teachers would be able to enter their Quizzes, Assignments, Tests, Grades, and Classes for Students. The Students would be able to added to these classes created by the Teacher through Administration. Administration would be able to view the Student's performance and add Student's to classes.

The three entities would be provided their own menu of options that they would be able to access through a Logon Screen. The Logon Screen would provide the security that will separate the data from what the Teacher and Administration enter. The Student will be provide viewing privileges on their Class performances and Grades.

# Use Case Model Diagram



## Students

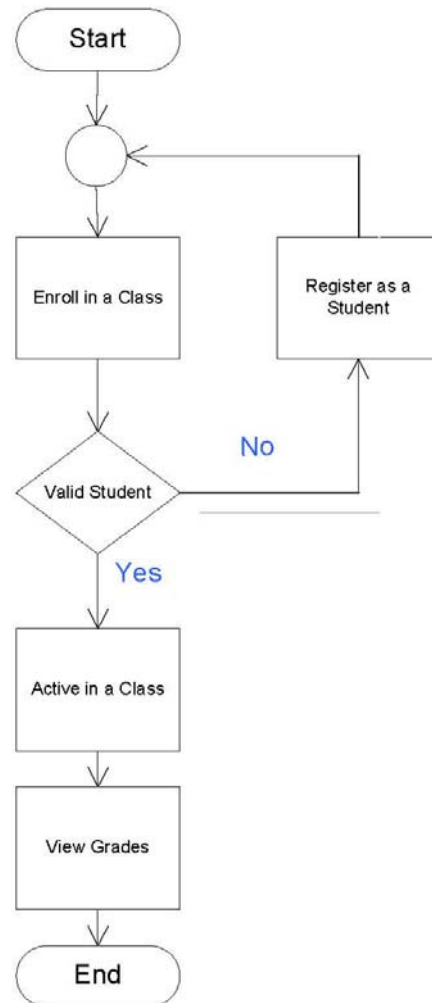
Student enroll into a class but have no control over creating or manipulation of things that go on in the class. They must be a student with the school they are going to be attending.

### Can NOT DO :

Create Tests, Quizzes, or Assignments

### Can DO :

They can only view grades that pertain to them



# Teacher

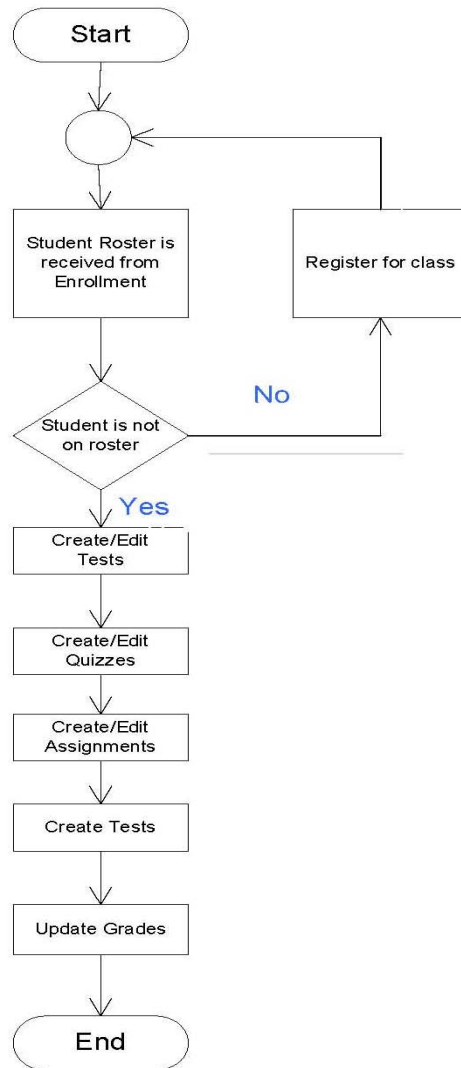
Teachers have control over what is done in the class room. The create tests, quizzes, and assignments. The teacher must be employed at the school they will be teaching the class.

## Can NOT DO :

They can not Edit the enrollment.

## Can DO :

Create classes, tests, quizzes, and assignments.  
Update grades.



# Enrollment

Enrollment is responsible for maintaining enrollment of classes that have been created by teachers. Valid students enroll into valid classes at a school with available enrollment.

## Can NOT DO :

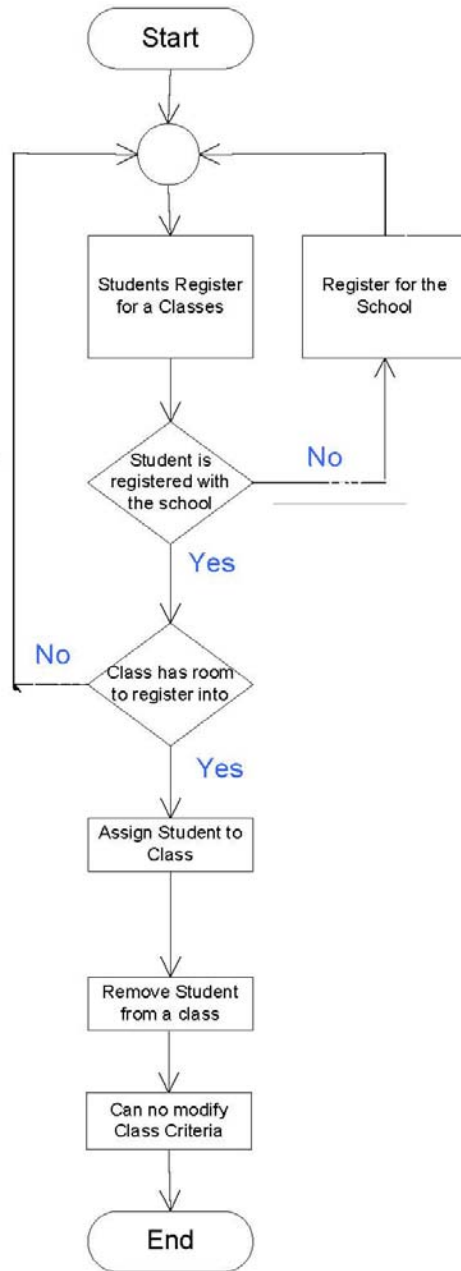
- Assign students to classes.
- Remove students from classes.
- View grades.

## Can DO :

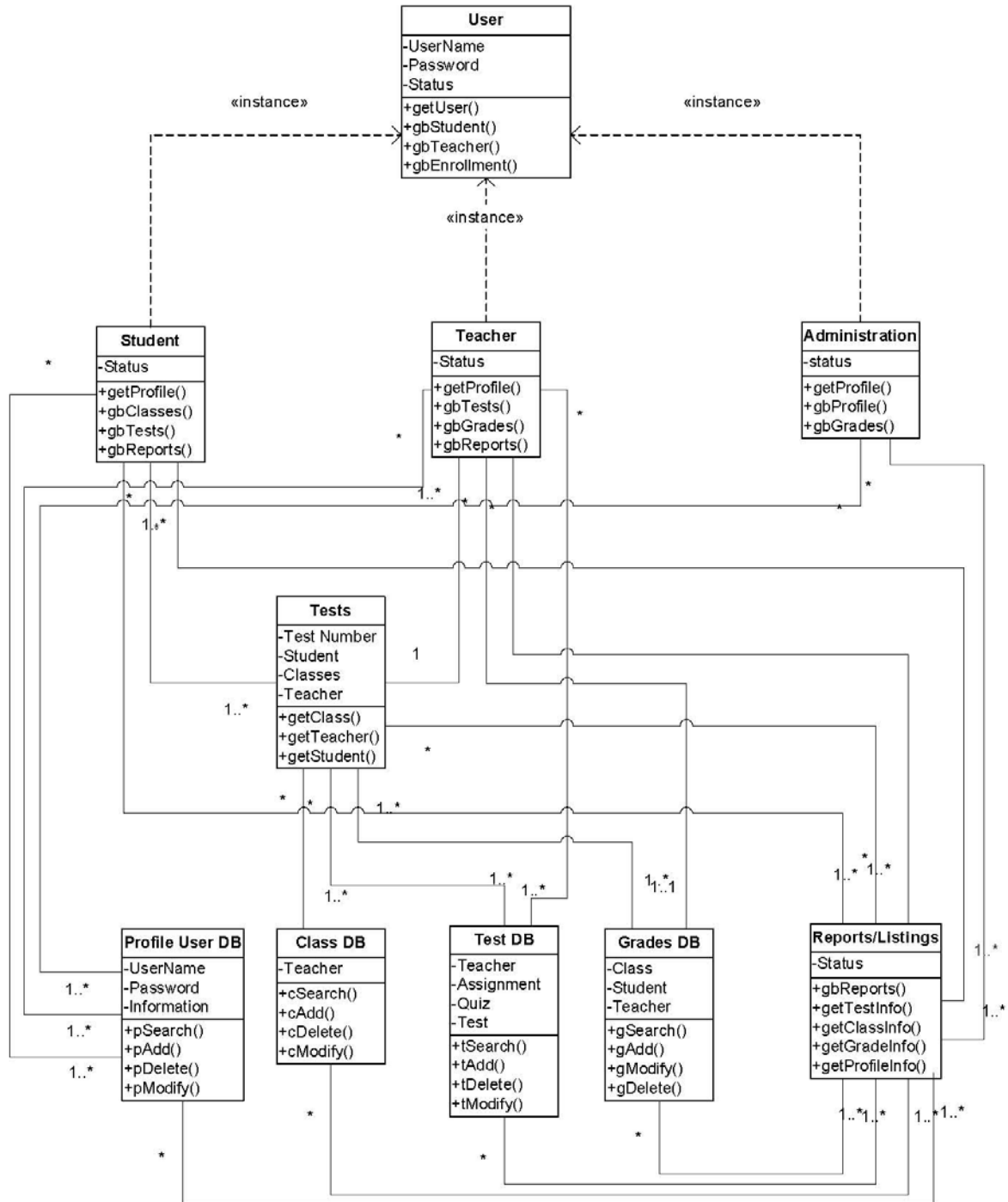
- Not able to modify students grades.

## Alternative things they Can DO :

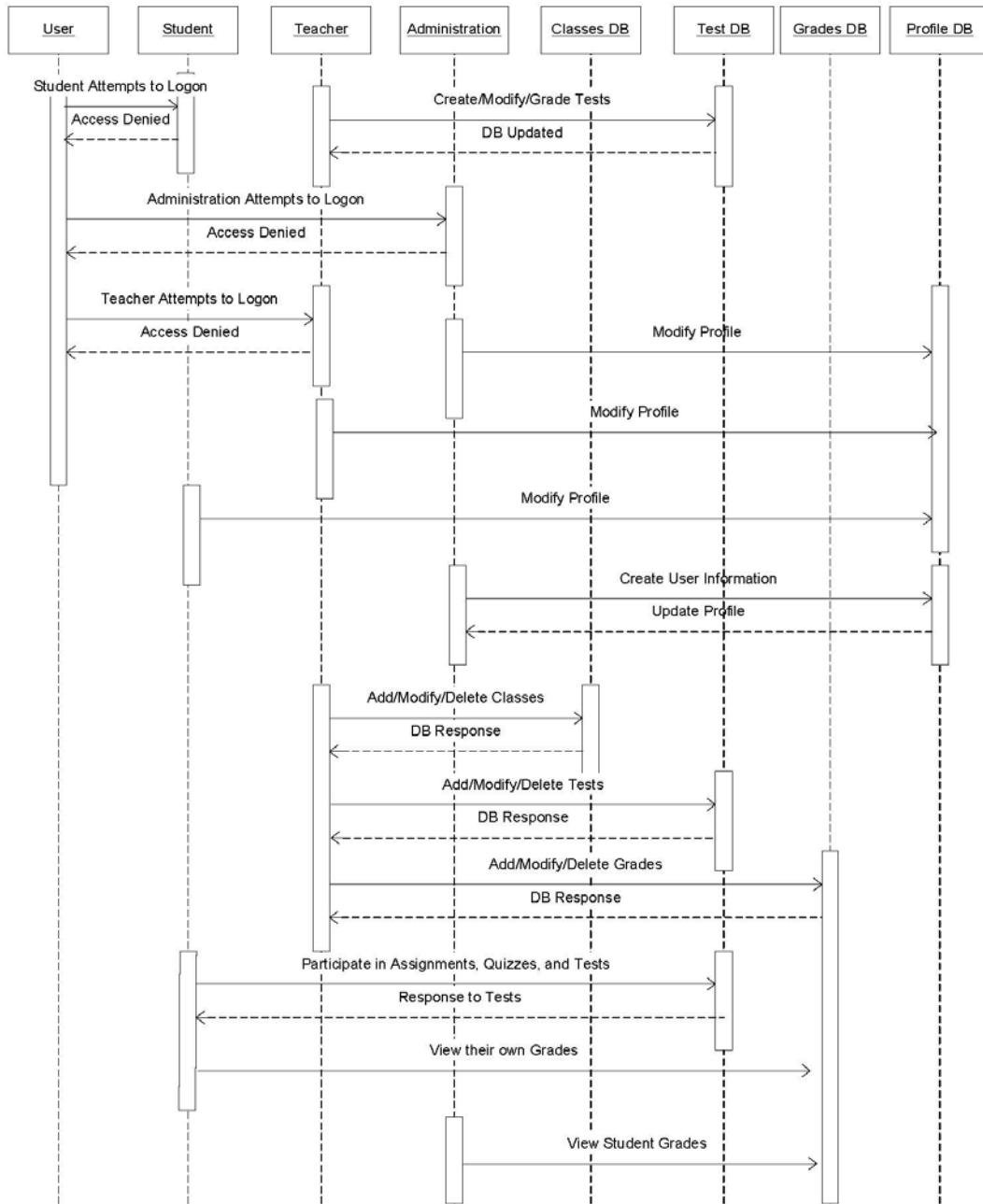
- They are not allowed to modify any teachers curriculum.



# Class Diagram



# Sequence Diagram



# ER Diagram and Scheme

## Entities Description

The Database is designed around 6 entities or tables. The six entities will consist of a security module, user personal information, teacher setup information, student information, and administrative information. The security module will control who can access information about specific individuals tables within the database.

The database is broken down into three groups. The first group is the security section, the second is the primary access tables, and the third is the tables that are that store the specific information in the classes and student history.

**Section One** is pretty clear; it controls what the security issue of who is allowed to use this Grade Book or not.

**Section Two** is directed against the three databases that control the primary data that is in the three main menus ( Students, Teacher, and Administrative )

**Section Three** is the miscellaneous databases that are manipulated by the Teacher and is provided in screens for view purposes for the Student's and Administrative.

I have created a Entity Relational Diagram to illustrate how these tables will interface with each other. The ER Diagrams will be based on the predefined guidelines that I have created.

Database Tables :

Section One :

Security Table

pUsername

Section Two :

Student Database ( Maintained by the Administrative )

sUsername

Teacher Database  
Administrative Database

tUsername  
tUsername

Section Three :

Assignment Database ( Created / Maintained by Teacher )  
Tests Database ( Created / Maintained by Teacher )  
Class Database ( Created / Maintained by Teacher )

aCourseNo + aAssignNo  
tCourseNo + tTestNo  
tClass

## ER Diagrams

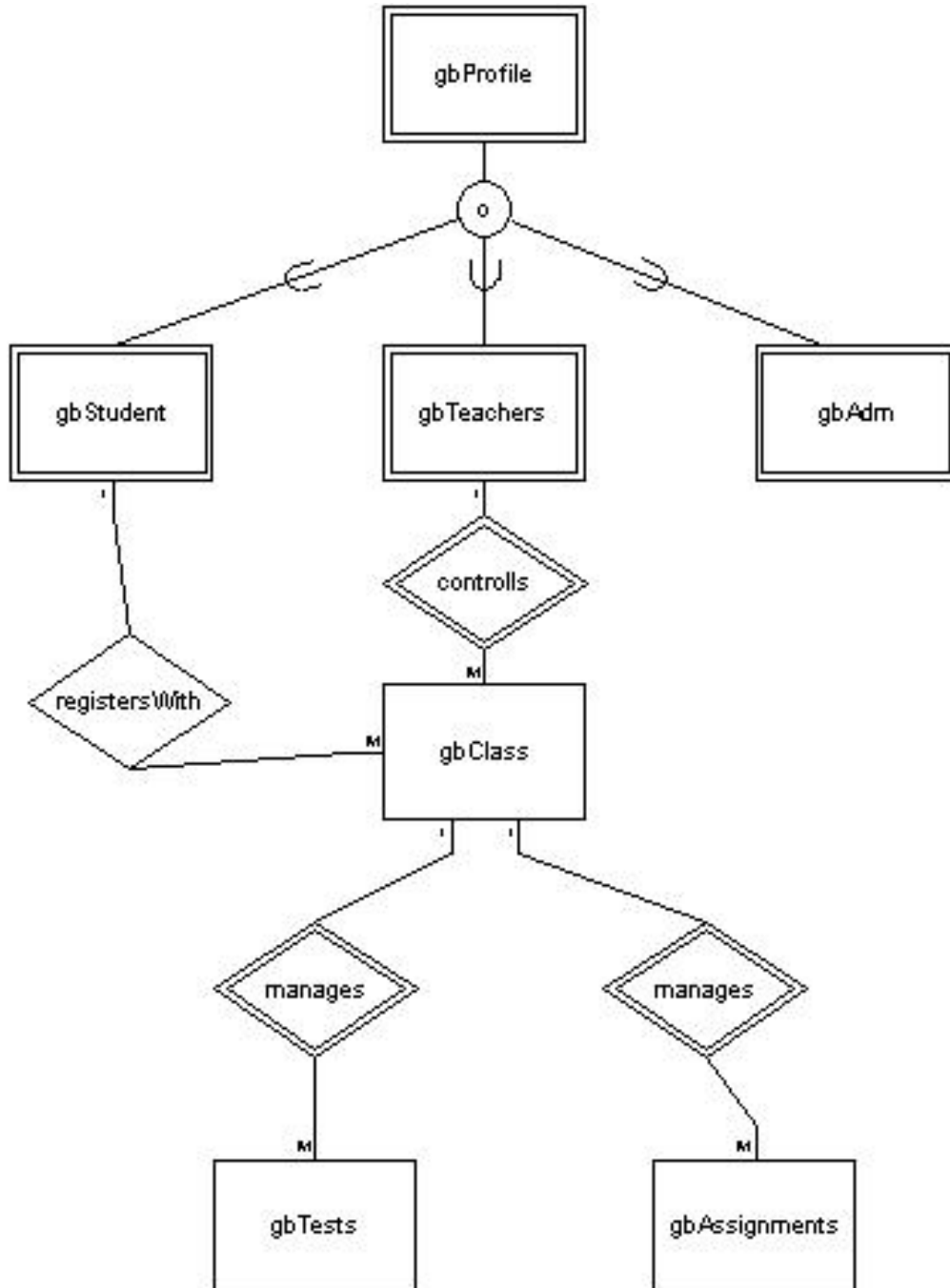
The ER Diagrams were created by using six different Entities. The Entities are structured to provide an environment that will provide a simple electronic Grade Book application. The different entities provide a Conceptual Model that will address the different tasks have been picked for the children to use. A “Conceptual Model represents a global view of the entire database as viewed by the entire organization.” (Rob & Cornel, 2009, p.50) The design was important when creating the six tables so I didn’t encounter usage of fields that were being duplicated. Normalization is what is needed to prevent abnormal usages. “Normalization is a process for evaluating and correcting table structures to minimize data redundancies.” (Rob & Cornel, 2009, p.153)

The six tables start with the controlling table that provides security to the Student, Teacher, and Administrative databases. The specific information is an already created user name and password code. The Administrative Office controls the creation of User Access Codes and this code is tied to all the databases.

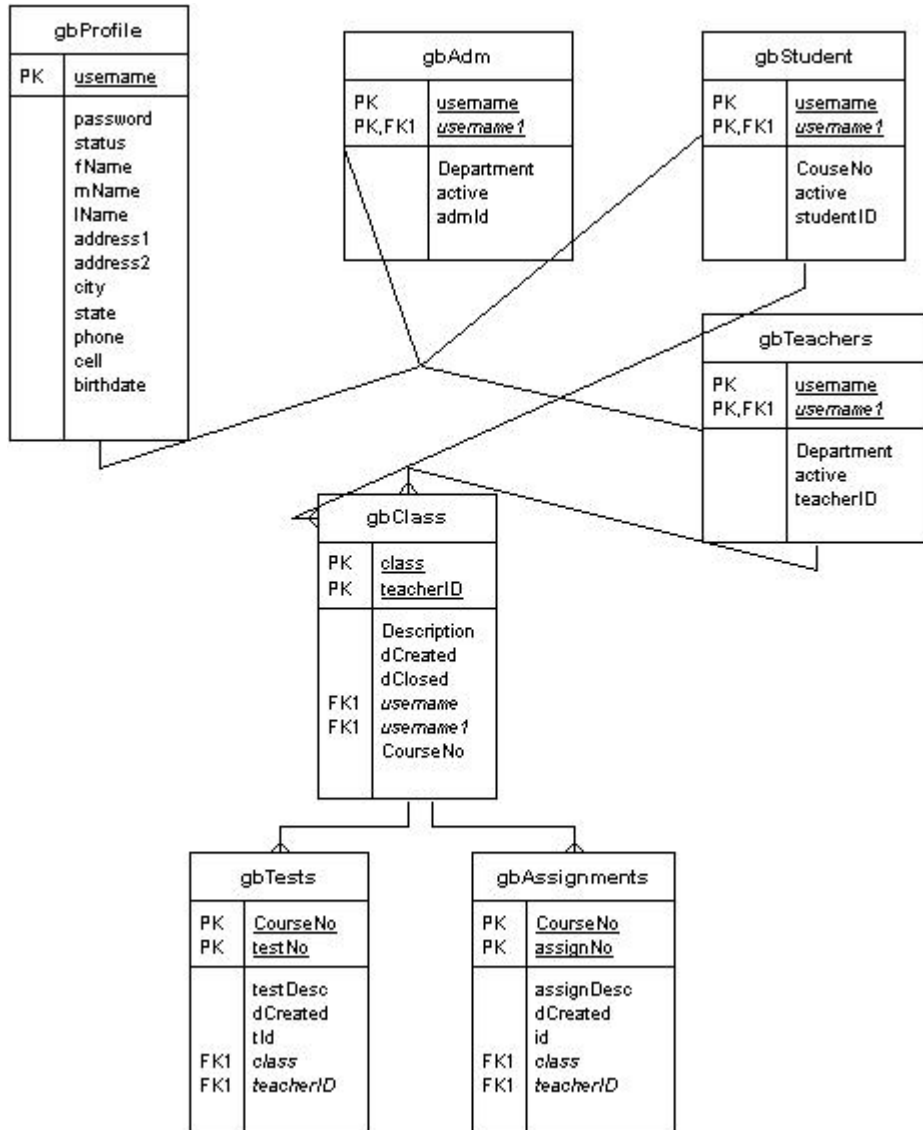
The information that will be manipulated by this Grade Book application will vary from Teacher created information and Administrative creating Student access abilities. The Students and Administrative are also provided the ability to view information that the Teacher creates for their classes and students.

The six tables reflect the information that will be needed to provide the needs we are providing within this database. The reports or queries will be generated by combining the metadata and information that is collected by the database.

## Entity Relational Diagram ( ERD )



## Crow Foot Notation



## Script for Populating the Database

This section will display the SQL code that shows the creating of the tables in the GradeBook Database is as follows :

```
--
-- Create Table      : 'gbProfile'
-- username          :
-- password          :
-- status            :
-- fName             :
-- mName             :
-- lName             :
-- address1          :
-- address2          :
-- city              :
-- state             :
-- phone             :
-- cell              :
-- birthdate         :
--
CREATE TABLE gbProfile (
    username      CHAR(10) NOT NULL UNIQUE,
    password      CHAR(10) NULL,
    status        DECIMAL(1) NULL,
    fName         CHAR(20) NULL,
    mName         CHAR(20) NULL,
    lName         CHAR(30) NULL,
    address1      CHAR(30) NULL,
    address2      CHAR(30) NOT NULL,
    city          CHAR(25) NULL,
    state         CHAR(2) NULL,
    phone         CHAR(15) NOT NULL,
    cell          CHAR(15) NOT NULL,
    birthdate     DATETIME NULL,
CONSTRAINT pk_gbProfile PRIMARY KEY CLUSTERED (username))
GO

--
-- Create Table      : 'gbStudent'
-- username          :
-- username1         : (references gbProfile.username)
-- CouseNo           :
-- active            :
-- studentID         :
--
CREATE TABLE gbStudent (
    username      CHAR(20) NOT NULL UNIQUE,
    username1     CHAR(10) NOT NULL,
    CouseNo       CHAR(30) NULL,
    active        CHAR(1) NULL,
```

```
        studentID          CHAR(20) NULL,
CONSTRAINT pk_gbStudent PRIMARY KEY CLUSTERED (username,username1),
CONSTRAINT fk_gbStudent FOREIGN KEY (username1)
        REFERENCES gbProfile (username)
        ON DELETE NO ACTION
        ON UPDATE CASCADE)
GO

--
-- Create Table      : 'gbTeachers'
-- username          :
-- username1         : (references gbProfile.username)
-- Department       :
-- active            :
-- teacherID        :
--
CREATE TABLE gbTeachers (
        username          CHAR(20) NOT NULL UNIQUE,
        username1        CHAR(10) NOT NULL,
        Department       CHAR(35) NULL UNIQUE,
        active            CHAR(1) NULL UNIQUE,
        teacherID        CHAR(20) NULL UNIQUE,
CONSTRAINT pk_gbTeachers PRIMARY KEY CLUSTERED (username,username1),
CONSTRAINT fk_gbTeachers FOREIGN KEY (username1)
        REFERENCES gbProfile (username)
        ON DELETE NO ACTION
        ON UPDATE CASCADE)
GO

--
-- Create Table      : 'gbAdm'
-- username          :
-- username1         : (references gbProfile.username)
-- Department       :
-- active            :
-- admID            :
--
CREATE TABLE gbAdm (
        username          CHAR(20) NOT NULL UNIQUE,
        username1        CHAR(10) NOT NULL,
        Department       CHAR(35) NULL UNIQUE,
        active            CHAR(1) NULL UNIQUE,
        admID            CHAR(20) NULL,
CONSTRAINT pk_gbAdm PRIMARY KEY CLUSTERED (username,username1),
CONSTRAINT fk_gbAdm FOREIGN KEY (username1)
        REFERENCES gbProfile (username)
        ON DELETE NO ACTION
        ON UPDATE CASCADE)
GO

--
-- Create Table      : 'gbClass'
-- class            :
-- teacherID        :
-- Description       :
```

```
-- dCreated      :
-- dClosed       :
-- username      : (references gbStudent.username)
-- username1     : (references gbStudent.username1)
-- CourseNo      :
--
CREATE TABLE gbClass (
    class          CHAR(20) NOT NULL,
    teacherID      CHAR(20) NOT NULL UNIQUE,
    Description     CHAR(30) NULL,
    dCreated       DATETIME NULL,
    dClosed        DATETIME NULL,
    username       CHAR(20) NULL,
    username1      CHAR(10) NULL,
    CourseNo       CHAR(30) NULL UNIQUE,
CONSTRAINT pk_gbClass PRIMARY KEY CLUSTERED (class,teacherID),
CONSTRAINT fk_gbClass FOREIGN KEY (username,username1)
    REFERENCES gbStudent (username,username1)
    ON UPDATE CASCADE)
GO

--
-- Create Table   : 'gbAssignments'
-- CourseNo      :
-- assignNo      :
-- assignDesc    :
-- dCreated      :
-- id            :
-- class         : (references gbClass.class)
-- teacherID     : (references gbClass.teacherID)
--
CREATE TABLE gbAssignments (
    CourseNo       CHAR(20) NOT NULL UNIQUE,
    assignNo       CHAR(20) NOT NULL,
    assignDesc     CHAR(35) NULL,
    dCreated       DATETIME NULL,
    id            CHAR(20) NULL,
    class          CHAR(20) NOT NULL,
    teacherID      CHAR(20) NOT NULL,
CONSTRAINT pk_gbAssignments PRIMARY KEY CLUSTERED (CourseNo,assignNo),
CONSTRAINT fk_gbAssignments FOREIGN KEY (class,teacherID)
    REFERENCES gbClass (class,teacherID)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
GO

--
-- Create Table   : 'gbTests'
-- CourseNo      :
-- testNo        :
-- testDesc      :
-- dCreated      :
-- tId           :
-- class         : (references gbClass.class)
-- teacherID     : (references gbClass.teacherID)
```

```
--
CREATE TABLE gbTests (
    CourseNo      CHAR(20) NOT NULL UNIQUE,
    testNo        CHAR(20) NOT NULL,
    testDesc      CHAR(35) NULL,
    dCreated      DATETIME NULL,
    tId           CHAR(20) NULL UNIQUE,
    class         CHAR(20) NOT NULL,
    teacherID     CHAR(20) NOT NULL,
CONSTRAINT pk_gbTests PRIMARY KEY CLUSTERED (CourseNo,testNo),
CONSTRAINT fk_gbTests FOREIGN KEY (class,teacherID)
    REFERENCES gbClass (class,teacherID)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
GO

--
-- Permissions for: 'public'
--
GRANT ALL ON gbProfile TO public
GO
GRANT ALL ON gbStudent TO public
GO
GRANT ALL ON gbTeachers TO public
GO
GRANT ALL ON gbAdm TO public
GO
GRANT ALL ON gbClass TO public
GO
GRANT ALL ON gbAssignments TO public
GO
GRANT ALL ON gbTests TO public
GO
```

## Create Tables

```
-- Create Table : 'gbProfile'
```

```
CREATE TABLE gbProfile (  
    username          CHAR(10) NOT NULL UNIQUE,  
    password          CHAR(10) NULL,  
    status            DECIMAL(1) NULL,  
    fName            CHAR(20) NULL,  
    mName            CHAR(20) NULL,  
    lName            CHAR(30) NULL,  
    address1          CHAR(30) NULL,  
    address2          CHAR(30) NOT NULL,  
    city              CHAR(25) NULL,  
    state             CHAR(2) NULL,  
    phone             CHAR(15) NOT NULL,  
    cell              CHAR(15) NOT NULL,  
    birthdate         DATETIME NULL,  
    CONSTRAINT pk_gbProfile PRIMARY KEY CLUSTERED (username))
```

```
-- Create Table : 'gbStudent'
```

```
CREATE TABLE gbStudent (  
    username          CHAR(20) NOT NULL UNIQUE,  
    username1         CHAR(10) NOT NULL,  
    CouseNo           CHAR(30) NULL,  
    active            CHAR(1) NULL,  
    studentID        CHAR(20) NULL,  
    CONSTRAINT pk_gbStudent PRIMARY KEY CLUSTERED (username,username1),  
    CONSTRAINT fk_gbStudent FOREIGN KEY (username1)  
        REFERENCES gbProfile (username)  
        ON DELETE NO ACTION  
        ON UPDATE CASCADE)
```

```
-- Create Table : 'gbTeacher'
```

```
CREATE TABLE gbTeachers (  
    username          CHAR(20) NOT NULL UNIQUE,  
    username1         CHAR(10) NOT NULL,  
    Department        CHAR(35) NULL UNIQUE,  
    active            CHAR(1) NULL UNIQUE,  
    teacherID        CHAR(20) NULL UNIQUE,  
    CONSTRAINT pk_gbTeachers PRIMARY KEY CLUSTERED (username,username1),  
    CONSTRAINT fk_gbTeachers FOREIGN KEY (username1)  
        REFERENCES gbProfile (username)  
        ON DELETE NO ACTION  
        ON UPDATE CASCADE)
```

```
-- Create Table : 'gbAdm'
```

```
CREATE TABLE gbAdm (  
    username      CHAR(20) NOT NULL UNIQUE,  
    username1     CHAR(10) NOT NULL,  
    Department    CHAR(35) NULL UNIQUE,  
    active        CHAR(1) NULL UNIQUE,  
    admID         CHAR(20) NULL,  
    CONSTRAINT pk_gbAdm PRIMARY KEY CLUSTERED (username,username1),  
    CONSTRAINT fk_gbAdm FOREIGN KEY (username1)  
        REFERENCES gbProfile (username)  
        ON DELETE NO ACTION  
        ON UPDATE CASCADE)
```

```
-- Create Table : 'gbClass'
```

```
CREATE TABLE gbClass (  
    class         CHAR(20) NOT NULL,  
    teacherID    CHAR(20) NOT NULL UNIQUE,  
    Description   CHAR(30) NULL,  
    dCreated     DATETIME NULL,  
    dClosed      DATETIME NULL,  
    username     CHAR(20) NULL,  
    username1    CHAR(10) NULL,  
    CourseNo     CHAR(30) NULL UNIQUE,  
    CONSTRAINT pk_gbClass PRIMARY KEY CLUSTERED (class,teacherID),  
    CONSTRAINT fk_gbClass FOREIGN KEY (username,username1)  
        REFERENCES gbStudent (username,username1)  
        ON UPDATE CASCADE)
```

```
-- Create Table : 'gbAssignments'
```

```
CREATE TABLE gbAssignments (  
    CourseNo     CHAR(20) NOT NULL UNIQUE,  
    assignNo     CHAR(20) NOT NULL,  
    assignDesc   CHAR(35) NULL,  
    dCreated     DATETIME NULL,  
    id           CHAR(20) NULL,  
    class        CHAR(20) NOT NULL,  
    teacherID    CHAR(20) NOT NULL,  
    CONSTRAINT pk_gbAssignments PRIMARY KEY CLUSTERED (CourseNo,assignNo),  
    CONSTRAINT fk_gbAssignments FOREIGN KEY (class,teacherID)  
        REFERENCES gbClass (class,teacherID)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE)
```

```
-- Create Table : 'gbTests'
```

```
CREATE TABLE gbTests (  
    CourseNo      CHAR(20) NOT NULL UNIQUE,  
    testNo        CHAR(20) NOT NULL,  
    testDesc      CHAR(35) NULL,  
    dCreated      DATETIME NULL,  
    tId           CHAR(20) NULL UNIQUE,  
    class         CHAR(20) NOT NULL,  
    teacherID     CHAR(20) NOT NULL,  
    CONSTRAINT pk_gbTests PRIMARY KEY CLUSTERED (CourseNo,testNo),  
    CONSTRAINT fk_gbTests FOREIGN KEY (class,teacherID)  
        REFERENCES gbClass (class,teacherID)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE)
```

The first table to be address would be the Security Table. The security entity would allow the user ( Student, Teacher, or Administrative ) to logon using an unique logon code. The Administrative department ( using their menu options ) can create/modify/delete existing users.

## Reflections on your learning experience

TS5356 has been a real experience. I have learned a lot about Java Programming but still have a long ways to go. I know that accessing the database information can be handled easier using created classes that direct the logic easier but still trying to understand the database environment in that area has brought confusion with that knowledge.

I will continue with my goal to learn this specific language and know with more time this project could have been more favorable in presentation.

## Reference

Rob & Coronel, (2009). Database Systems: Design, Implementation, and Management, Eighth Edition. Boston: Course Technology.