Web Application Development Using UML

Dilip Kothamasu
West Chester University
West Chester, PA - 19382
dk603365@wcupa.edu

Zhen Jiang
Department of Computer Science
Information Assurance Center
West Chester University
West Chester, PA - 19382
zjiang@wcupa.edu

INDEX:

1. Goal

The goal of the Independent study is to develop a Student Grading System, a web based application in Java using Object Oriented Design in Unified Modeling Language (UML). UML is used for developing projects in Object Oriented Design and helps in specifying, visualizing, designing the structure software applications meeting all the requirements of a project. Student Grading System is a web-based application that deals with maintaining grades of various students in a particular course by instructor. This system will be providing interfaces for professor to creating course, adding students to a course, assigning grades to students. Also provides various reports for instructors showing the average in a class or for a particular student. The GUI of the system is very user friendly.

2. Requirement Specifications in Use Cases

Use cases are a part of UML, are a way of representing requirements of a project in a effective way. Following are the use cases that are depicted from the requirements of project:

1. **Student**:

   a. A student can view his grade by entering his/her student id, also by selecting the course sequence number, semester, year and section in which he wants to check his grades.

2. **Instructor**:

   a. Instructor can create a class by entering course sequence number, semester, year and section. After creating class, the instructor will be asked about the number of students in the class and then each student's id is entered.
   b. He can even edit student by entering course sequence number, semester, year, section and student id.
   c. Instructor can add/update/delete a student's grade by entering course sequence number, semester, year, section and student's id.

3. **Reports**:

   a. A report for student showing all the grades obtained by him during the semester including the final grade.
   b. Graphical report for Instructor showing the final grades of all the students in a class.
   c. Graphical report for Instructor showing the in-depth details of a grade of a student.

3. Analysis

Analysis is an important step in the development of a project and UML provides a better way of analyzing the requirements. There are many ways to draw UML diagrams from the requirements. The method used to gather information and draw UML [1] diagrams is an easier and effective way. This is done in 5 major steps [2].

STEP 1: Finding all the information needed for the system.

The instructor will create a new class sequence number such as CSC581 at the start of each semester for each class. So to track the class, we need to take the sequence number, the year in which it was started, the semester, and the section. After feeding these details the instructor then adds students to this class by entering the number of students and their student numbers. At each time he conducts a test or quiz he evaluates them and then enters the marks into the system. But the number of tests or quiz is not defined at the beginning of the semester. It also depends on the instructor and is not same for all the classes. The student then checks his grade by entering the student number, the class sequence number, the year, the semester and the section. The instructor can even update a student's record like by changing the marks.

This is the description or the information should be gather from the requirement specifications. Understanding the requirements and collecting the pieces of information should be done carefully, otherwise it may lead in developing a project something different from the one given in requirements. From the information available in the above description we can trace out some important pieces of information which are listed below.

1. Student number
2. Class/Course sequence number
3. Course description
4. Section number
5. Year
6. Semester
7. Instructor
8. Homework
9. Quiz
10. Test
11. Project
12. Letter grade
13. Student
14. Class/Course
15. Grade

STEP 2: Discussing whether the above mentioned pieces of information can be defined as objects or not

For any piece of information to be a class or object it should have:

a. Some action.
b. Cause change of its environment.

1. Student number

   Student number is the one that holds the identification of a student through which he can see the grade in a class/course. There is no action associated with student number because student number is just used to know the grade of a student and change in its value doesn't effect its environment. Examples are 603365, 603366, etc.

2. Class/Course sequence number

   Class/Course sequence number is of the format CSC 581 which is used to identify a course. This has no action associated like deleting the course sequence, as it is only a name given to course and changing the value of it doesn't change the environment. Example if we consider CSC 581 as an object then changing the value of it should reflect some change in the environment. But there is not change as whole course remains the same but only one of the attributes of the course has changed. Examples are CSC581, CSC540, CSC560, etc.

3. Course Description

   Course description is something similar to Course Sequence number. It is also an attribute of course, which has no action and has no effect on the environment due to its change. Consider the course name as Software Development, there is no action like deleting the course name because we only delete the course if the course is removed from the list but not the course description "Software Development" as of which it has no separate identity for itself.

4. Section number

   Section number is a number given to a section like Section 80. Section number is also not an object because it has not action and the change of value does not affect the system. It should be an attribute of a course, which will give some information about the course. Examples are section 80, section 81.

5. Year

   Year describes to which year does a course belong and has no action associated with it. So Year is not an object. For example if we consider year as an object, then suppose its value at a particular time is 2005.Even if we

change the value to 2003 or 2004. It will have not effect on the system. The system remains as it is. Examples are 2005, 2004.

6. Semester

   Semester describes to which semester does a course belong and has no action associated with it. So Semester is not an object. For example if we consider semester as an object, then suppose its value at a particular time is spring. Even if we change the value to fall. It will have not effect on the system. The system remains as it is. Examples are spring, fall semesters.

7. Instructor

   Instructor is on top of all the different pieces of information. Instructor is only a user of the system. He just enters data into the system. There is nothing like deleting the instructor or updating the instructor. So instructor is not an object. From the Tip 1: given in the handout which says that "Don't expect you can have super power software that can handle everything. There is no universal solution. But this can be considered as a user (an Actor) to the system we are designing (Discussed later).

8. Homework

   Homework does have effect on the system. Changing the value of homework changes the grade. For example if by mistake the homework values for a class have been entered as 0 and later changed to the original values, it will have a great effect on the overall grade of the student. But it doesn't have direct effect on the system. So it is a deciding factor for Grade. So homework cannot be a object.

9. Quiz

   Same is the case with Quiz as is the case with Homework. So Quiz is also an attribute of Grade and not an object. Examples are Quiz1, Quiz2 ,etc.

10. Test

    Same is the case with Test as is the case with homework. So Test is also an attribute of Grade and not an object. Objects are like Test1,Test2, etc.

11. Project

    Same is the case with Project as is the case with Homework. So Project is also an attribute of Grade.

12. Letter grade

    Letter grade is just an alphabet which represents the value of Grade at a particular instance. It is something like a name given to Grade. Letter grade has no action associated with it. So Letter Grade is not an object. Examples are 'A', 'B' and 'C'.

13. Student

    Student has actions associated with it. Deleting a student from a class means there is a lot of change in the environment and this effect a lot on the overall system. Student has various actions like deleting, updating, creating new student. So student is an object in this system.

14. Class/Course

    Class/Course has also actions associated with it. If we delete a class, then there is a lot of change in the environment of the system because deleting a class means deleting the students in that class and all their records. Also there are actions like updating and creating a new classes. So Class/Course is definitely an object.

15. Grade

    Grade depends on the values of Homework, Quiz, and others. Changing of grade has an effect of the environment and also has actions like deleting the grade, updating the grade. It has to be calculated depending on the other values. So grade is an object.

So from all the discussion above these are the following objects:

i. Student1, student2,student3 .....
ii. Course1, course2 ,course3 .....
iii. Grade A, grade B.....

Step 3: Find different kinds of objects and describe their types in classes.

From all the objects obtained in the above discussion we can come out with 3 classes which are STUDENT, COURSE, GRADE.

STEP 4: Find relationships of objects and describe then in class relationship

The student and Course has an association relation. Each student will at least get registered in one of the course. Example student1 gets registered in Course1. Each student gets a grade in a particular course. So Course and Grade have also association relationship.

The student and the grade has an association relationship since each student gets a grade in a particular course. For example consider a student Student1 and he has taken a course CSC581 and he gets a grade ‘A’.

Another important issue while designing a system is security which plays a major role. If the system has various users who are authenticated by there passwords, then there is no problem of changing the data or corruption of data. So there should be some users for this system. From the information available above we can identify two users:

a. Student
b. Instructor

These two have a common functionality associated with them, which is authentication, i.e to verify with the username, and passwords they enter. So we can define one more object instructor who has only one action associated with the system and it is authentication. We can define another class called Instructor. So from student and instructor we can generalize the two classes and define a new class called SystemUser which is a super class to student and instructor.

a. Class diagram

From the above analysis the UML Class diagram is drawn which is show in Figure 1. This shows all the relationships between various classes that have been depicted from the above analysis.

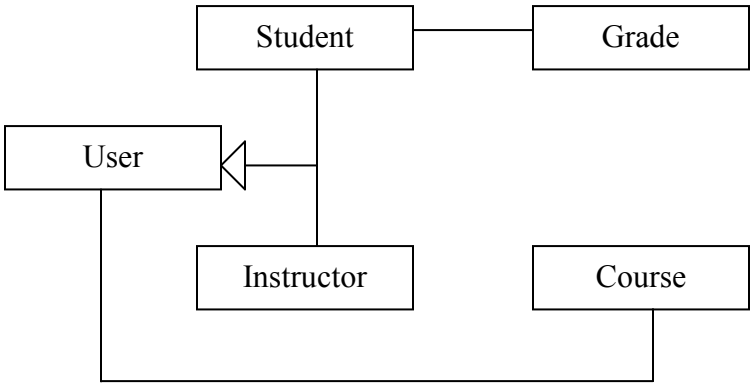

Figure 1 : UML Class diagram showing the relationship between various classes

b. Time Sequence diagram

Time sequence diagram displays the sequence of interaction between objects participating in an action, which consists of the objects and the time at which they are interacting with each other. The time sequence diagram shows a clear picture of which object should be involved while developing a particular action. The picture in Figure 2 shows the time sequence diagram for the various actions of instructor and a student while interacting with the system.

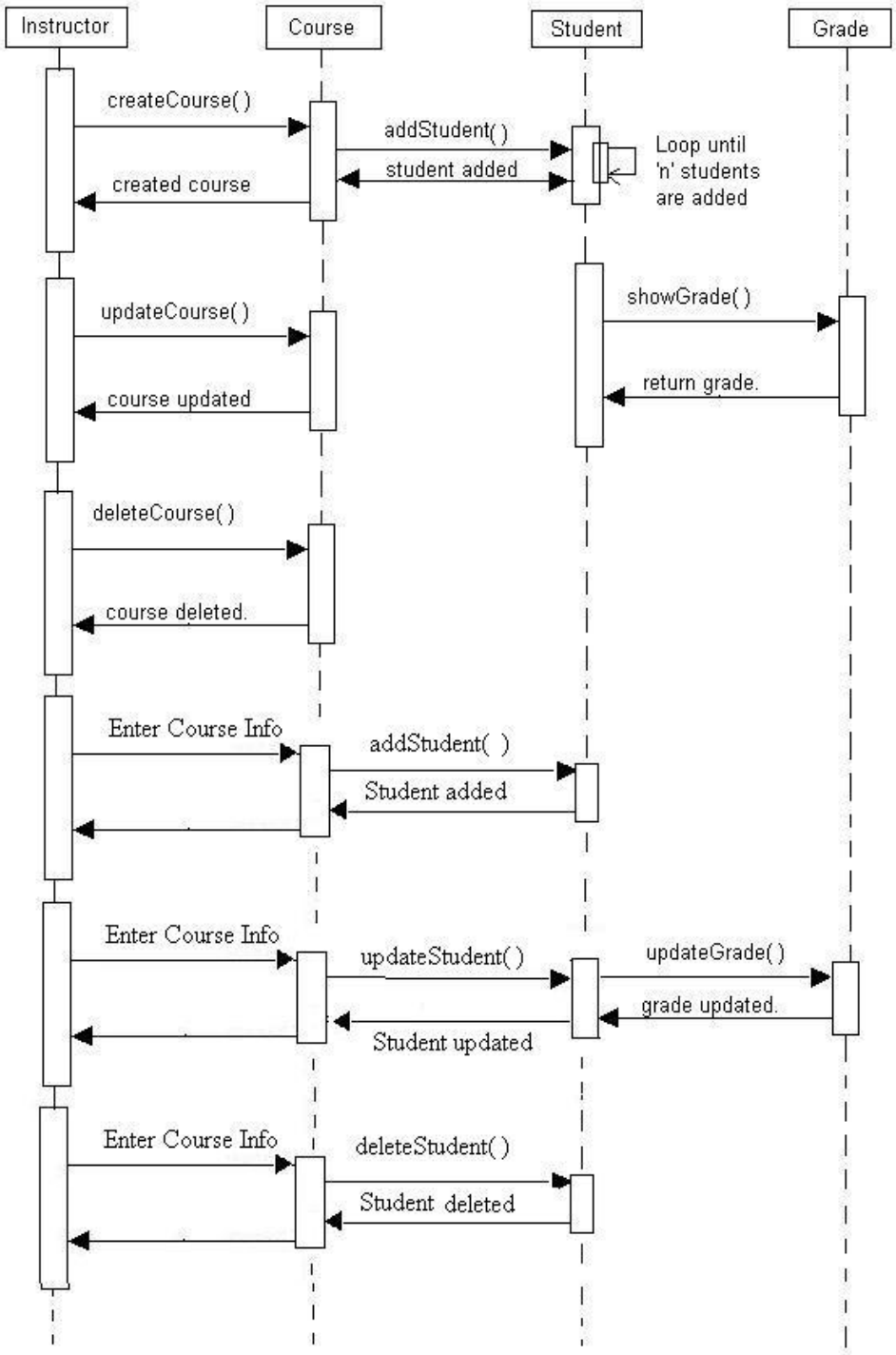

Figure 2: Time sequence diagram showing the sequence of interaction between objects

4. Implementation

a. Constraints

Security:

- Each user of the system is given username and passwords.
- The sessions for each logged in user is maintained and on expiration/logout, re-login is required.
- Database level security is also maintained by securing with username and password.
- Using “role” and “service” concept only user who has a valid service can access that particular service.

Features:

- Each user is given a role and to that role various services are offered. For example a student is assigned a “student” role for which the service that is available is only to check his grades.
- Javascript menu is used for navigation after logging in.
- Chart Director is used for graphical reports.

b. Multi-layer system using UML

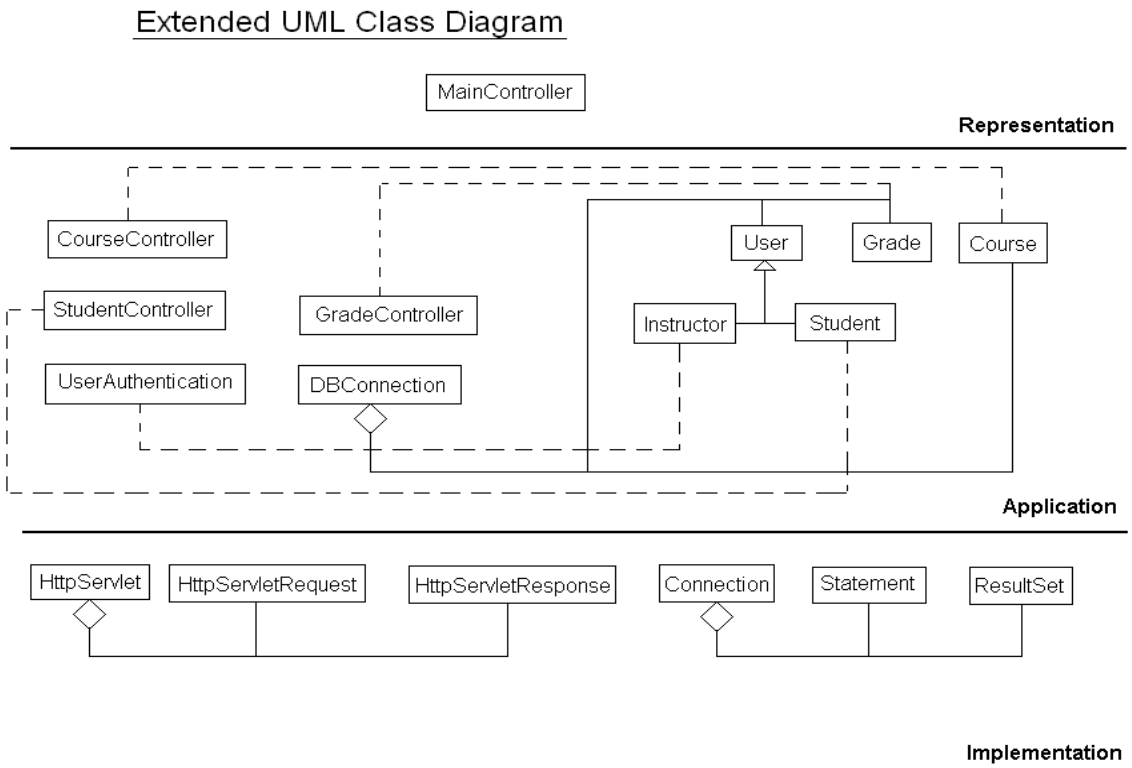

Figure 3: Extended Class diagram showing the relationships between various classes.

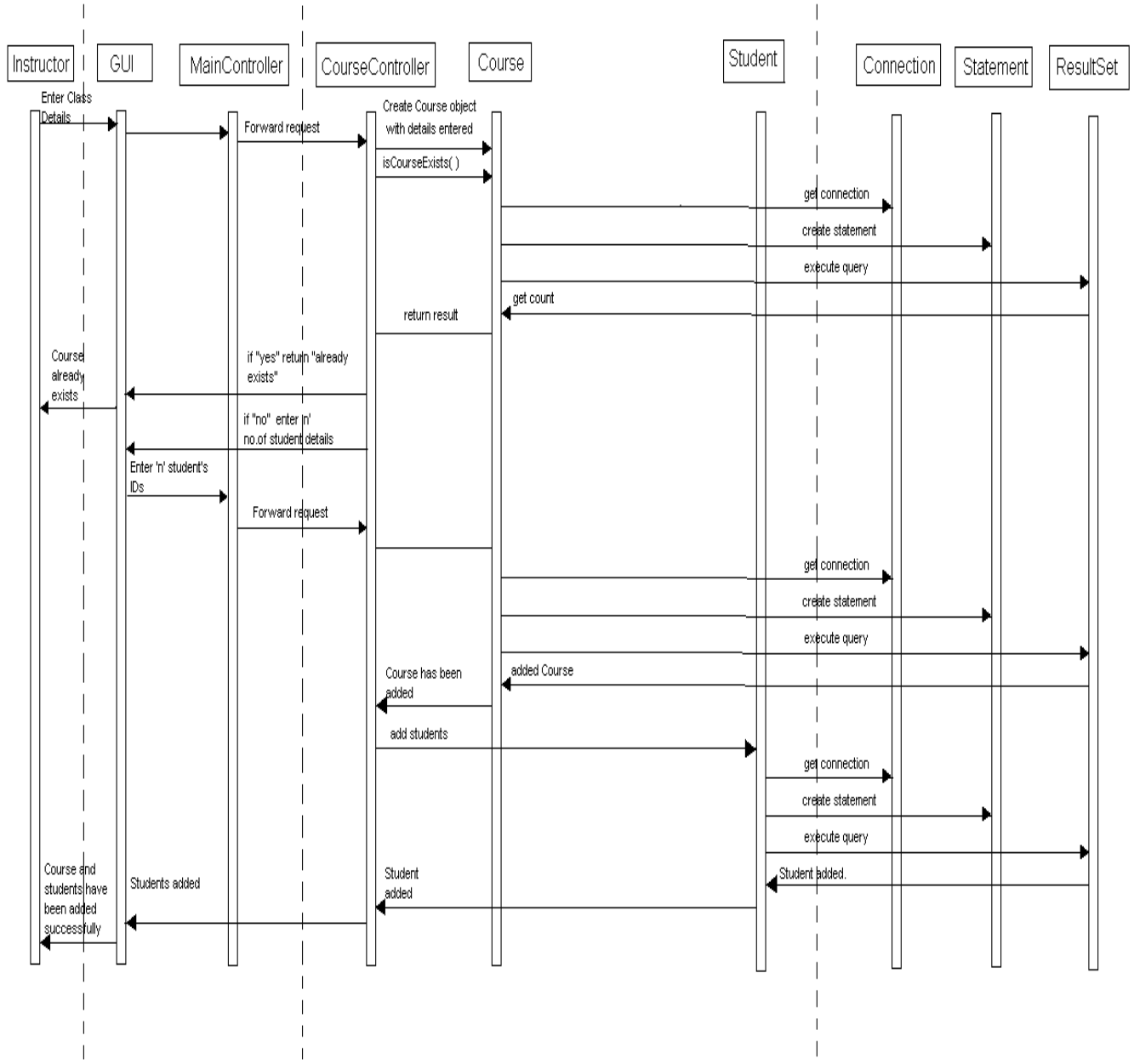

Figure 4: Extended Time sequence diagram for adding a course.

c. Implementation of UML Class diagram & Development

From the class diagram we can design the database by taking each class as a table and have a table which joins the two tables. The resulting database is in 3NF. So the flowing diagram shows the database design of the project. The following figure 5 shows the tables and the dependences on each other.

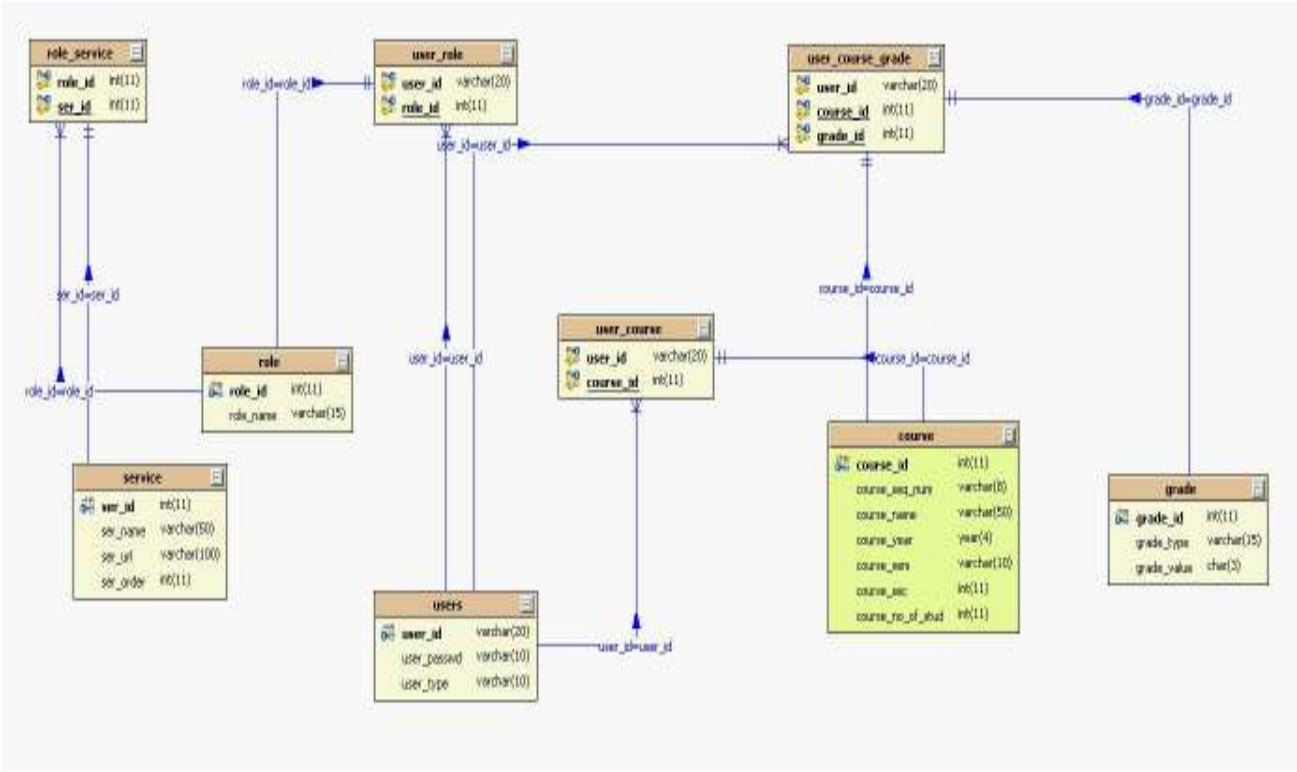

Figure 5: ER-Diagram showing all the tables and the relation ship between them.

Software Architecture:

The architecture used in the project is MVC Architecture [3]. The model object knows about all the data that need to be displayed. It also knows about all the operations that can be applied to transform that object. However, it knows nothing whatever about the GUI, the manner in which the data are to be displayed, nor the GUI actions that are used to manipulate the data. The data are accessed and manipulated through methods that are independent of the GUI. A simple example of a model would be a clock object. It has intrinsic behavior whereby it keeps track of time by updating an internal record of the time ever second. The object would provide methods, which allow view objects to query the current time. It would also provide methods to allow a controller object to set the current time.

The view object refers to the model. It uses the query methods of the model to obtain data from the model and then displays the information. The display can take any form, in the clock example, one view object could display the time as an analogue clock, and another could show it as a digital clock. The different displays would have no bearing whatsoever on the intrinsic behavior of the clock.

The controller object knows about the physical means by which users manipulate data within the model. In a GUI for example, the controller object would receive mouse clicks or keyboard input which it would translate into the manipulator method, which the model understands. For example, the clock could be reset directly by typing the current time into the digital clock display. The controller object associated with the view would know that

a new time had been entered and would call the relevant "SetTime" method for the model object. Figure 6 shows the pictorial representation of the MVC Architecture.

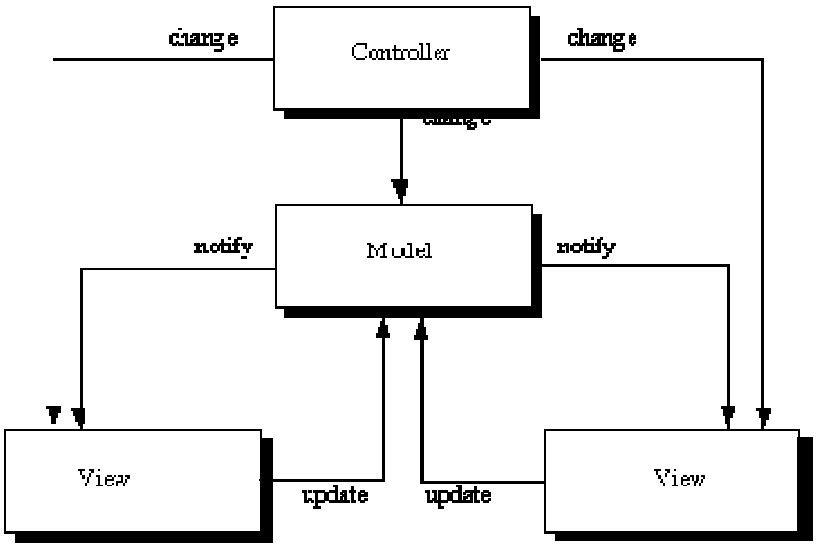

Figure 6: Model of MVC Architecture

d. Environment setup

System Requirements:

1. An operating system as Windows or Linux.
2. Java 1.4.2 or later installed [4].
3. Tomcat 4.0 or later [5].
4. MySQL 4.1 [6].

Following are the steps required to use this software:

1. Open a command prompt and type mysql.
2. Create a database with name as ‘gradingsystem’ at MySQL command prompt using the following command
   mysql > create database gradingsytsem;
3. Change the database to gradingsystem using
   mysql > use gradingsystem;
4. Create the tables listed in APPENDIX I by executing those queries.
5. Grant permissions to the database to a user having username as ‘root’ and password as ‘admin’ using the following command
   mysql > grant all on gradingsystem to root identified by password admin
6. Open a web browser and goto Tomcat manager which asks for username and password. Type in the username and password for Tomcat Manager which can be set by going to Tomcat’s Home Directory -> config -> tomcat-users.xml .

7. Scroll down on the same webpage until you see an option for uploading a war file. Click on browse ,select the file which is ‘indepstudy.war’ from your hard disk and click on Deploy.
8. To access the software type

   http://<server name or IP address >:<port> /indepstudy

   We type indepstudy after the port and IP Address is because tomcat creates a web folder in its webapps directory with the same name as the war file uploaded.

   Eg: http://localhost:8080/indepstudy

5. Reference

1. Zhen Jiang. UML and Pattern, available at http://www.cs.wcupa.edu/~zjiang/csc581index.htm

2. Zhen Jiang. Software Development, available at http://www.cs.wcupa.edu/~zjiang/csc581sd_index.htm

3. The Model View Controller Architecture, available at http://rd13doc.cern.ch/Atlas/Notes/004/Note004-7.html

4. How to install Java, available at http://www.jibble.org/settingupjava.php

5. Tomcat setup, available at http://jakarta.apache.org/tomcat/tomcat-5.0-doc/setup.html

6. Installing Mysql at http://dev.mysql.com/doc/mysql/en/installing.html

APPENDIX I

SQL SCRIPT

```
create table users(
        user_id varchar(20) primary key,
        user_passwd varchar(10),
        user_type varchar(10)
);

create table role(
        role_id integer primary key AUTO_INCREMENT,
        role_name varchar(15)
);

create table service(
        ser_id integer primary key AUTO_INCREMENT,
        ser_name varchar(50),
        ser_url varchar(100),
        ser_order integer
);

create table grade(
        grade_id integer primary key AUTO_INCREMENT,
        grade_type varchar(15),
        grade_value varchar(3)
);

create table course(
        course_id integer primary key AUTO_INCREMENT,
        course_seq_num varchar(8),
        course_name varchar(50),
        course_year date,
        course_sem varchar(10),
        course_sec integer,
        course_no_of_stud integer
);

create table user_role(
        user_id varchar(20) NOT NULL,
        role_id integer NOT NULL,
        primary key(user_id,role_id),
        foreign key (user_id) references users(user_id) on delete cascade,
        foreign key (role_id) references role(role_id) on delete cascade
);

create table role_service(
        role_id integer NOT NULL,
```

```
        ser_id integer NOT NULL,
        primary key (role_id,ser_id),
        foreign key (role_id) references role (role_id) on delete cascade,
        foreign key (ser_id) references service (ser_id) on delete cascade
);

create table user_course(
        user_id varchar(20) NOT NULL,
        course_id integer NOT NULL,
        primary key (user_id,course_id),
        foreign key (user_id) references users(user_id) on delete cascade,
        foreign key (course_id) references course (course_id) on delete cascade
);

create table user_course_grade(
        user_id varchar(20) NOT NULL,
        course_id integer NOT NULL,
        grade_id integer NOT NULL,
        primary key(user_id,course_id,grade_id),
        foreign key (user_id) references users(user_id) on delete cascade,
        foreign key (course_id) references course (course_id) on delete cascade,
        foreign key (grade_id) references grade (grade_id) on delete cascade
);
```