Score my teacher

Weiren Wang, Ying she, Qile Zhu, and Pengjie Zhang
University of Florida
Gainesville, FL 32611, USA
{weirenwang, yshe, valder, pengjiezhang}@ufl.edu

April 4, 2014

Contents

1 Requirement Analysis

1.1 Purpose of this document

The purpose of this document aims to normalize the design of our database system, enhance the visibility of designing procedure, and set a step to organize and control the software development life cycle. This document specifically describes the background of web-based rating system, the disadvantage of current web-based rating system, thus give rise to the need for developing a new rating system that could overcome those problems and satisfy the users demands. In addition, we will briefly state the functionality that would be provided via our forum-like Score my teacher database system, the data acquisition method and system development platform.

1.2 Background

Modern people like to express their ideas, share their ideas and view others ideas, all these social actions have become an indispensable part of our normal life. As with development of W3, it becomes more and more convenient to express, share and view ideas. Forums such like Yahoo, Stack Overflow are two of those big web-based systems which are used by people to satisfy a specific social life needs. There are also many other kinds of web-based systems that have similar functionality. One major kind of these "expressing-sharing" like web-based system is rating system. Plenty of rating systems exist in our surroundings, such as urlhttp://blog.ratemyprofessors.com/, which is used for rating our professors, http://www.apartmentratings.com/, which is used for rating the apartment in the US, http://www.filmratings.com/, which is used for rating films. All this rating systems give users the chance to score the things they have ever experienced with, and also make it possible for users to learn more before they really get stuck with those affairs. Users would never want to waste time on watching a film that is "bad, all of us would never rent a house that has bad public reputation and students would always want to register a course which is instructed by a nice professor who is both helpful and generous on giving high grades. Online scoring of courses and teachers are available almost in every university since people know the student cares. Expression of such information is beneficial to both teachers and students. On one side, students could make their decisions on whether to take a course or not based on the socre for that course which was made by previous students. On the other side, the teacher could adjust the lecture contents and teaching method after he/she views the scores (comments) graded (provided) by the students if he/she was willing to do that. Therefore, a platform for the students to freely express their real feelings about the course and the teacher, freely view and share others thoughts about one course or one teacher is ready to come out.

1.3 Disadvantages of current rating system

Currently, there are mainly two kinds of rating systems that are used for rating our professors/teachers. One is a public rating system, which serves to rate all the professors and courses in all universities, such as http://blog.ratemyprofessors.com/. Although its coverage is wide in terms of the number of professors and universities, there are still several disadvantages:

1. All the information is professor oriented, not course oriented. For example, a professor has held 4 different courses. The average course grade is calculated based on all 4 courses not individual courses. This doesnt make sense. Since for each course, the grading scale should be different.

2. Missing functionality. The semester in which the student took the course is not an input of this system. Nevertheless, this information is important because although a professor holds the same course at different semesters, the overall situation may be different. He or she may have improved a lot based

on the feedback of previous students. In other words, users care more about the ratings for one course held in recent semesters other than the same course held many years ago.

3. The course number is not a key in the current system. Users cannot search the rating information based on the course number. Although the name of the system is Rating My Professor, the issue that users really care about is the course quality of the professor.

4. Incomplete information. Though this kind of system involved many professors, but actually, so many teachers in University of Florida have no records in that system. Thus students of university of Florida often encounter a situation in which there is a blank page after his/her searching.

Another existing rating system is university-oriented. For example, University of Florida has its own rating system https://evaluations.ufl.edu/evals/, this rating system will gather rating details from students at the end of each semester. Students have the chance to express their ideas about the teacher and the course that he has already taken, he or she is also able to give a brief comment about the course. Generally speaking, this system overcomes a lot of disadvantages of the public rating system. However, it is still not user friendly in some aspects:

1. The system is not user-oriented. Users are not able to see the detailed comments and ratings of other users toward a course. The system could only provide overall ratings of a course.

2. The detailed course information is missing, such as the prerequisites. The expectation of users from this system is to get a complete overview of the courses. Course prerequisites are significantly important for the users to decide whether to take the course or not.

3. Each user contributes equally to the overall ratings. This is not often the case. Some students may have dishonest rating for a teacher or a course only because he/she doesnt like that teacher or that course.

4. Furthermore, this system only provides one chance to the students, and this chance often only exists in a specific period (often at the end of each semester). The user cannot rate at anytime that they prefer.

1.4 "Score my teacher" database system

In knowing of such needs of the UF students and the disadvantages of current rating systems, we decide to design our own web-based "Score my teacher database system.

System target: our own "Score my teacher database system aims to provide a user friendly web-based rating system which will solve the problems list above, thus essentially satisfy the needs of the UF students (users).

System functionality: in our system, students can rate their teachers and courses, search both overall and individual detailed ratings about a teacher and/or a course. Major system functions are list below:

1. User sign up: any student in University of Florida has the right to sign up;

2. User login in: any valid user can login in;

3. Start a post: user can start a post to rate one teacher and the corresponding course. By using a 1-5 level ranking to rank the course(teacher) which he/she has taken(be under instruction from) before, and we will get an overall result about this course(teacher), the user can also comment on that course(teacher), which we will save for others' viewing.

4. Search and view post: user can search post based on their inputing searching keyword, the result will show both overall system-summarizing information about the search and individual post which is related to the searching keyword;

5. User reliability calculating mechanism: other users can like or dislike others comment, and this information will affect the post owner's reliability;

6. Course/teacher info notifying: this web-based system will provide users with detailed course/teacher information;

7. User log out: user should have ways to log out;

8. Administrators login in: administrators would have needs to login in in order to edit some data.

System data acquisition: there are two major data acquisition sources. One is users posts, one is data edited by the system administrators.

1. Users post: user would fill in some fields when posting their ratings about a specific teacher and course. After posting, our system will first insert that post into our backend database. At the same time, we will gather useful information from that post and calculate overall rating for a specific teacher and a specific course combination using our calculating algorithm. While user search post based on the keywords he/she provided, our system will first extract all related post from the backend database, present them to the users and extract the overall ratings related and display them.

2. Administrators edition: we will store large information of both courses and teachers into the system beforehand, such as the course prerequisite, the teachers website etc. The information will be inserted into the database by our administrators.

System development platform: since our system is a web-based database system, we will use php to build the systems logic, use html to construct the structure of the webpage, use css to "rendering" the presentation, and use oracle as the backend database.

2 Framework of our database system

Our database system mainly concentrate on software part, so we will focus on the framework of the software. Before detail the framework, we would like to review our system goal: Our system is a forum-like web-based database system, most of the users should be the students in University of Florida. On one hand, after simple steps of registration, user can have his own account and password, he can view posters, rate posters, reply posters, search specific-info-related posters, and most important, he can post his own posters , all of the posters should concerned with teachers and courses (in the poster, user can give his/her opinion about a teacher+course combination). On the other hand, our system should have some mechnism to protect it from dangerous actions either triggered by malicious users or unintentional system activities, and the user should have respect to his/he teacher when give the ratings to the teacher. Our system is a platform to freely express the idea, not a platform to insult or laught at the teacher, so we also need to inspect the content of the posters. The convenient way to execute such safty mechinism is to have some system administrators to monitor the running of the system.

2.1 System structure

After reviewing our system goal, we can obviously design our system as having two subsystems:

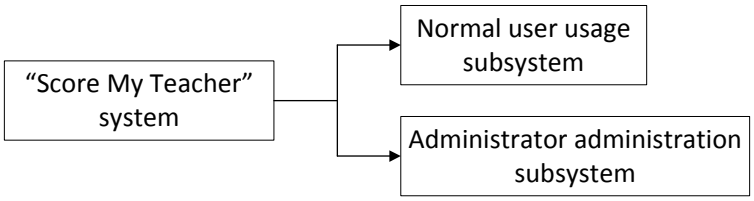

Figure 1: overall framework

The functions that should be provided by these two subsystems are briefly described below:

Number	name of the subsystem	Functions of the subsystem
M1	Normal user usage subsystem	User login up, login in, view poster, rate poster, search poster, post poster, change password, log out
M2	Administrator administration subsystem	Admin the accounts of the users, delete insulting posters , update information about teachers and courses, log out

Table 1: Subsystems and function description

2.2 System flow chart

Normal user can first get to the "login in" webpage of our system, in this webpage, if he/she is not a registered user, then he/she can click the link, be redirected to the "registration" webpage and finish the registration; if he/she is a registered user, he/she can fill the required verification fields and be directed to the "main" webpage of our system; if he/she forget the password, the user can require the system to resend a new password to his/her email. In the "main" webpage, the user can view the posters which are the hottest recently, he can search specefic-info related posters, he/she can change the password, the "change password" webpage will handle this activity, he/she can reply a poster if he/she choose to "reply" a poster, this action will direct him/her to a "reply" webpage, and most important the user can start his/her own poster by click "start new post", this action will direct the user to "start new post" webpage, after reply or start a new poster, the "main" webpage will be showed again, the user can do all other valid actions again. After the user login in, all of the successive webpage can have a link to allow the user to log out.

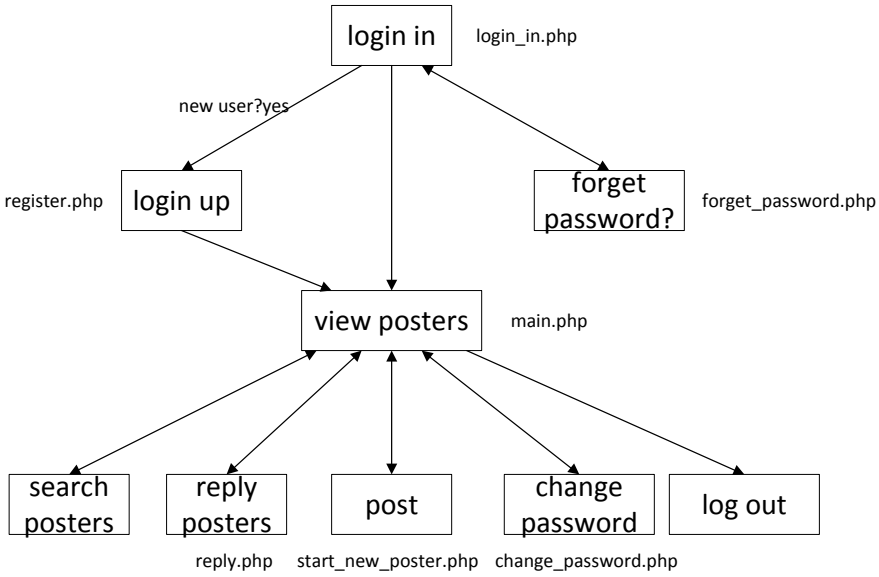

Figure 2: normal user

The administrator can also login in to the system, after login in to the “main” webpage, the administrator can select to delete some insulting posters using “delete poster” webpage, he can update teacher and course info by “updating teacher info” webpage and “update course info” webpage. The administrator can also log out from the system.

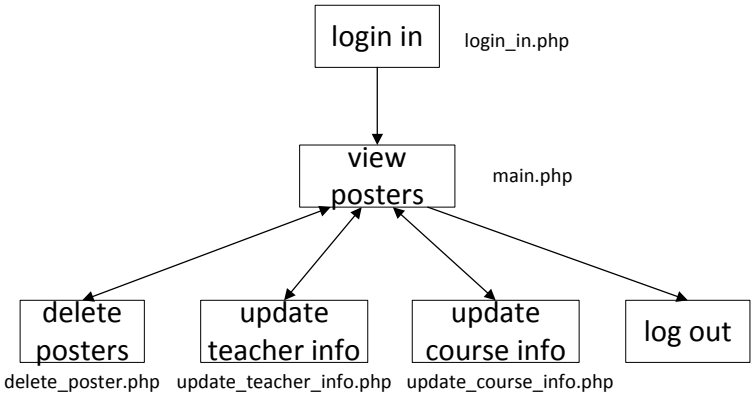

Figure 3: administrator

2.3 Detailed function module description

Module number	Module name	Function description
M1-1	User login up	Any student/teacher can register using a valid account name and password
M1-2	User login in	Users who has registered can login in by provide corresponding account name and password
M1-3	Search posters	Users who has login in can search posters based on the search key word
M1-4	Reply posters	Users who has login in can reply to any poster he is viewing
M1-5	Post	Users who has login in can start his own initial post
M1-6	Change password	Users who has login in can change his/her password
M1-7	Log out	Users should be freely log out anytime he/she like

Table 2: Function modules of the normal user subsystem

Module number	Module name	Function description
M2-1	Delete posters	Administrators can delete poster which is disrespectful to the teacher
M2-2	Update teacher information	Add new coming teachers info or modify the existing teachers info
M2-3	Update course info	Update the current course info, such as the change of credits etc.
M2-4	Log out	Administrators can log out

Table 3: Function modules of the administrator subsystem

3 ER-diagram

Now we should figure out possible entities which can exist in our database system and the relationships between them, i.e. ER diagram. After careful analysis of our system, we came up with following entities (and we also connect the possibly related entities):

1. User (User is a generalization of Student, Administrator and Teacher)
2. Student
3. Administrator
4. Teacher
5. Department
6. Course

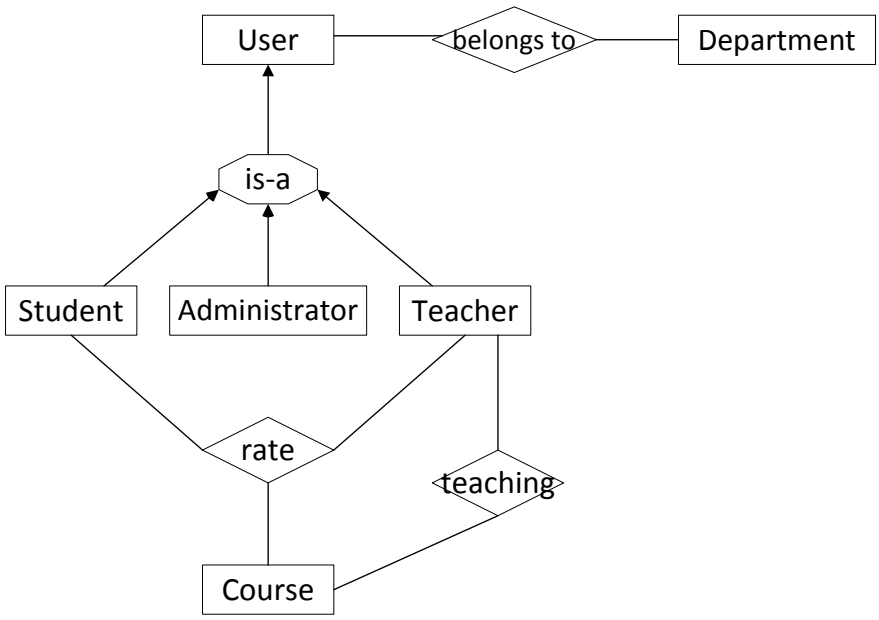

Figure 4: Possible entity sets and possible connections

3.1 Entity sets

According to our real world knowledge, we construct our entities as belows:

Property	Description
name	Primary key. The name of the user. For teacher, this would be the real world name, for student and administrator, this would be the account name (or nickname).
image	Image associate with the user. For a teacher, if there is a real world image then use it, for students and administrators, the image can be any one they like.
email	The contact information of the user.

Table 4: User entity set

Then properties of Student. Since User is a generalization of Student and we have show the properties of User, so below we will only show the properties specialized to Student, the same situation facing by Administrator and Teacher:

Property	Description
studentid	studentid is also a key of Student, this is unique
password	Student have the account password in order to login in as valid user.
reliability	We would collect information from the posters about the reliability of this user by calculating the degree of favorite of his/her posters.

Table 5: Student entity set

Property	Description
password	Administrator should also have password.

Table 6: Administrator entity set

Property	Description
teacherid	teacherid is also a key of Teacher, this is unique.

Table 7: Teacher entity set

Then Department entity set and Course entity set:

Property	Description
deptname	Primary key, the name of the department

Table 8: Department entity set

Property	Description
courseid	The id for this course, primary key
coursename	The name of the course
credit	The credit of the course
description	brief description of this course, including the possible preconditions

Table 9: Course entity set

3.2 Relationship sets

User : Department Since every user should be in one and only one department (we do not concern whether the students were in many departments or not), and every department could have many students, so the relation between them is

$$User : Department := n : 1$$

Teacher : Course Every teacher can teach many courses (at least one) and each course can be taught by several teachers, so

$$Teacher : Course := n : m$$

The teacher can also decide to use which book to teach a specific course.

Student : Teacher: Course (Ternary relationship) Student can rate several courses taught by several teachers as long as he has registered that course+teacher combination before. And every course+teacher combination can be rated by many students. When rate course+teacher, the rating must be associated with specific semester (so we also set the semester as a key), a optional comment, a overall rating (must be NOT NULL but is not key), so

$$Student : Teacher : Course := m : n : h$$

3.3 Overall ER diagram

After above detailed description, our overall ER diagram looks like below:

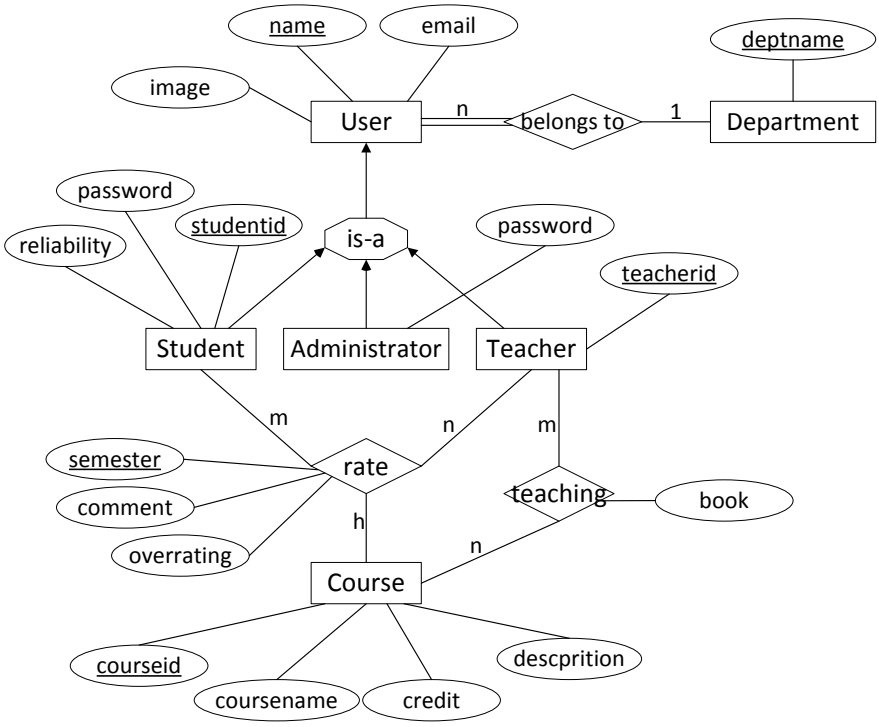

Figure 5: The ER diagram of our system

4 Transformation of our ER Schema into a Relational Schema

Now we should transform the ER diagram into a database schema. In our project we would use relational data model and use Oracle as the backend database, so we would transform our ER diagram to relational schema, i.e. use tables. We have been taught in our DBMS course about how to transform ER diagram into a relational database schema, now we briefly describe the strategy below:

1. for every strong entity set, we would create one relation schema for it;
2. for 1 : n like relationship, we would add the primary key of left relation schema to the right relation schema, and for n : 1 vice verse;
3. for n : m like relationship, we would create a new relation schema, the primary keys of the left and right relation schemas would both be included in the new relation, the new relation may have other keys and/or attributes;
4. for ternary relationship, we would create a new relation schema, the primary keys of the related three relations would be included into the new relation schema plus possible other attributes and/or keys.

In order to avoid conflict with the Oracle sytem, we will prefix all our tables with letters “forum”.

4.1 Department table

We won't store too many things about the department, department name is enough for checking whether the student is coming from a real existed department or not:

$$Depatment(\underline{deptname : string})$$

The sql in creating this table in Oracle is:

```
create table forumDepartment(
        deptname varchar(40) primary key
);
```

4.2 User table

Entity "User" in our ER diagram is a generalization, the schema should be:

$$User(\underline{name : string}, email : string, image : blob, deptname : string)(FK : deptname)$$

And the SQL in creating this table in Oracle is:

```
create table forumUser(
        name varchar(40) primary key,
        email varchar(40)       NOT NULL,
        image blob,
        deptname varchar(40),
        foreign key (deptname) references forumDepartment(deptname)
);
```

4.3 Student table

$$Student(\underline{name : string}, studentid : integer, password : string, reliability : numeric)$$

```
create table forumStudent(
        name varchar(40) primary key,
        studentid int unique,
        password char(40) NOT NULL,
        realiability numeric(2, 1)
);
```

4.4 Administrator table

$$Administrator(\underline{name : string}, password : string)$$

```
create table forumAdministrator(
        name varchar(40) primary key,
        password char(40) NOT NULL
);
```

4.5 Teacher table

$$Teacher(\underline{name : string}, teacherid : integer)$$

```
create table forumTeacher(
        name varchar(40) primary key,
        teacherid int unique
);
```

4.6 Course table

$$Course(\underline{courseid : string}, coursename : string, credit : integer)$$

For course we would like add "coursedescription" as a new attributes, the attributes will briefly describe what the course is about:

```
create table forumCourse(
        courseid varchar(15) primary key,
        coursename varchar(40) NOT NULL,
        coursedescription varchar(100),
        credit int
);
```

4.7 Poster table

Student rating the teacher+course combination, the rating is stored in every poster, so we use the name "Poster" as the relationship name and the poster should have a date to indicate when the poster comes into being, and in the same time, since student can reply others' posters, this means there is a connection between some posters, somehow like "father-son" relationship. In order to tell the kind of the poster (intial poster or replying poster) and clarify the connection between initial poster and replying poster, we decided to create two tables, the content of these two table is a little different. The relation schema for entity "Poster" is as follows:

$$Poster(\underline{studentid : integer, courseid : string, teacherid : integer, semester : string}, comment : string,$$

$$overallrating : numeric)(FK : studentid, courseid, teacherid)$$

For initial poster (father poster), we create table as:

```
create table forumFatherPoster(
        studentid int,
        courseid varchar(15),
        teacherid int,
        semester varchar(15),
        comment varchar(200),
        overallrating numeric(2, 1),
        postid int unique,
        postingdata date,
        primary key (studentid, courseid, teacherid, semester),
        foreign key(studentid) references forumStudent(studentid),
        foreign key(courseid) references forumCourse(courseid),
        foreign key(teacherid) references forumTeacher(teacherid)
);
```

For replying poster, we create a new table as:

```
create table forumSonPoster(
        studentid int,
        comment varchar(200),
        postid int unique,
        postingdata date,
        fatherposterid int,
        primary key (studentid, fatherposterid),
        foreign key(studentid) references forumStudent(studentid),
        foreign key(fatherposterid) references forumFatherPoster(postid)
);
```

4.8 Teaching table

$$Teaching(\underline{teacherid : integer, courseid : string}, book : string)$$

```
create table forumTeaching(
        courseid varchar(15),
        teacherid int,
        book varchar(40),
        primary key (courseid, teacherid),
        foreign key(courseid) references forumCourse(courseid),
        foreign key(teacherid) references forumTeacher(teacherid)
);
```