Web Application Development
Using UML
Student Grading System

Dilip Kothamasu
dk603365@wcupa.edu

Xiang Gao
xg604024@wcupa.edu

Master's in Computer Science
West Chester University
West Chester, PA - 19382

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

1. Objective

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.

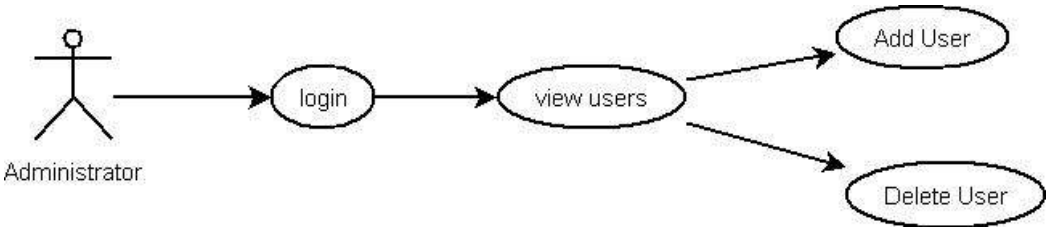
Student Grading System is a web-based application that deals with maintaining grades of various students in a particular course by instructors. This system will be providing interfaces for professors to creating courses, adding students to a course, assigning grades to students. Also provides various 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

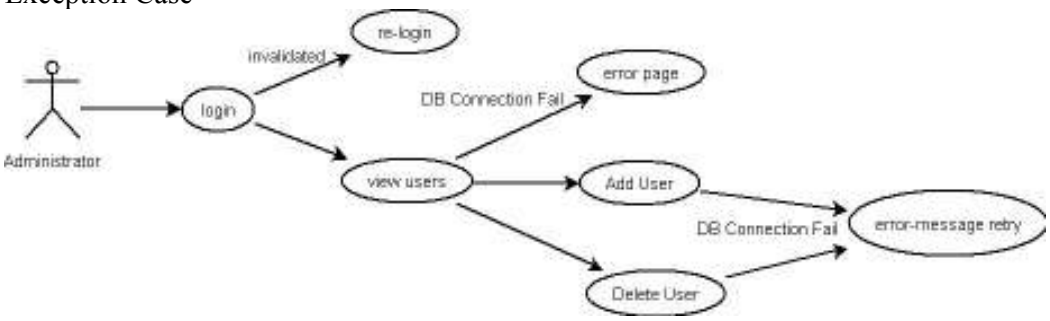
Use cases are a part of UML, are a way of representing requirements of a project in a effective way. There are three use cases in the system. For every use case, there are two diagrams, one for the OK case, and the other describes the exceptional case.

Administrator Use Case

OK Case



Exception Case

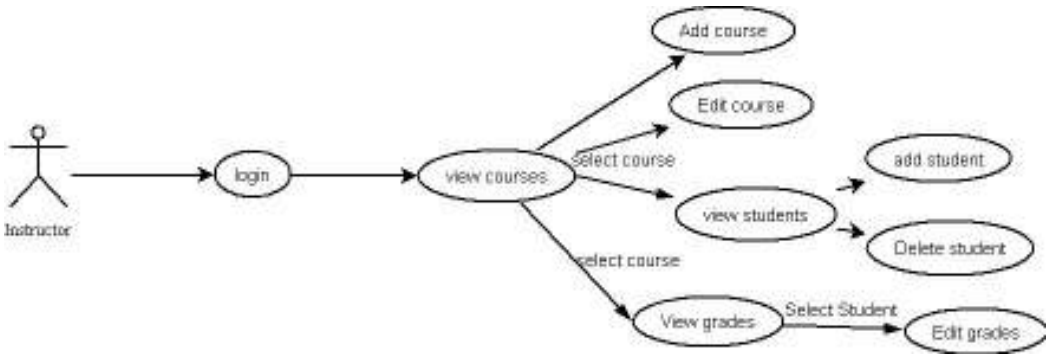


1. Administrator login to the system, a pair of user id and password is needed.
2. If login is failed, then the page is redirected to the login and let administrator try it again.
3. After login, administrator can view the list of users.
4. Administrator can add a user by the user's information such as user id, user type, user name.

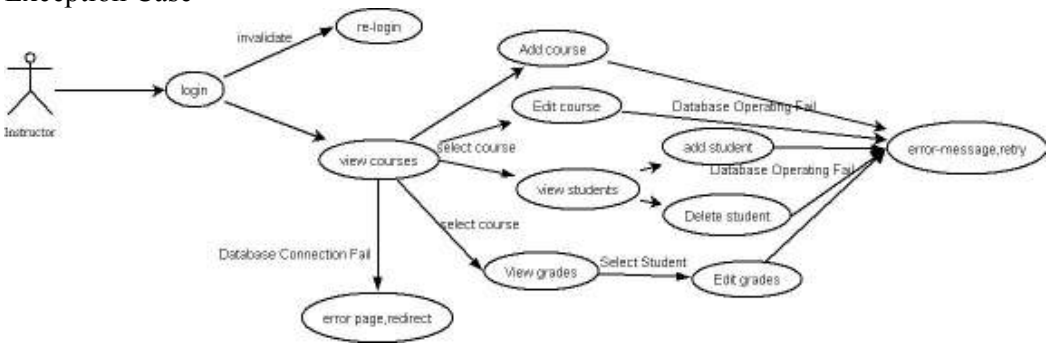
5. Administrator can delete the user record from the system.

Instructor Use Case

OK Case



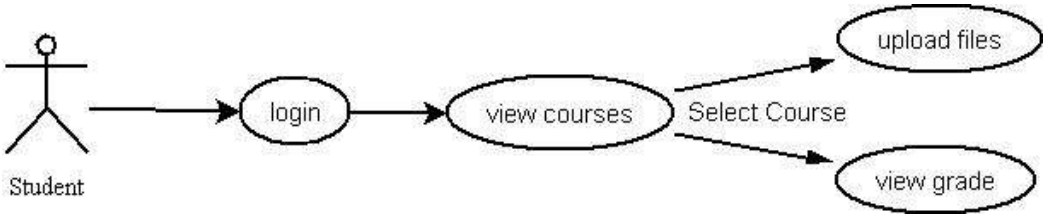
Exception Case



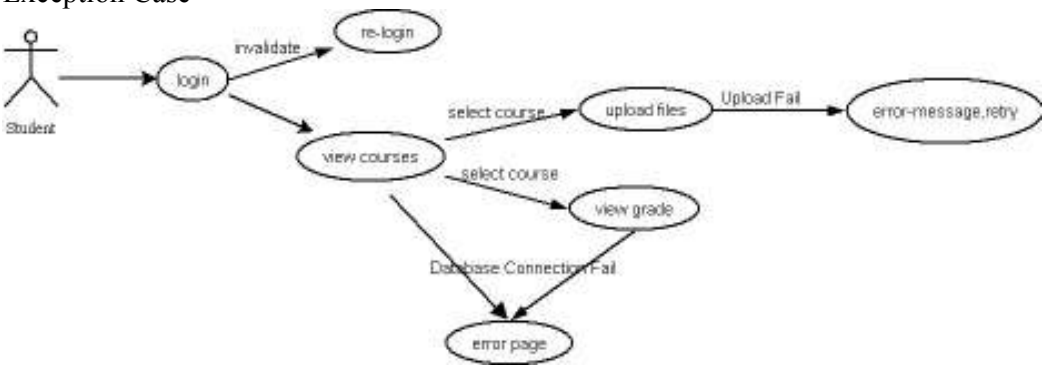
1. Instructor needs log in the system by user id and password firstly.
2. A list of the courses he teaching is shown.
3. The instructor can add a course belongs to him.
4. The instructor can edit the information of the course or delete the record.
5. After select a certain course, the instructor can view the list of the students who are taking the course.
6. The instructor can add a student into the list who takes the course.
7. The instructor can delete the student from the name list of the course.
8. After select a certain course, the instructor can view the list of the of the course.
9. The instructor can edit the grades.

Student Use Case

OK Case



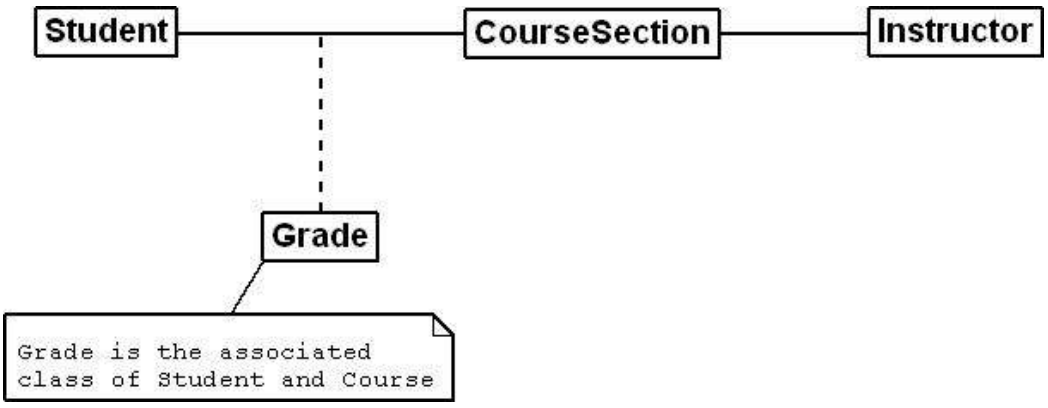
Exception Case



1. Student should login to the system by user id and password
2. After login, the student can view the course list that he is taking
3. The student can upload the assignment files by select the course. If the file is exist, then the new file will overwrite the old one.
4. The student can view the grade by the course.

3. System Classes

Analysis the system described above, we can get the classes of Student, CourseSection, Instructor and Grade, which are shown as below:



The Instructors, can teach courses, input all kinds of assignments, and set the grades. The Students can enroll in the courses, upload the project files, and view the grades. The assignments are the one attribute within CourseSection, so the class, CourseSection, can get the information of all assignments belong to. And the Grade can be specified by one CourseSection, and one Student.

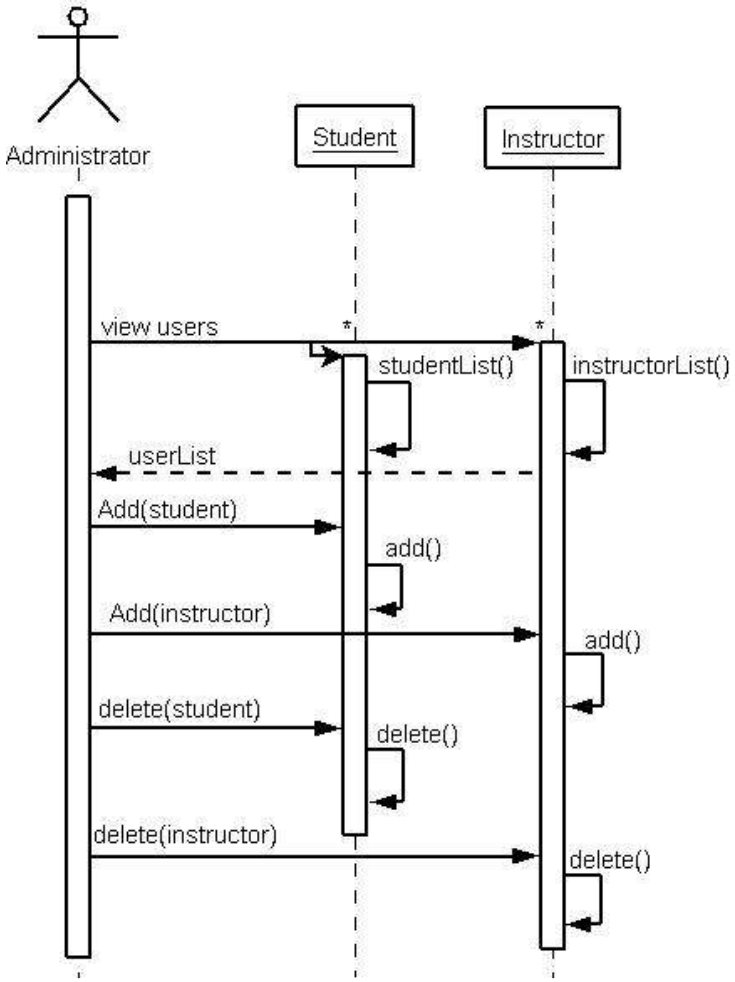
The Instructor teaches a certain course, and the same course, say CSC585, might be taught by different faculties (Instructors). When we talk about a CourseSection, we need the course number and the instructor as well.

4. Time Sequence Diagrams

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.

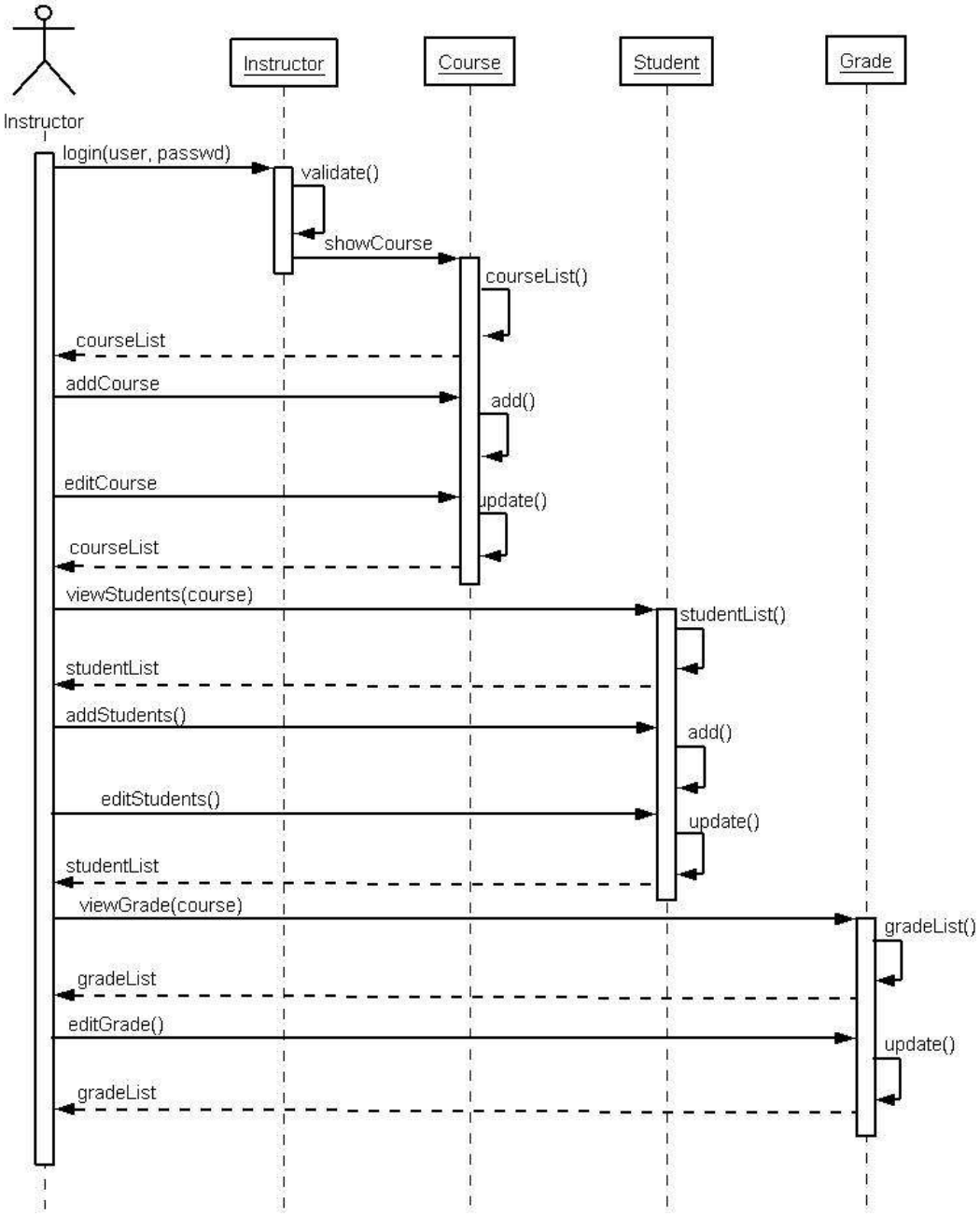
High Level Design

Administrator

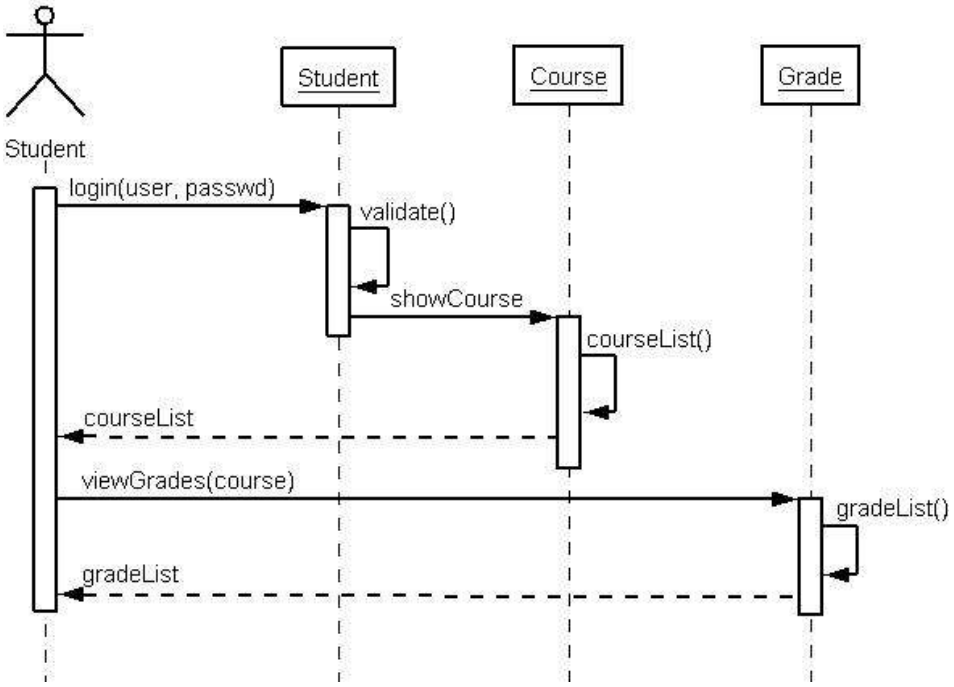


The asterisks' mean that there are multiple accesses to the Student objects and Instructor objects.

Instructor

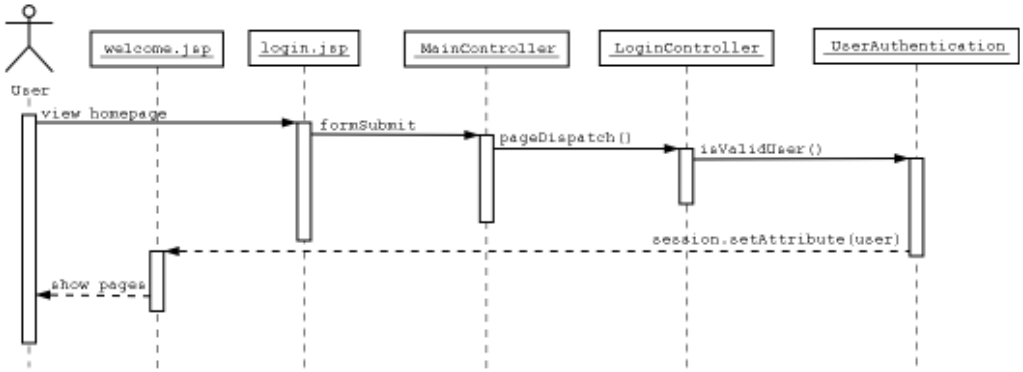


Student

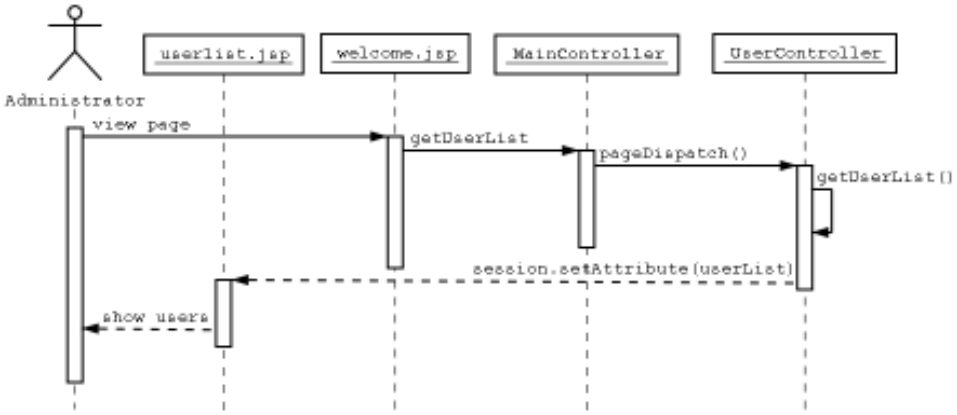


Functional Design

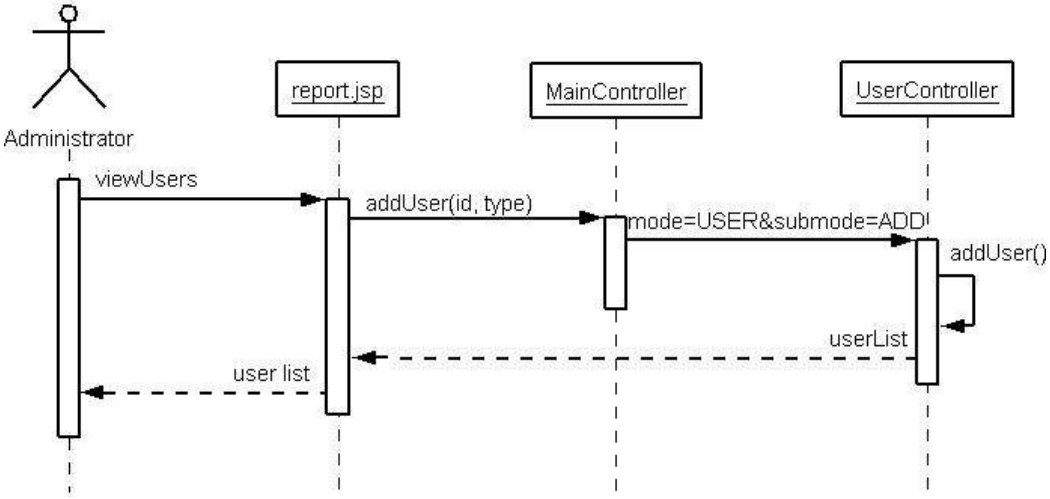
User Login (All users follow the same process)



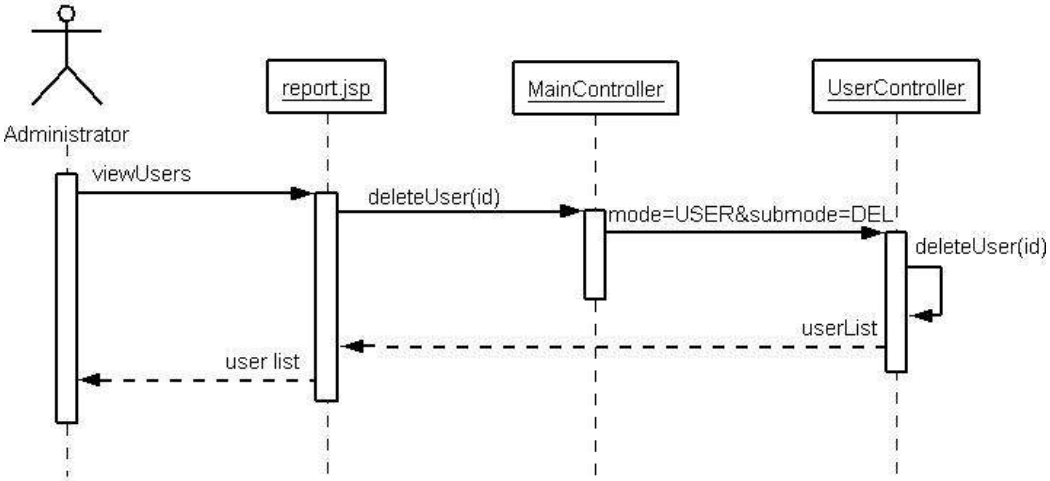
Administrator views user list



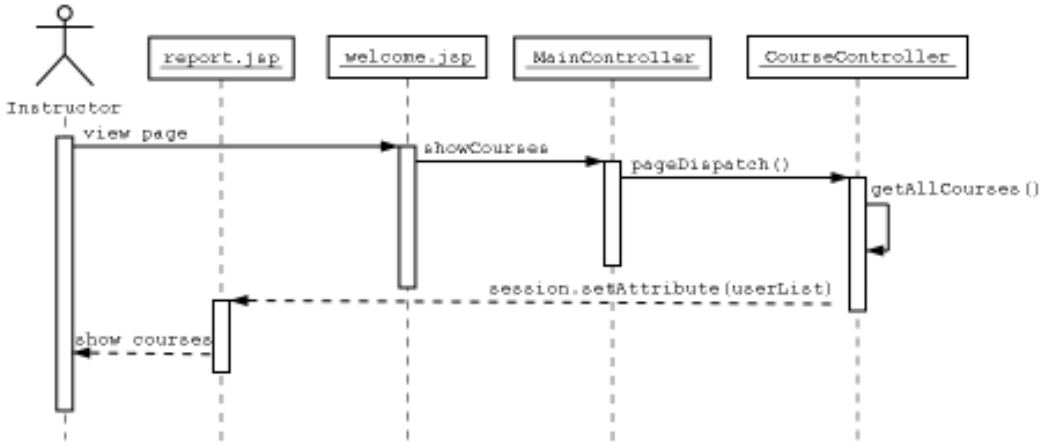
Add a user



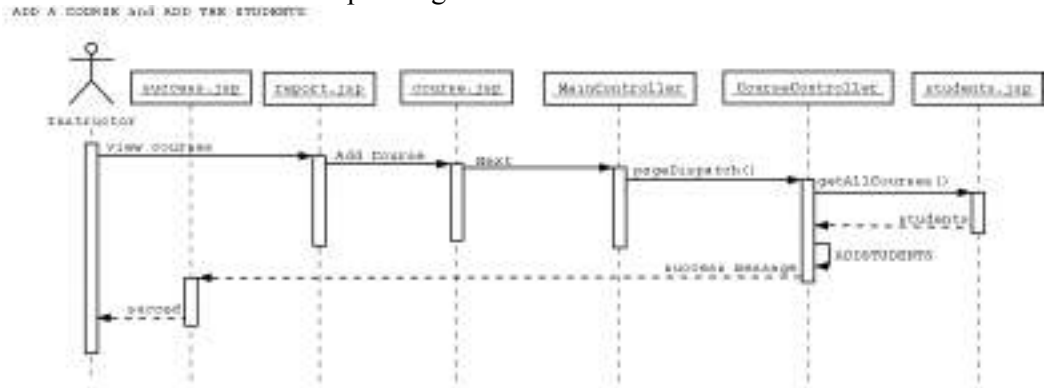
Delete a user



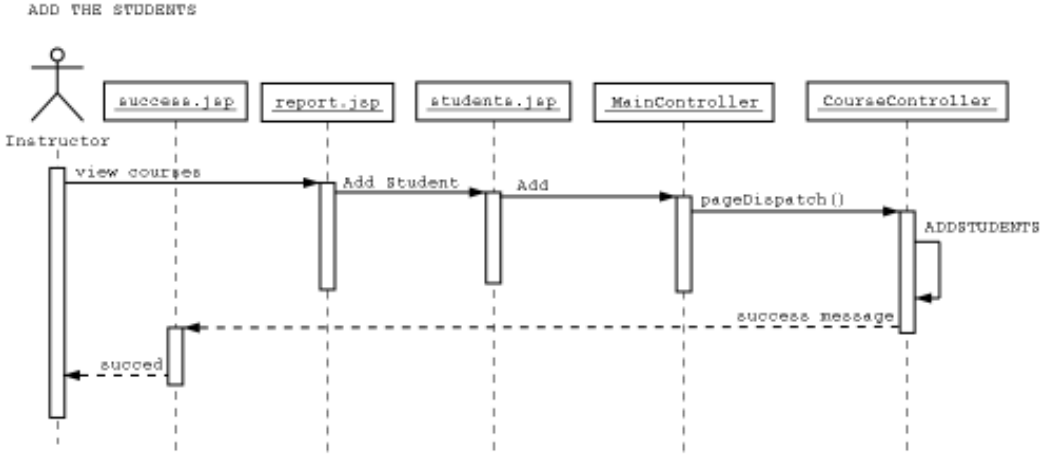
Instructor view courses



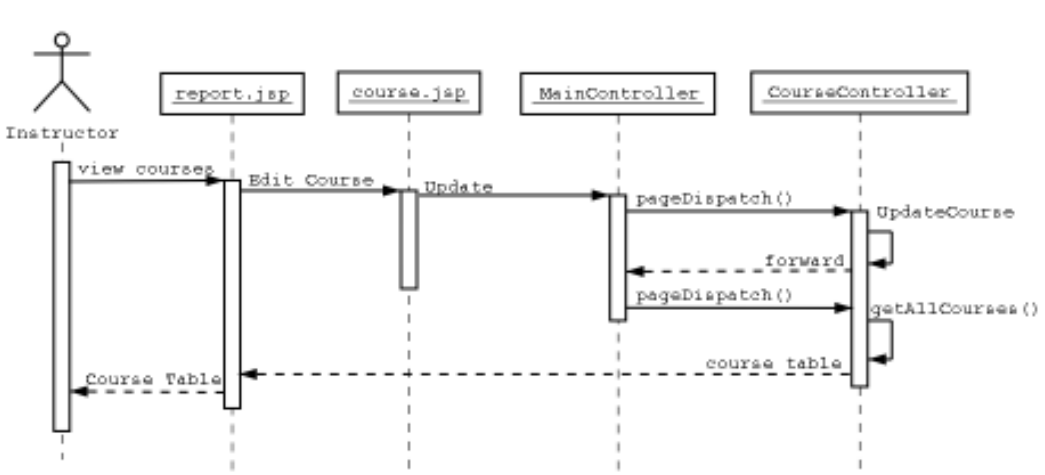
Add a course with the corresponding students



Add one student

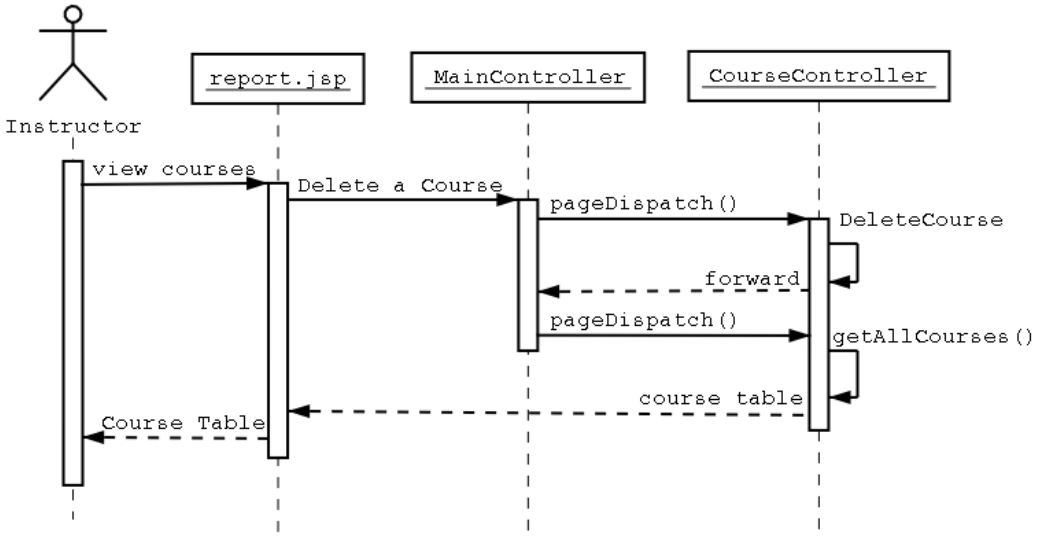


Update the course information



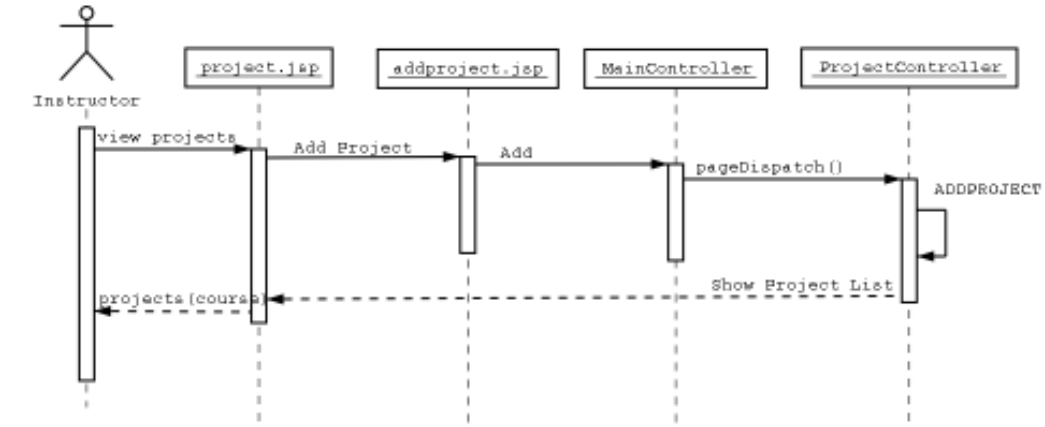
Delete a course

DELETE A COURSE



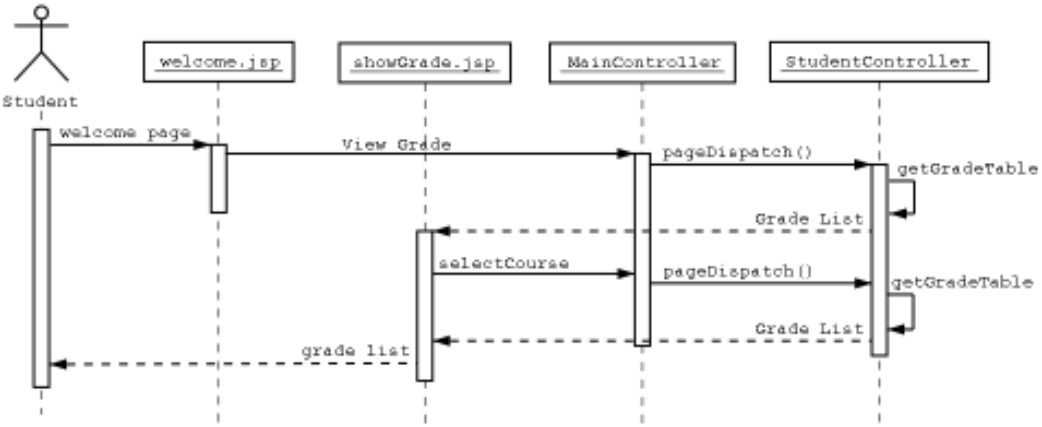
Add a project

ADD A PROJECT



Students view grades

VIEW GRADES

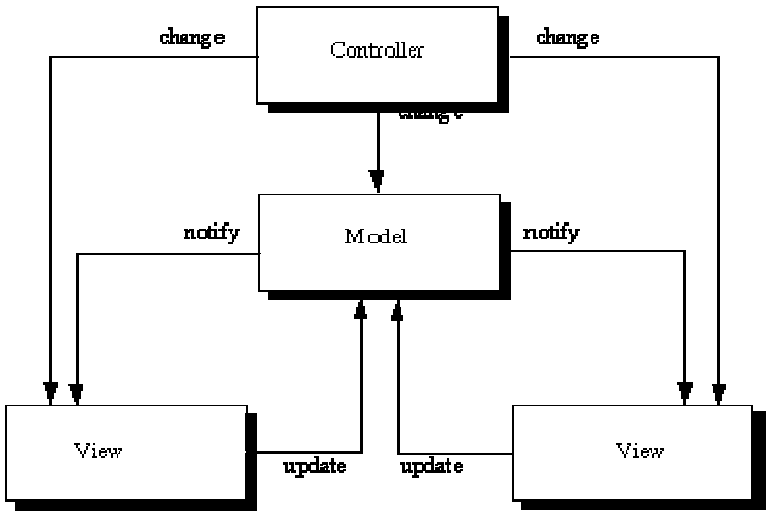


4. Software Architecture:

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.



5. Environment setup

System Requirements:

1. An operating system as Windows or Linux.
2. Java SE Development Kit (JDK) 1.5.0 or later installed [4].

3. Tomcat 5.5.20 or later [5].
4. MySQL 5.0 or later [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 '`grading`' and password as '`grading2006`' using the following command
   ```
   mysql > grant all on gradingsystem to grading
   identified by password grading2006;
   ```
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 'gradingsystem.war' from your hard disk and click on Deploy.
8. To access the software type
   http://<server name or IP address >:<port> /gradingsystem
   We type gradingsystem 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/gradingsystem/

6. Database Scripts

```
DROP TABLE IF EXISTS `usrcourseproj`;
DROP TABLE IF EXISTS `user_course`;
DROP TABLE IF EXISTS `course_project`;
DROP TABLE IF EXISTS `grade`;
DROP TABLE IF EXISTS `project`;
DROP TABLE IF EXISTS `course`;
DROP TABLE IF EXISTS `users`;

CREATE TABLE `users` (
  `user_id` varchar(20) NOT NULL default '',
  `user_passwd` varchar(20) default NULL,
  `user_type` varchar(20) default NULL,
  PRIMARY KEY  (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

LOCK TABLES `users` WRITE;
INSERT INTO `users` VALUES ('admin','admin','administrator');
UNLOCK TABLES;

CREATE TABLE `course` (
  `course_id` int(11) NOT NULL auto_increment,
```

```
  `course_seq_num` varchar(8) default NULL,
  `course_name` varchar(50) default NULL,
  `course_year` year(4) default NULL,
  `course_sem` varchar(10) default NULL,
  `course_sec` int(11) default NULL,
  `instructor` varchar(20),
  `course_no_of_stud` int(11) default NULL,
  PRIMARY KEY  (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `project` (
  `project_id` int(11) NOT NULL auto_increment,
  `project_name` varchar(50) default NULL,
  `project_max_marks` int(11) default NULL,
  `course_id` int(11) NOT NULL default '0',
  `project_due_date` timestamp NOT NULL default CURRENT_TIMESTAMP
on update CURRENT_TIMESTAMP,
  PRIMARY KEY  (`project_id`),
  CONSTRAINT `user_course_ibfk_12` FOREIGN KEY (`course_id`)
REFERENCES `course` (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `usrcourseproj` (
  `proj_id` int(11) NOT NULL auto_increment,
  `user_id` varchar(20) NOT NULL default '',
  `course_id` int(11) NOT NULL default '0',
  `project_id` int(11) NOT NULL default '0',
  `project_filename` varchar(100),
  `project_file` MEDIUMBLOB,
  `project_sub_date` timestamp default CURRENT_TIMESTAMP,
  `grade` varchar(5) default NULL,
  PRIMARY KEY  (`proj_id`),
  KEY `course_id` (`course_id`),
  KEY `project_id` (`project_id`),
  CONSTRAINT `usrcourseproj_ibfk_1` FOREIGN KEY (`user_id`)
REFERENCES `users` (`user_id`),
  CONSTRAINT `usrcourseproj_ibfk_2` FOREIGN KEY (`course_id`)
REFERENCES `course` (`course_id`),
  CONSTRAINT `usrcourseproj_ibfk_3` FOREIGN KEY (`project_id`)
REFERENCES `project` (`project_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


CREATE TABLE `course_project` (
  `course_id` int(11) NOT NULL default '0',
  `project_id` int(11) NOT NULL default '0',
  PRIMARY KEY  (`project_id`,`course_id`),
  KEY `course_id` (`course_id`),
  CONSTRAINT `course_project_ibfk_1` FOREIGN KEY (`project_id`)
REFERENCES `project` (`project_id`),
  CONSTRAINT `course_project_ibfk_2` FOREIGN KEY (`course_id`)
REFERENCES `course` (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `grade` (
  `grade_id` int(11) NOT NULL auto_increment,
  `grade_type` varchar(15) default NULL,
  `grade_value` char(3) default NULL,
```

```
  PRIMARY KEY  (`grade_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

LOCK TABLES `grade` WRITE;
INSERT INTO `grade` VALUES (1,'project','100');
UNLOCK TABLES;


CREATE TABLE `user_course` (
  `user_id` varchar(20) NOT NULL default '',
  `course_id` int(11) NOT NULL default '0',
  PRIMARY KEY  (`user_id`,`course_id`),
  KEY `course_id` (`course_id`),
  CONSTRAINT `user_course_ibfk_21` FOREIGN KEY (`user_id`)
REFERENCES `users` (`user_id`),
  CONSTRAINT `user_course_ibfk_22` FOREIGN KEY (`course_id`)
REFERENCES `course` (`course_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

7. Update Log

JavaBeans: which are used to get information from database, update, delete records. Because the system is changed to multiple-used, a class must be assigned an instructor and the course number. (Change the SQL statements)

`Course.java (WEB-INF/bean/)`
`User.java (WEB-INF/bean/)`

Main controller of the framework, to add the path for administrator

`MainController.java (WEB-INF/)`

JSPs

`menu.jsp` (Add the portion for administrator, /jsp/)
`report.jsp` (Change for the user list, /jsp/)

Administrator Mudule

`UserController.java (WEB-INF/user/)`

Configuration files:

`web.xml`: change the value of '`basedir`' to the corresponding path of gradingsystem application.

8. 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