

**CMPT 497 CAPSTONE PROJECT  
COURSE MANAGEMENT SYSTEM**

by

Kamchon Chio

A REPORT SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE SFU-ZU DUAL DEGREE OF  
BACHELOR OF SCIENCE  
in the School of Computing Science  
Simon Fraser University  
and  
the College of Computer Science and Technology  
Zhejiang University

© Kamchon Chio 2010  
SIMON FRASER UNIVERSITY AND ZHEJIANG UNIVERSITY  
Spring 2010

All rights reserved. This work may not be  
reproduced in whole or in part, by photocopy  
or other means, without the permission of the author.

## APPROVAL

**Name:** Kamchon Chio  
**Degree:** Bachelor of Science  
**Title of Report:** CMPT 497 Capstone Project Course Management System

**Examining Committee:** None  
Chair

---

Dr. Greg Baker, Supervisor

---

Dr. Ramesh Krishnamurti, Supervisor

---

Dr. Qianping Gu, SFU Examiner

**Date Approved:** \_\_\_\_\_

# Abstract

This project develops a course management web system that aims to replace the current systems used by the School of Computing Science at Simon Fraser University (SFUCS). SFUCS has several disparate systems for course management. Those generally used by instructors are: GradeBook, Assignment Submission Web Service and WebCT. These systems lack some critical functions. In addition a user has to manually (or using other software) transfer data from one system to another for the data sharing among the systems. The new course management system developed in this project provides a central point in terms of course management which includes four components: Grades component, Marking component, Submission component and Group Management component. The system also implements new functionalities and stringent validation logic within each system component. All the functionalities are developed with intuitive and effective web user interface. My responsibility in this project is to develop the Grades component.

# Contents

<b>Approval</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Contents</b>	<b>iv</b>
<b>List of Figures</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Old system . . . . .	1
1.1.1 GradeBook . . . . .	1
1.1.2 Assignment submission web service . . . . .	2
1.1.3 WebCT . . . . .	2
1.1.4 Marking service . . . . .	2
1.2 New system . . . . .	3
<b>2 Course Management System Components</b>	<b>4</b>
2.1 Grades component . . . . .	4
2.2 Marking component . . . . .	5
2.3 Submission component . . . . .	6
2.4 Group Management component . . . . .	7
<b>3 Project Technologies and Management</b>	<b>8</b>
3.1 Technologies . . . . .	8
3.1.1 Django . . . . .	9
3.1.2 HTML5 . . . . .	9

3.1.3	JQuery . . . . .	10
3.2	Project management . . . . .	10
<b>4</b>	<b>System Design</b>	<b>11</b>
4.1	MVC paradigm . . . . .	11
4.2	Middleware . . . . .	12
4.3	Logging service . . . . .	13
<b>5</b>	<b>Overview of the Grades Component</b>	<b>14</b>
5.1	The data models . . . . .	14
5.1.1	Activity type model . . . . .	15
5.1.2	Grade type model . . . . .	17
5.2	Core functionalities at instructor/TA side . . . . .	18
5.2.1	Course view . . . . .	18
5.2.2	Activity view . . . . .	19
5.2.3	Activity group view . . . . .	21
5.2.4	Adding/editing an activity . . . . .	22
5.2.5	Adding/Editing a calculated numeric activity . . . . .	23
5.2.6	The formula parser . . . . .	23
5.2.7	Reordering the activity position . . . . .	25
5.2.8	Calculating the numeric grade . . . . .	26
5.2.9	Marking student . . . . .	27
5.3	Core functionalities in student side . . . . .	27
5.3.1	Course view . . . . .	28
<b>6</b>	<b>Overview of Marking Component</b>	<b>30</b>
6.1	Core functionalities at instructor/TA side . . . . .	30
6.1.1	Configuring marking component . . . . .	30
6.1.2	Marking student . . . . .	31
6.1.3	Examining marking summary . . . . .	32
6.2	Core functionalities at student side . . . . .	33
6.2.1	Examining marking summary . . . . .	33

<b>7</b>	<b>Overview of the Submission Component</b>	<b>34</b>
7.1	Core functionalities at instructor/TA side . . . . .	34
7.1.1	Configuring submission component . . . . .	34
7.1.2	Examining student's submission . . . . .	34
7.2	Core functionalities at student side . . . . .	36
7.2.1	Submitting activity components . . . . .	36
<b>8</b>	<b>Overview of the Group Management Component</b>	<b>38</b>
8.1	Core functionalities at instructor/TA side . . . . .	38
8.1.1	Managing student group . . . . .	38
8.2	Core functionalities at student side . . . . .	41
8.2.1	Managing group . . . . .	41
<b>9</b>	<b>Conclusion and Future Improvement</b>	<b>42</b>
	<b>Bibliography</b>	<b>44</b>

# List of Figures

4.1	Code structure . . . . .	11
4.2	System middleware . . . . .	12
5.1	Activity models . . . . .	15
5.2	Grade models . . . . .	17
5.3	Instructor's landing view in a course . . . . .	18
5.4	Instructor/TA's view in a course activity . . . . .	19
5.5	News feeds . . . . .	20
5.6	Instructor/TA's group view in a group course activity . . . . .	21
5.7	Adding/editing activity . . . . .	22
5.8	Adding/editing calculated numeric activity . . . . .	23
5.9	Instructor/TA's view in a calculated numeric activity . . . . .	26
5.10	Marking student . . . . .	27
5.11	Student's landing view in a course . . . . .	28
5.12	Student's course activity view . . . . .	29
6.1	Instruct/TA' view on configuring marking components . . . . .	30
6.2	Instruct/TA' view on marking components . . . . .	31
6.3	Instruct/TA' view on marking summary . . . . .	32
7.1	Instruct/TA' view on configuring submission components . . . . .	35
7.2	Instruct/TA' view on examining student's submission . . . . .	35
7.3	Student's view on examining activity submissions . . . . .	36
7.4	Student's view on submitting activity components . . . . .	37
8.1	Instruct/TA' view on managing student group . . . . .	39

8.2	Instruct/TA' view on creating group . . . . .	39
8.3	Student's view on managing group . . . . .	40
8.4	Student's view on creating group . . . . .	40



# Chapter 1

## Introduction

GradeBook, Assignment Submission Web Service and WebCT are the most prevalent course management services used within the School of Computing Science at Simon Fraser University (SFUCS). These services just provide limited functionalities within their own service domain. Users always find the systems hard to use and find the user interfaces not very user-friendly. A lot of work has to be done manually due to the lack of central management among these services. Moreover, some systems have been in existence for over a decade, and do not take advantage of modern software development technologies. In order to see more clearly why the old systems need to be replaced, I provide an overview of the old systems and their pitfalls.

The rest of the paper is structured as follow: In the rest of this chapter, I provide an overview of the old systems and their pitfalls. In Chapter 2, I describe the function domain of the new system. In Chapter 3, I describe the web development technologies and the project management. In Chapter 4, I briefly describe the system architecture. In Chapter 5, I give an overview of the system component that I work on. In Chapter 6, I give a conclusion of the new system and share my idea of future improvement.

### 1.1 Old system

#### 1.1.1 GradeBook

Gradebook is widely used within SFUCS. Course registration data are retrieved from Student Information System (SIS) and reside in GradeBook's database system. This system

provides basic functions to manage course activities and student grades. However, limited functionalities are provided for an instructor to manage student grades. Student grade can only be in numeric format and there is no way to distinguish the grade status (e.g., academic dishonesty). There is no communication bridge between GradeBook and Assignment Submission Web Service (see Section 1.1.2) due to which an instructor has to manually relate information between these systems. Functions such as import/export as CSV file format<sup>1</sup>, news feed, group management and calculation of numeric grade are not provided. Thus, the system does not effectively facilitate an instructor to manage a course.

### 1.1.2 Assignment submission web service

Assignment submission web service is widely used within SFUCS. Assignment submission data are retrieved from GradeBook. However, as mentioned in the previous section, this service is loosely coupled with GradeBook and users find it troublesome to use. Moreover, every submission can have only one submission component for which the submission type is limited to a compressed file. This restricts the submission type that can be applied to an assignment, for example, a coding assignment requires only code files, a web application requires only a web link, etc. There is no way to manage the ownership of marking a submission and manual work is required to maintain the information.

### 1.1.3 WebCT

WebCT is widely used within all SFU faculties. It tries to deliver every possible function that is needed for course management. Instructor finds it hard to use because of the wide range of functionalities and some misleading operations. The interface is not user friendly. Thus, instructors in SFUCS are not willing to use WebCT. Instead, they use GradeBook for simplicity.

### 1.1.4 Marking service

There is a marking service that was written by a SFUCS faculty member, Greg Baker. This system gets the course activity data from GradeBook and provides marking functionalities based on the activity information. However, the two systems are still loosely coupled and

---

<sup>1</sup>CSV is a comma-separated values file used for the digital storage of data structured in a table of lists form[8]

users have to log into two different systems in order to relate the information with each other. Moreover, there is no communication bridge between marking service and assignment submission web service. Thus, the marking service is not popular and has only a few users.

## 1.2 New system

In this project, a new system is implemented to replace the old systems to provide more convenient and efficient course management. The new system integrates the GradeBook, Assignment Submission Web Service and Marking Service into a central course management system with single user authentication and authorization point, common databases and enhanced functionalities. The system is called Course Management System (CMS). We divide the CMS into four major components: Grades, Marking, Submission and Group Management. These components, excluding Group Management, are related to the original systems by the names. The Group Management component which is not implemented in the old system provides student group management functionalities. The rest of this report is structured as follows: In Chapter 2, I describe the function domain of the new system. In Chapter 3, I describe the web development technologies and the project management. In Chapter 4, I briefly describe the system architecture. In Chapter 5, I give an overview of the system component that I work on. In Chapter 6, I give an overview of Marking component. In Chapter 7, I give an overview of Submission component. In Chapter 8, I give an overview of Group Management component. In Chapter 9, I give a conclusion of the new system and share my idea of future improvement.

## Chapter 2

# Course Management System Components

Each component in the CMS is targeted to realize the functionalities of the corresponding old system and additional new functionalities. The remaining sections in this chapter list the function scope of each component in details.

### 2.1 Grades component

- Course activity has four types: numeric activity, letter activity, calculated numeric activity and calculated letter activity. Activity has a due date/time. Letter activity has any of the standard SFU grade values. Calculated numeric activity is based on a user specified formula to calculate a numeric grade according to other course activities. Calculated letter activity is based on the grade cutoff (e.g., A+ when the final numeric grade is over 90) that are specified by the users to generate a letter grade for a numeric activity.
- Activity has released/unreleased status such that the activity can be configured to be unreleased when students grades are not releasable (e.g., only a part of the students have an assignment grade).
- Activity can be visible/invisible to student.

- Activity has grade summary statistics: average, minimum, maximum, median, standard deviation and a histogram showing grades distribution.
- Activity can be reordered.
- Instructor/TA view: full list of student information which can be searched and ordered.
- Student view: a summary of the course information containing course metadata, course activities, activity grades and activity statistics.
- Seamless integration with the Marking and Submission component such that instructor/TA can access the marking details and assignment submissions through the student list.
- Student grades can be exported to or imported from the CSV format.
- Letter grades should be exportable to the format used in SIS.
- News item (or news feed) is created for a new mark.
- Course setup can be copied from semester to semester, including marking components (see Section 2.2) and submission components (see Section 2.3).

## 2.2 Marking component

- Activity in the Grades component can be marked.
- Every markable assignment can have multiple marking component (e.g., validation of code, readability of code, execution time of code, etc).
- Each marking component has a maximum mark (e.g., out of 5), title (e.g., validation of code) and description (e.g., the submitted code passes all the test cases).
- Marking components can have "common problems" associated with them. Thus, Instructor/TA can identify "common problems" with a comment and a mark penalty (e.g., one test failure has mark penalty -1). When marking, these can be selected for insertion into a marking component.
- When marking, instructor/TA can enter a mark and comment for a marking component.

- It should be able to attach a file to a marked assignment (e.g., to return the corrected/commented code to the student).
- Marking should include late penalty.
- Course setup can be copied from semester to semester.

## 2.3 Submission component

- Activity in the Grades component can configure submission.
- Submission can have multiple submission components (e.g., archive submission, URL submission or pdf submission, etc).
- Submission component type can be: text file, ZIP/TGZ/RAR archive files, URL, pdf, code file.
- Every submission component has a size limit (e.g., maximum size of text file is 100kb).
- Submission component should be validated reasonably (e.g., submitted URL should exist, code file has extension ".java"). Some submission component validation can be overridden by the student (e.g., student still want to submit an uncompileable code for marking purpose).
- Multiple submissions of an assignment are allowed, and displays the most recent one.
- Student can submit based on the submission configuration.
- Submission reflects the due date and is flagged with extent of lateness.
- When marking a submission component, instructor/TA can take ownership. This is used to ensure others don't mark it while one person is already working on it. It should be possible to override ownership if necessary (e.g., others can take ownership away if they have a reason).
- Submission component can be flagged new/in-progress/done. Submission component starts with new, then in-progress when it is taken, and done after it is marked or explicitly set.

- Automate testing of code file submission.
- News item (or news feed) is created for new submission.

## 2.4 Group Management component

- Students can form a group if the course activity is a group activity.
- Students can create a group, select group members and choose a group for approval. All members of a group must approve the group membership if the group is created by students. Instructor/TA created group does not need to be approved.
- Instructor/TA can assign students to groups.
- Marks can be associated with a group if it is group activity. Instructor/TA can still mark individually.
- Submissions are associated with a group if it is group activity.

## Chapter 3

# Project Technologies and Management

The goal of the project is to develop a functional system in one academic semester. This limited time frame, together with the team's insufficient web development experience, narrows down the technologies we can choose from.

### 3.1 Technologies

We use Python[3]+Django[1] as the server-side technologies. Django is a rapid application web development framework that is based on the Python language. It fulfills the stringent timeline requirement of the project. Django provides an ease-of-use framework to handle the complex design nature of database-driven web application with the least programming effort. Section 3.1.1 briefly describes the Django framework. We use HTML[10]+CSS[7]+Javascript[12] as the client-side technologies. We use JQuery and its plugins as the Javascript library to build the rich internet application interface. Using the JQuery library also help us to solve compatibility issues which occur between different versions of a browser as well as among different vendors of browsers. We also integrate the latest HTML5[11] technology (canvas html tag) into the client-side.



### 3.1.1 Django

Django is a Python web framework that encourages rapid development and clean design[1]. It emphasizes automating as much as possible and adheres to the Don't Repeat Yourself (DRY) principle. Developers focus on the implementation of business logic rather than implementing the web framework. The following lists some the core functions in Django which are most valuable to our project.

- The core Django framework offers an object-relational mapper which mediates between data model (defined as Python class) and the relational database[9]. Developers usually do not need to deal with database queries. The Django database API is competent enough to manipulate the database in most cases.
- An easy-to-configure request processing system which consists of a regular-expression based URL dispatcher.
- A message framework that facilitates message delivery
- Automatic admin interface for people to manage the data.
- A template system to display server processed data into client-side.
- A light-weight, standalone web server for deployment and testing.
- Support for middleware classes which can intervene at various stages of request processing and carry out custom functions, for example, authentication and authorization middleware, session middleware.
- An interface to Python's built-in unit test framework.

### 3.1.2 HTML5

HTML5 is the next major version of HTML specification[11]. It contains new elements to facilitate the creation of rich internet application. For example, the canvas element can be used to generate graphs, audio or video elements to embed media, scripting API to implement drag and drop behaviour, etc. It reduces the need for proprietary rich internet technologies such as Flash, Silverlight and JavaFX. However, the specification is still under development and only a few of the HTML5 elements are supported by major browsers.

At the moment, the only HTML5 element implemented in the project is canvas, which is used to show the course activity histogram to users. This element is well supported by the latest version of most major browsers (except IE): Firefox, Chrome, Opera and Safari. Fortunately, there is a google Javascript library that enables the canvas support in IE.

### 3.1.3 JQuery

JQuery is a featured library that simplifies HTML document traversal, event handling, animation and Ajax interaction for rapid web application[5]. We use JQuery for basic HTML manipulation and JQuery plugins for advanced features. For example, we use the DataTable plugin[4] for advanced table manipulations such as search, sort, pagination and filter. We also use the JQuery UI[6] to build the highly interactive web application.

## 3.2 Project management

The team uses Scrum[13] as the project management tool. Greg Baker, the professor in charge of the project, is the product owner and Scrum master. The project consists of one orientation sprint (from Jan 5 to Jan 11), two major development sprints (one from Jan 12 to Mar 1, and the other from Mar 2 to Apr 5). After that there is a two week sprint to test and validate the system so that it can be tested in the summer semester on a real course.

In the orientation sprint, the team learns Python programming and how to use the Django framework to develop web applications. During this phase, the team developed an advisor portal which keeps advising notes on students. The system allows the notes to be shared among advisors.

In the two major development sprints, the team focuses on the system user requirements and implements the system accordingly. The product owner (Greg Baker) cooperates with some domain experts, who are the professors in SFUCS using the current course management systems, to elicit the product backlog and utilize the Trac project management tool to keep track of the backlog. I, as part of the development team, pick up tasks from the product backlog and implement them. I am in charge of the Grades component of the course management system. Greg Baker also plays the roles of code reviewer and technical consultant and code developer.

In the final two week sprint, the team focuses on the validation and testing of the system and leave the remaining user requirements in the backlog.

# Chapter 4

# System Design

## 4.1 MVC paradigm

Django provides well-defined architectures to design a web system. We adapt the MVC design paradigm implemented by Django. Thus, the code base of CMS shows a clear separation between model, view and controller, as shown in Figure 4.1.

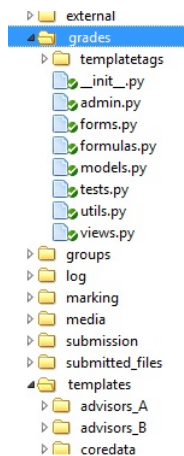


Figure 4.1: Code structure

The models.py file under the grades folder (the Grades component) defines the models of Grades component. These models define the essential fields and behaviors of the data the component is storing. The model is implemented using an object-relational mapper, thus

every model is mapped to a database table. The controller which is called view in Django (hereinafter I refer to controller to view in accordance with adapt the Django conventions) is defined in the views.py file. These views are mapped by the URL dispatcher and are called when the user requests the corresponding URL, for example:

```
url(r'^' + COURSE_SLUG + '/$', 'grades.views.course_info')
```

The above code snippet defines a URL mapping to a view function. The first part is the regular expression of the URL (`COURSE_SLUG` is a regular expression that represents a unique string that can identify a course, e.g., 1101-cmpt-165-d100<sup>1</sup>). The second part is the mapped view function that is defined in the views.py file under grades folder.

The view (of MVC) is implemented using Django's template system. As can be seen in Figure 4.2, the templates is placed in the templates folder.

## 4.2 Middleware

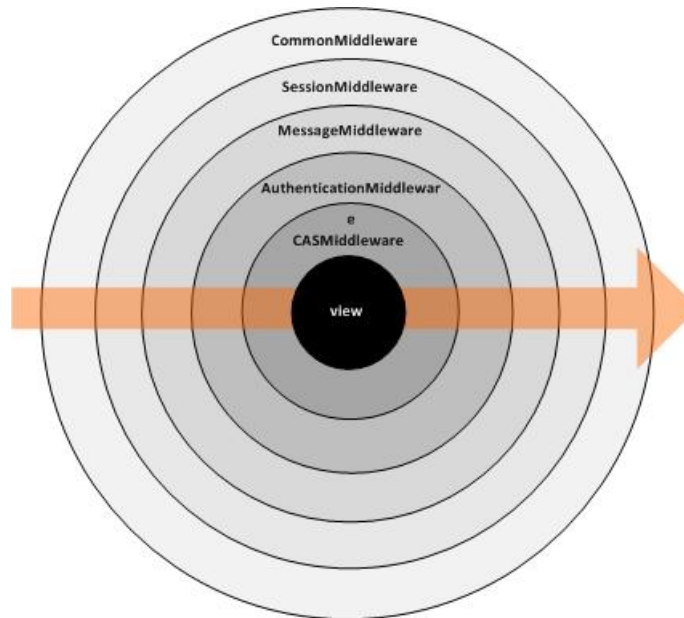


Figure 4.2: System middleware

<sup>1</sup>the course slug conforms to semester and course code defined by SFU

Middleware is a framework of hooks into Django's request/response processing. Each middleware component is responsible for doing some specific function. In Figure 4.2, it shows the five middlewares used by the CMS system. These middlewares are applied to the request/response processing sequentially.

The `CommonMiddleware` add convenience to web development. It can append slash "/" to the end of the request URL and the prepend WWW to the front of the URL. Both of these options are meant to normalize URLs. The philosophy is that each URL should exist in one, and only one, place. Technically a URL `foo.com/bar` is distinct from `foo.com/bar/` - a search-engine indexer would treat them as separate URLs - so it is best practice to normalize URLs.

The `SessionMiddleware` activates the session management of the system. The session framework lets you store and retrieve arbitrary data on a per-site-visitor basis. It stores data on the server side and abstracts the sending and receiving of cookies. Cookies contain a session Id and not the data itself.

The `MessageMiddleware` provides full support for cookie- and session-based messaging. It allows the system to temporarily store a message in one request and retrieve them for display in a subsequent request (usually the next one). Every message is tagged with a level to determine its severity and priority, e.g., info, success, warning or error.

The `AuthenticationMiddleware` handles user accounts, groups, permissions and cookie-based user sessions. It maintains the user information in the backend and provide an authentication framework. It also hooks the user information into the request object for easy access.

The `CASMiddleware` is developed by our team to provide integration with the SFU CAS authentication system. This middleware utilizes the `AuthenticationMiddleware` to handle user accounts and permissions but uses the login service provided by CAS system to authenticate the user.

### 4.3 Logging service

We have implemented a logging service to log user activity (e.g., instructor A creates Assignment1 to cmpt165). We have defined a data model to store the log information into the database and provide a simple user interface for the system administrator to retrieve the log information.

## Chapter 5

# Overview of the Grades Component

This chapter provides an overview of the Grades component and some of the core technical implementations. Since I am only in charge of the grade component, I only elaborate the technical implementation of Grades component and I leave other components out.

The Grades component mainly deals with the course activity management and student grade manipulation. We have defined four activity types and two grade types. The activity types are numeric activity, letter activity, calculated numeric activity and calculated letter activity. The grade types are numeric grade and letter grade. These four activity types and two grade types are mapped to the database model directly. Most of the functionalities in the Grades component are built for these six models.

### 5.1 The data models

The system has defined several data model types that are used across all the components and mapped to the database directly. These types include but are not limited to: `Person`, `Semester`, `CourseOffering` (e.g., CMPT165 is offered in 2010 spring semester) and `Member` (e.g., student A is a student in the CMPT165 `CourseOffering`). These data models hold general course offering and student information that can be retrieved from other SFU systems such as SIS for which the course offering information is maintained. Thus, the manipulation of these data does not belong to the function scope of the CMS.

### 5.1.1 Activity type model

We define a base abstract activity type `Activity` for holding general activity information:

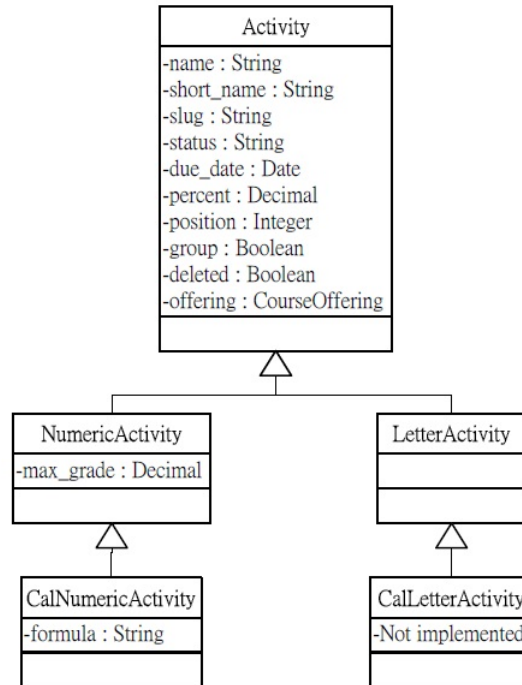


Figure 5.1: Activity models

The following fields are defined in the abstract activity type `Activity`:

- **name**: the activity name that is displayed. This field is unique within a course offering.
- **short\_name**: the activity short name for activity manipulation, e.g., when specify the activity reference in the calculated numeric activity formula (see Section 5.2.6). This field is unique within a course offering.
- **slug**: Auto generated field when the data model is saved to database. This is a unique identifier within a course offering that is generated based on the `short_name`. It is mainly used in URL manipulation.
- **status**: activity status which can be released/unreleased/invisible. Instructor/TA uses this attribute to control the display of activity information for a student and

to restrict student action on an activity. For example, student cannot submit any assignment when the activity is released; student cannot view his/her grades when the activity is unreleased; a student cannot see the activity when it is invisible, etc.

- **due\_date**: activity due date. Submission beyond the due date may have a penalty applied.
- **percent**: how much this activity contributes towards the final grade.
- **position**: keep track of activity ordering in a course offering. Instructor/TA can reorder an activity.
- **group**: specify whether this is a group activity. This will affect the layout and work flow when the corresponding activity is manipulated.
- **deleted**: specify whether the activity is deleted.
- **offering**: the course offering reference, see Section 5.1.

Numeric activity type `NumericActivity` is inherited from `Activity` with an additional field:

- **max\_grade**: the maximum numeric grade in this activity.

Letter activity type `LetterActivity` is inherited from `Activity` without any additional field:

- **formula**: the formula for calculating the numeric grade. See Section 5.2.6.

Calculated letter type has yet to be implemented.



### 5.1.2 Grade type model

We define two Grade types as shown in Figure 5.2:

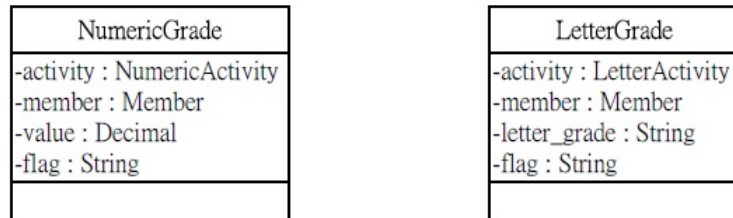


Figure 5.2: Grade models

Numeric grade type `NumericGrade` holds numeric grade information for a student in a course numeric activity:

- **activity**: the numeric activity reference.
- **member**: the course member reference, see Section 5.1.
- **value**: the numeric grade value.
- **flag**: grade status which can be `graded/no grade/calculated/dishonesty/excuse`. This is useful to specify additional information for a student grade, e.g., student who flouts the academic policies is flagged `dishonesty`; grade generated from the calculated numeric activity is flagged `calculated` (see Section 5.2.8).

Letter grade type `LetterGrade` holds the letter grade information for a student in a course letter activity.

- **activity**: the numeric activity reference.
- **member**: the course member reference, see Section 5.1.
- **letter\_grade**: the letter grade which can be one of the standard SFU values.
- **flag**: grade status which can be `graded/no grade/calculated/dishonesty/excuse`. This is useful to specify additional information for a student grade, for example, a student

who flouts the academic policies is flagged dishonesty; grade generated from the calculated letter activity is flagged calculated (not implemented).

## 5.2 Core functionalities at instructor/TA side

### 5.2.1 Course view

**CMPT 165 COURSE INFORMATION**

Course Number	CMPT 165
Section	D100
Semester	Spring 2010
Title	Intro Internet World Wide Web
Campus	Burnaby Campus
Instructor(s)	Gregory Baker (ggbaker@sfu.ca)
TA(s)	Doug Gradstudent (0grad@sfu.ca)
Number of Students	20

**Actions**

- [Display All Grades](#)
- [Manage Groups](#)
- [Message Class](#)
- [Copy Course Setup](#)

**Course Activities**

[Add Activity](#)

Order	Activity	Group?	Max Grade	Percentage	Status	Due Date
↓	<a href="#">Assignment 1</a> Info   Mark   Edit	group	25	10%	released	2010-03-17 00:00:00
↑ ↓	<a href="#">Assignment 2</a>	individual	25	10%	released	2010-03-31 00:00:00
↑ ↓	<a href="#">Midterm Exam</a>	individual	90	20%	unreleased	2010-04-21 00:00:00
↑ ↓	<a href="#">Project</a>	individual	—	—	unreleased	None
↑	<a href="#">Final Grade</a>	individual	100	—	released	None

Showing 1 to 5 of 5 entries First Previous 1 Next Last

Figure 5.3: Instructor’s landing view in a course

In Figure 5.3, Instructor/TA can see the course information and the full list of course activities, including all four activity types. The course information is retrieved from the CourseOffering data model, see Section 5.1. Each table row of the activities list contains the primary information of a course activity. All the information can be mapped to the Activity data model directly. The order column contains reorder button so that instructor/TA can reorder the activity accordingly, see Section 5.2.7 for reordering activities. Links are provided to add/view/mark/edit course activity, see Section 5.2.4 for adding/editing course activity. The search and pagination functionalities of the list are enabled by JQuery plugin

(dataTable) which is a client side operation. This minimizes unnecessary server requests and thus provides better user experience. On the top-right corner, there is an action box containing the list of actions the instructor/TA can perform for the course. These actions include an all-students-all-activities-grades view of the course, managing the course groups (bridge to Group Management component), creating course messages (e.g., if an instructor wants to deliver a message to the class) and copying the course setup from a previous course that is taught by the instructor.

### 5.2.2 Activity view

#### ASSIGNMENT 1 INFORMATION

Activity Type	Numeric Graded
Name	Assignment 1
Short name	A1
Status	released
Due date	2010-03-17 00:00:00
Percentage	10
Maximum grade	25

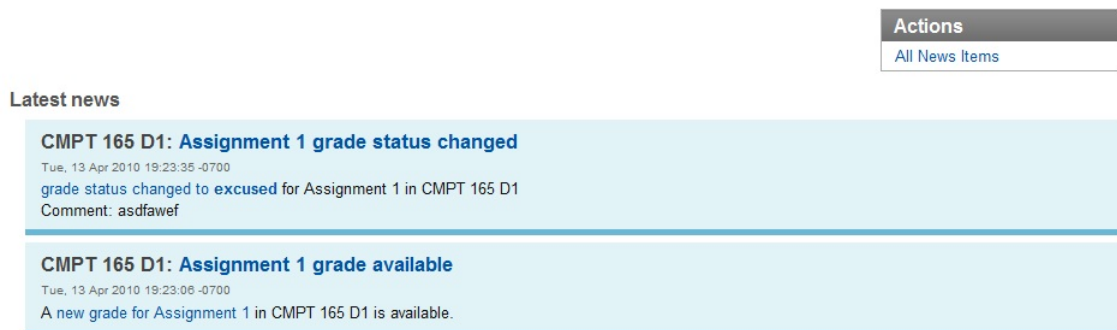
Actions
<a href="#">Edit Activity Details</a>
<a href="#">Display Statistics</a>
<a href="#">Configure Submission</a>
<a href="#">Configure Marking</a>
<a href="#">Configure Common Problems</a>
<a href="#">Mark all</a>
<a href="#">Mark all by Group</a>
<a href="#">View By Group</a>
<a href="#">Export all</a>

Show 50 entries							Search:
	Last name	First name	User ID	Stu #	Grade Status	Grade	
	Student	A	0aaa0	200000169	graded	22/25	
	Student	B	0aaa1	200000170	graded	18/25	
	Student	C	0aaa2	200000171	no grade		
	Student	D	0aaa3	200000172	no grade		
	Student	E	0aaa4	200000173	no grade		

Figure 5.4: Instructor/TA's view in a course activity

By clicking an assignment link in the activities table as shown in Figure 5.3, the system will redirect the user to a page showing detailed information on the course activity, as shown in Figure 5.4. Instructor/TA can examine all the activity information on the top-left corner of the page; user can also examine the full list of students with their primary information and activity grade information. The student information is retrieved from two data models: 1) Member (see Section 5.1) for 'Last name', 'First name', 'UserID' and 'Stu #'; 2) grade model (see Section 5.1.2) for 'Grade Status' and 'Grade'. Instructor/TA can mark the student by

clicking on the link (bridge to Marking component) in the 'Grade' column and examine the student's activity submission through the link (bridge to Submission component) in the last column. Instructor/TA can also change student's grade status (bridge to marking component) through the 'Grade Status' column. These changes are reflected as news items once the activity is released, and thus, the student is notified with the updated grade and grade status. Figure 5.5 shows the news feeds that are displayed in the student's landing page.

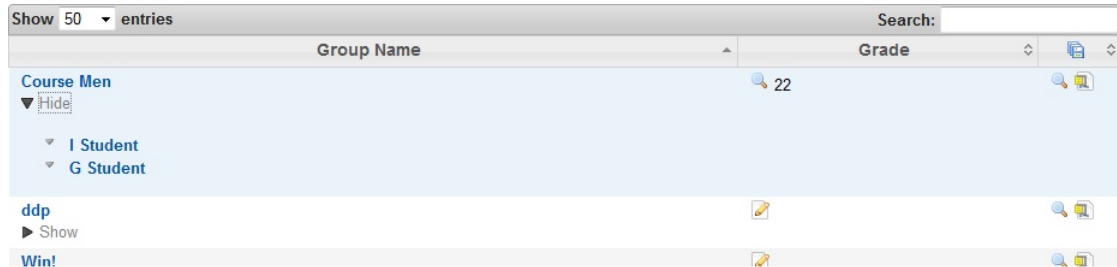


The screenshot shows a user interface for a student's landing page. In the top right corner, there is a dark grey box labeled "Actions" with a white button labeled "All News Items". Below this, on the left, is a section titled "Latest news". The news feed contains two items, each in a light blue box with a thin border. The first item has the heading "CMPT 165 D1: Assignment 1 grade status changed", a timestamp "Tue, 13 Apr 2010 19:23:35 -0700", and the text "grade status changed to **excused** for Assignment 1 in CMPT 165 D1" followed by a comment "Comment: asdfawef". The second item has the heading "CMPT 165 D1: Assignment 1 grade available", the same timestamp, and the text "A new grade for Assignment 1 in CMPT 165 D1 is available."

Figure 5.5: News feeds

On the top-right corner of Figure 5.4, there is an action box containing the list of actions the instructor/TA can perform for the activity. Instructor/TA can edit the activity (see Section 5.2.4), view the activity statistics (e.g., max grade, min grade, stddev, histogram, etc), configure the submission component (bridge to Submission component) and marking component (bridge to Marking component) and import/export the list of students' grades as CSV, see Section 5.2.9.

### 5.2.3 Activity group view



Group Name	Grade
Course Men	22
▼ Hide	
▼ I Student	
▼ G Student	
ddp	
► Show	
Win!	

Figure 5.6: Instructor/TA's group view in a group course activity

Instructor/TA can view the students as groups if it is a group activity, as shown in Figure 5.6. Instructor/TA can mark the activity as a group and examine the group's submission. The group information in this page is retrieved from the data model in Group Management component which is not discussed in this paper.

### 5.2.4 Adding/editing an activity

⚠ Please correct the error below

Numeric Graded Activity

Letter Graded Activity

Calculated Numeric Activity

**Activity Info**

★ Indicates required field

Name:★   
(name of the activity, e.g "Assignment 1" or "Midterm")

Short name:★  ⚠ This field is required.  
(short version of the name for column headings, e.g. "A1" or "MT")

Status:★ unreleased ▾  
(visibility of grades/activity to students)

Due date: Date:    
(Time format HH:MM:SS)

Percentage:   
(percent of final mark)

Group activity:★  Yes  
 No

Maximum grade:★   
(maximum grade for the activity)

Figure 5.7: Adding/editing activity

Forms are provided to add/edit a course activity. At the moment, three types of activity can be added/edited, namely, numeric activity, letter activity, calculated numeric activity. As can be seen in Figure 5.7, all form fields correspond to the activity data model with the required field indicated by a red asterisk. The Date picker in the “Date” field is implemented using JQuery UI. Form validations are also implemented with proper error messages shown.

In the Django backend, the form is implemented using Django Form[2]. Django Form provides handy form manipulations of common form behaviours. It provides data field conversion (e.g., transforming string to object, object to string), data field validations and form HTML generation.

Name:  (name of the activity, e.g. "Assignment 1" or "Midterm")

Short name:  (short version of the name for column headings, e.g. "A1" or "MT")

Status:  (visibility of grades/activity to students)

Percentage:  (percent of final mark)

Maximum grade:  (maximum grade of the calculated result)

Formula:

(parsed formula to calculate final numeric grade)

**Formula Tester**

▼ Hide the list of applicable numeric activities:

Name	Short Name	Max Grade	Percentage	Status
Assignment 1	A1	25	10%	released
Assignment 2	A2	25	10%	released
Midterm Exam	MT	90	20%	unreleased
Final Grade	Final	100	—	released

**Formula Tester**

★ Indicates required field

**Activity Info**

Give status and grade to the numeric activities to test formula

Name	Short Name	Percentage	Status	Grade
Assignment 1	A1	10%	released	<input type="text" value="/25"/>
Assignment 2	A2	10%	released	<input type="text" value="/25"/>
Midterm Exam	MT	20%	released	<input type="text" value="/90"/>
Final Grade	Final	—	released	<input type="text" value="/100"/>

**Specify Formula**

Formula:

(parsed formula to calculate final numeric grade)

**Result**

Figure 5.8: Adding/editing calculated numeric activity

### 5.2.5 Adding/Editing a calculated numeric activity

Figure 5.8 shows the form page of adding/editing a calculated numeric activity. When instructor/TA specifies the formula field, he/she can refer to the applicable activities in the drop down list and use the formula tester to test the integrity of the formula. In the formula tester, instructor/TA can see the full list of applicable numeric activities. He/she can input testing values for each of them and input the formula to evaluate the result. This formula tester helps instructor/TA to ensure the correctness of the formula so that they do not need to use the real student data to experiment with the result. The formula field is a plain text that can be evaluated by a formula parser (see Section 5.2.6) to produce the evaluated result.

### 5.2.6 The formula parser

The formula parser accepts name and short\_name field (see Section 5.1.1) of the activity model and exposes the value, percent and max\_grade field through the activity reference in the formula plain text (e.g.,  $[A1]$ ,  $[A1.percent]$ ,  $[A1.max\_grade]$ ). It accepts real number;

operators:  $+$ ,  $-$ ,  $*$  and  $/$ ; functions: *SUM*, *AVG*, *MAX*, *MIN*, *BEST*; and sign operator. For example, we can specify the following as the formula:  $[A1] + [A2] * 2$ .

The formula parser is implemented using python's *pyparsing* module<sup>1</sup>. *Pyparsing*'s class library provides a set of classes for building up a parser from individual expression elements, up to complex, variable-syntax expressions. For example, the *pyparsing* grammar of an IP address can be expressed as:

```
ipField = Word(nums, max = 3)
ipAddr = Combine(ipField + "." + ipField + "." + ipField + "." + ipField)
```

The *ipField* is a *Word* class denoting a 3-digit number field. The *ipAddr* is a *Combine* class denoting the combination of four *ipField(s)* intercepted with "."

Once the grammars of the formula are specified, the formula parser can parse the formula plain text into a well-defined data structure that can be evaluated easily. The following examples explain the data structure and how it is used by the formula parser to evaluate the result.

### Example 1

Formula:  $[A1]$

Data structure after parsing: ('col', set(['A1']), 'A1')

The first element indicates the type of the evaluation; 'col' means an activity reference. Five evaluation types have been defined: 1) 'col' for activity reference; 2) 'sign' for sign evaluation; 3) 'num' for number evaluation; 4) 'func' for function evaluation 5) 'expr' for expression containing operators.

'col' type has three elements in the data structure. The second element indicates the full set of activity references used in the formula; in the example, 'A1' is the only one and thus the set contains only 'A1'. The third element is the string representation of the activity reference, 'A1'. Thus, the formula parser can evaluate the result based on the value field of the activity model.

---

<sup>1</sup>*pyparsing* is an approach to creating and executing simple grammars, <http://pyparsing.wikispaces.com/>



**Example 2**

Formula:  $[A1] + 1$

Data structure after parsing:  $(\text{'expr'}, \text{set}([\text{'A1'}]), (\text{'col'}, \text{set}([\text{'A1'}]), \text{'A1'}), \text{'+'}, (\text{'num'}, \text{set}([], 1)))$

The formula belongs to *'expr'* type because it has an operator *'+'*. The activity reference set (second element) contains only *'A1'*. However, there are three more elements after the first two elements which are different from Example 1. This is a distinction between *'expr'* and other types.

The parser separates the formula into two sub-formulas based on the first operator. Thus, the two sub-formulas are:  $[A1]$  and  $1$  respectively. Their corresponding parsed data structures are:  $(\text{'col'}, \text{set}([\text{'A1'}]), \text{'A1'})$  and  $(\text{'num'}, \text{set}([], 1))$ .

Therefore, the parsed data structure of the original formula has to contain these two sub data structures plus the operator. The formula parser can examine the data structure and recursively evaluate its sub data structures to produce the final result.

**5.2.7 Reordering the activity position**

Activity positioning is implemented in order to display the right ordering of activities. In Figure 5.3, the order column enables instructor/TA to reorder the activity position. This function has been implemented in both the Ajax way and the non-Ajax way to provide the best user experience. In the Ajax way, instructor/TA can smoothly reorder activities without refreshing the page. However, in the non-Ajax way, instructor/TA has to wait for the page to refresh in order to reorder an activity. The non-Ajax way acts as a backup solution when Javascript malfunctions, not supported or is disabled in the browser. In the Django backend, the reordering is done by swapping the position field (see section 5.1.1) of the two Activity models.

## 5.2.8 Calculating the numeric grade

**FINAL GRADE INFORMATION**

Activity Type	Calculated Numeric Graded
Name	Final Grade
Short name	Final
Status	released
Due date	None
Percentage	None
Maximum grade	100
Formula	[A1] + [A2] + [Midterm Exam]

**Actions**

- [Edit Activity Details](#)
- [Display Statistics](#)
- [Export all](#)

**Calculate all**

Show 50 entries		Search:				
	Last name	First name	User ID	Stu #	Grade Status	Grade
	Student	A	0aaa0	200000169	calculated	22/100
	Student	B	0aaa1	200000170	calculated	18/100
	Student	C	0aaa2	200000171	calculated	0/100
	Student	D	0aaa3	200000172	calculated	0/100

Figure 5.9: Instructor/TA's view in a calculated numeric activity

Figure 5.9 is the instructor/TA's view in a calculated numeric activity. The student information is retrieved from two data models: 1) **Member** for 'Last name', 'First name', 'UserID' and 'Stu #'; 2) grade model for 'Grade Status' and 'Grade'. Functions are provided for instructor/TA to calculate the student grade based on the formula specified in the calculated numeric activity. Two ways of calculation are provided: calculation of all students and calculation of an individual student. The calculation of an individual student is implemented using Ajax technologies to provide the best user experience (no page refresh).

Error handling is implemented for the Ajax request. A proper error message will be displayed within the field that triggers the calculation. A non Ajax-way to calculate an individual student grade is also implemented in case Javascript malfunctions, disabled or not supported by the browsers. Thus, it ensures the universal accessibility from different devices while enabling the advance features for those devices that support the technologies.

### 5.2.9 Marking student

**MARK ALL STUDENTS**

			Search:	<input type="text"/>
Student Name	Student Number	Current Grade	New Grade	
A Student	200000169	22	<input type="text"/>	/ 0
B Student	200000170	no grade	<input type="text"/>	/ 0
C Student	200000171	no grade	<input type="text"/>	/ 0

Figure 5.10: Marking student

In Figure 5.4, Instructor/TA can mark an individual student through the link provided in the Grade column. It will redirect instructor/TA to a page containing forms to mark a student based on individual marking components. Additional marking information such as late penalty can be added through that page. By clicking the 'Mark all' action link in Figure 5.4, the instructor/TA can mark all the students in a single page, as shown in Figure 5.10. However, marking in this way will not associate the information to any marking component defined in the activity. The instructor/TA can also import the students' grades from a CSV file. The CSV file is manipulated through Python's csv module which provides handy functions to write and read a CSV file.

## 5.3 Core functionalities in student side

The student side of the CMS has a simpler user interface and less functionality compared with the instructor/TA side. Students can examine the course information and their grade in each course activity; students can manage their course group, submit assignments and view marking details.

### 5.3.1 Course view

#### CMPT 165 COURSE INFORMATION

Course Number	CMPT 165
Section	D100
Semester	Spring 2010
Title	Intro Internet World Wide Web
Campus	Burnaby Campus
Instructor(s)	Gregory Baker ( <a href="mailto:ggbaker@sfu.ca">ggbaker@sfu.ca</a> )
TA(s)	Doug Gradstudent ( <a href="mailto:0grad@sfu.ca">0grad@sfu.ca</a> )

Actions
<a href="#">Manage Groups</a>

#### Course Activities

Activity	Status	Grade
<a href="#">Assignment 1 (10%)</a>	graded	22/25
<a href="#">Assignment 2 (10%)</a>	no grade	—
<a href="#">Midterm Exam (20%)</a>	—	—
<a href="#">Project</a>	—	—
<b>Final Grade</b>	calculated	22/100

Figure 5.11: Student’s landing view in a course

In Figure 5.11, students can examine the general course information on the top-left corner. This information is retrieved from the `CourseOffering` data model. Students can also view the list of course activities and the corresponding grade information. The information is retrieved from two data model types; they are `Activity` type and `Grade` type respectively. Students cannot examine the grade information when the activity is unreleased and cannot examine the activity information when the activity is invisible. The activity order is displayed according to the activity order management in the instructor/TA side. Links are provided for every activity so that students can examine the details associated with them. Group management is also provided in the action box on the top-right corner if it is a group activity so that students can form their group, invite students to join the group and perform actions on behalf of the group (e.g., group submission).

**ASSIGNMENT 1 INFORMATION**

Status	released
Due date	2010-03-17 00:00:00
Percentage	10%
Grade Status	graded
Your Grade	22/25 (88.00%)

Submit

**Summary Statistics**

Mean Grade	4.20
Median Grade	0.00
Standard Deviation	8.44
Minimum Grade	0.00
Maximum Grade	22.00

**Histogram**

(refresh the page if histogram is not displayed properly)

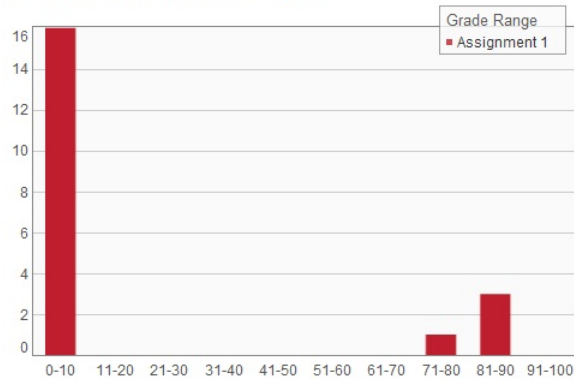


Figure 5.12: Student's course activity view

By clicking the activity link in Figure 5.11, students is redirected to a new page showing the detailed information of the activity, as shown in Figure 5.12. The information in the first table is retrieved from the Activity model and the Grade model. The information of the summary statistics and histogram is generated in the Django backend using the grades information which is retrieved from the Grade model. The histogram is generated using HTML5 canvas element. This is implemented by generating an html table with all the grade range information and use a JQuery plugin to transform it into the HTML5 canvas. Students can also look at their marking summary and submit their assignment.

## Chapter 6

# Overview of Marking Component

This chapter provides an overview of the Marking component from the user perspective. The Marking component mainly deals with the marking service for a numeric graded activity. Typical use cases will be: Instructor/TA specifies the marking components for an activity, then marks the students based on the marking configurations. After that, a student can see the marking summary.

### 6.1 Core functionalities at instructor/TA side

#### 6.1.1 Configuring marking component

**ASSIGNMENT 1 MARKING COMPONENTS**

[Edit Components Content](#) [Edit Components Order](#)

Title	Description	Max Mark	Delete?
Part1	Part 1 was done correctly: good code, comments, etc.	5	<input type="checkbox"/>
Part2	Here, we were looking for...	10	<input type="checkbox"/>
Integration	Parts 1 and 2 were correctly combined into a working solution	10	<input type="checkbox"/>

Figure 6.1: Instruct/TA' view on configuring marking components

In Figure 6.1, instructor/TA specifies all the marking components for the activity and submits. Instructor/TA can also reorder the component position through 'Edit Components Order' link above the table. This is similar to the activity reordering. After that, instructor/TA is ready to mark an activity.

### 6.1.2 Marking student

**ASSIGNMENT 1 MARKING**

Mark for Student: 0aaa1

**Part 1**

Mark  Out of 5

Comment

Common problems:

**Part 2**

Mark  Out of 10

Comment

Common problems:

**Additional Information**

Late penalty   
Percentage to deduct from the total mark got due to late submission

Mark adjustment   
Points to add or deduct for any special reasons

Mark adjustment reason

Overall comment

File attachment

Figure 6.2: Instruct/TA' view on marking components

In Figure 6.2, instructor/TA marks the student based on the marking components; additional information can be given such as late penalty, mark adjustment, comment and file attachment (e.g., instructor/TA may want to send back the commented code assignment

to the student). After instructor/TA submit the marking, the grade change will be displayed immediately in the course activity page. If the activity is a group activity, this step can be applied to group marking as well. By marking on the group, every student in the group receives the same grade.

### 6.1.3 Examining marking summary

**Student 0aaa0 marked on Assignment 1**

**Basic Information**

Marked by	ggbaker
Time	Wed, 14 Apr 2010 03:42:08 -0700
Total Mark	21 out of 25
Mark Approach	Directly to individual

**Additional Information**

Overall comment	
Late Penalty	0
Adjustment Mark	0
Adjustment Mark Reason	
File Attachment	no attachment

**Components Details**

*Part 1*

Description: Part 1 was done correctly: good code, comments, etc.

**Mark:** 4 out of 5

**Comment:**

*Part 2*

Description: Here, we were looking for...

**Mark:** 8 out of 10

**Comment:**

*Integration*

Description: Parts 1 and 2 were correctly combined into a working solution

**Mark:** 9 out of 10

**Comment:**

[Mark based on this](#)

[Marking History](#)

Figure 6.3: Instruct/TA' view on marking summary

Instructor/TA can examine the student's marking summary, see Figure 6.3 which shows the marks on every marking component and the additional information associated with the activity. Actions are also provided for instructor/TA to remark the assignment or to view



the marking history.

## **6.2 Core functionalities at student side**

### **6.2.1 Examining marking summary**

Students can view the marking summary through the course activity page, see Figure 5.12. The marking summary information is displayed in the same way as for the instructor/TA side, but without the ability to mark and view the history.

## Chapter 7

# Overview of the Submission Component

This chapter provides an overview of the Submission component from the user perspective. Submission component provides services for instructor/TA to specify submission components for an activity and provides services to manipulate the submission components. Typical use cases will be: instructor/TA specifies the submission components of an activity; students submit their assignments; after the activity is due, instructor/TA examines student's submissions and marks based on the submissions.

### 7.1 Core functionalities at instructor/TA side

#### 7.1.1 Configuring submission component

Instructor/TA can specify the submission components for the activity, see Figure 7.1. The component types can be URL, archive file, pdf or code file. Instructor/TA can also reorder the component position. This is similar to the activity ordering.

#### 7.1.2 Examining student's submission

Instructor/TA can examine the student's submission, see Figure 7.2. Instructor/TA can view the submission information (e.g., last submission date, last submitter and the group name if the activity is a group activity) and can download the submission for marking purpose. In the action box, marking functionalities are provided to mark student directly

**CONFIGURE SUBMISSION FOR ASSIGNMENT 1**

[Add Component](#)

**Assignment URL**

Type: URL Position:

Description: URL

[Edit](#)

**C/C++ code**

Type: Code Position:

Description:

Maximum Size: 2000 KB

Allowed Type: .cpp,.c

[Edit](#)

[Update Position](#)

Figure 7.1: Instruct/TA' view on configuring submission components

**SUBMISSION FOR ASSIGNMENT 1**

**⚠ This is a group submission. You will submit on behalf of the group Win!**

Due Date:	2010-03-17 00:00:00	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #555; color: white;">Actions</th> </tr> </thead> <tbody> <tr> <td><a href="#">Mark Student</a></td> </tr> <tr> <td><a href="#">Mark Group</a></td> </tr> <tr> <td><a href="#">View History</a></td> </tr> <tr> <td><a href="#">Download All</a></td> </tr> </tbody> </table>	Actions	<a href="#">Mark Student</a>	<a href="#">Mark Group</a>	<a href="#">View History</a>	<a href="#">Download All</a>
Actions							
<a href="#">Mark Student</a>							
<a href="#">Mark Group</a>							
<a href="#">View History</a>							
<a href="#">Download All</a>							
Latest Submission:	2010-04-14 02:23:37						
Late for:	4 weeks						
Current Time:	2010-04-14 04:49:23						
Group:	Win!						
Latest Submitter:	A Student						

**Assignment URL**

Submission Type: URL  
Description: URL

✔ You have made submission to this component.  
**Latest Submission:** 2010-04-14 02:23:15  
**Submitter:** Student, A  
[Download](#)

**C/C++ code**

Submission Type: Code  
Max File Size: 2000 KB  
Allowed File Types: .cpp,.c

✔ You have made submission to this component.  
**Latest Submission:** 2010-04-14 02:23:38  
**Submitter:** Student, A  
**File Size:** 2.8 KB  
[Download](#)

Figure 7.2: Instruct/TA' view on examining student's submission

from this page. Instructor/TA can view the submission history and can download all the submission as a single archive file.

## 7.2 Core functionalities at student side

### 7.2.1 Submitting activity components

**SUBMISSION FOR ASSIGNMENT 1**

⚠ This is a group submission. You will submit on behalf of the group Win!.

Due Date:	2010-08-13 00:00:00
Latest Submission:	2010-04-14 02:23:37
Current Time:	2010-04-14 05:30:19
Group:	Win!
Latest Submitter:	A Student

Actions
<a href="#">New Submission</a>
<a href="#">View History</a>
<a href="#">Download All</a>

Assignment URL  
 Submission Type: URL  
 Description: URL

✔ You have made submission to this component.  
 Latest Submission: 2010-04-14 02:23:15  
 Submitter: Student, A  
[Download](#)

C/C++ code  
 Submission Type: Code  
 Max File Size: 2000 KB  
 Allowed File Types: .cpp, .c

✔ You have made submission to this component.  
 Latest Submission: 2010-04-14 02:23:38  
 Submitter: Student, A  
 File Size: 2.8 KB  
[Download](#)

Figure 7.3: Student's view on examining activity submissions

Student can submit the activity components through the course activity page, see Figure 5.12. Student will first see the submission information of the activity, as shown in Figure 7.2. Student can submit the activity components by clicking on the 'New Submission' action in the action box. Then, student will be redirected to a page to submit the components, see Figure 7.3. If the activity is a group activity, all submission actions are done on behalf of the group.

**⚠ This is a group submission. You will submit on behalf of all your group members.**

Add new submission for Assignment 1

[Back](#)

**Activity Submission**

**Assignment URL**

Description: URL

Url:

**C/C++ code**

Max File Size: 2000 KB

Allowed File Types: cpp, c

Code:  [Browse...](#)

[Submit](#)

Figure 7.4: Student's view on submitting activity components

## Chapter 8

# Overview of the Group Management Component

This chapter provides an overview of the Group Management component from the user perspective. Group Management component provides services for both instructor/TA and student to manage student group. Typical use cases will be: a student creates a group for the course activity; the student invites other students to join the group; these other students either confirm or reject the invitation to join the group. Once the student group is formed, only instructor/TA has the authority to switch students or remove students from group.

### 8.1 Core functionalities at instructor/TA side

#### 8.1.1 Managing student group

Instructor/TA has the full authorization to manage student groups, see Figure 8.1. Instructor/TA can see all the groups for all the activities in a course and the list of students not in a group. These groups can belong to multiple activities. Instructor/TA can change the name of the group, remove students from the group, assign students to a group and form a new group, see Figure 8.2.

**Current Groups**

- Win1 (for Assignment 1)
  - A Student, 0aaa0
  - B Student, 0aaa1
  - C Student, 0aaa2
  - D Student, 0aaa3

- Course Men (for Assignment 1)
  - I Student, 0aaa8
  - G Student, 0aaa6

- ddp (for Assignment 1)
  - M Student, 0aaa12
  - K Student, 0aaa10
  - L Student, 0aaa11

**Students not in any group**

- E Student, 0aaa4
- F Student, 0aaa5
- H Student, 0aaa7
- J Student, 0aaa9
- N Student, 0aaa13
- O Student, 0aaa14
- P Student, 0aaa15
- Q Student, 0aaa16
- R Student, 0aaa17
- S Student, 0aaa18
- T Student, 0aaa19

Figure 8.1: Instruct/TA' view on managing student group

**Create Group**

Group Name

**If this group is for whole semester**

If this checkbox is checked, then when a new group activity is created in the future, it will be automatically added for the group

**Activities for this Group**

Selected	Title	Percent	Due Date
<input checked="" type="checkbox"/>	Assignment 1	10	2010-08-13 00:00:00

**Students for this Group**

Selected	User ID	First Name	Last Name	Student ID
<input type="checkbox"/>	0aaa0	A	Student	200000169
<input type="checkbox"/>	0aaa1	B	Student	200000170
<input type="checkbox"/>	0aaa2	C	Student	200000171

Figure 8.2: Instruct/TA' view on creating group

**CMPT 165 D1 GROUPS**

Win!

A group for Assignment 1.

- A Student, 0aaa0
- B Student, 0aaa1
- C Student, 0aaa2
- D Student, 0aaa3

[Invite](#)

Actions
<a href="#">Create New Group</a>

Figure 8.3: Student’s view on managing group

**Create Group**

Group Name

Group is for whole semester?

This group will stay together for any newly-created activities in this course

**Activities for this Group**

Selected	Title	Percent	Due Date
<input checked="" type="checkbox"/>	Assignment 1	10	2010-08-13 00:00:00
<input checked="" type="checkbox"/>	Assignment 2	10	2010-03-31 00:00:00
<input checked="" type="checkbox"/>	Project	None	2010-04-16 19:12:04

[create](#)

Figure 8.4: Student’s view on creating group



## 8.2 Core functionalities at student side

### 8.2.1 Managing group

Student can examine the groups for a course when he/she is one of the group members, see Figure 8.3. He/she can invite other students to join the group. In this case, the invited student has to accept the invitation in order to join the group. Students can also create a new group for the course, see Figure 8.4. He/she specifies which activity this group belongs to and specifies whether the group will be associated with the newly created activity.

## Chapter 9

# Conclusion and Future Improvement

We have implemented most of the functionalities of the four major components in the CMS. By examining the problems of the current systems used by SFUCS, we have defined the functions domain of the CMS. The principal advantage that the CMS has over the current systems is CMS provides a central access point to manage courses. All the course management information sits within a system sharing the same database. Users no longer manually relate an assignment submission with the student grade. Users now have the choice not to spend hours trying to figure out how WebCT works. Instead, they can use CMS which has the just-enough functionalities. We emphasize the simple-is-beautiful principal throughout the project development; we promise every function is straightforward and is simple to use. The user interface is intuitive and eye-catching and users can efficiently perform the functions they anticipate. We have also implemented some miscellaneous functions to facilitate course management, such as group management, news feed and calculated numeric activity.

From the developer perspective, we have developed a system with a very clean code base. It is attributed to the design pattern that Python+Django emphasize on. We also insist on the Don't Repeat Yourself principle that we maximize the use of utilities module provided by the framework as well as the software communities. At the front end, we bear in mind the browsers compatibility and functional gap between different vendors. We have chosen technologies that work for different vendors of a browser as well as different versions of the same browser. CMS is tested against all major browsers.

However, this project is still under development. The project scope, technical implementation as well as the layout of the system may be subject to change. Future improvement will also be advised by the users when the system is released for SFUCS. However, some features that are not in the current project scope have been considered as future improvement such as course offering planning. So far, the course offering planning has been done manually. There is no central database system handling this information. Instructors and staff need to convey this information through email or manually. Then, they manually resolve the course planning conflicts and issues and post it in the SFUCS website.

# Bibliography

- [1] Django Software Foundation. Django. <http://www.djangoproject.com/>.
- [2] Django Software Foundation. Working with forms. <http://docs.djangoproject.com/en/1.1/topics/forms/>.
- [3] Python Software Foundation. Python programming language. <http://www.python.org/>.
- [4] Allan Jardine. Datatables (table plug-in for jquery). <http://www.datatables.net/>.
- [5] JQuery Team. JQuery. <http://jquery.com/>.
- [6] JQuery UI Team. JQuery ui. <http://jqueryui.com/>.
- [7] Wikipedia. Cascading style sheets. [http://en.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](http://en.wikipedia.org/wiki/Cascading_Style_Sheets).
- [8] Wikipedia. Comma-separated values. [http://en.wikipedia.org/wiki/Comma-separated\\_values](http://en.wikipedia.org/wiki/Comma-separated_values).
- [9] Wikipedia. Django (web framework). [http://en.wikipedia.org/wiki/Django\\_\(web\\_framework\)](http://en.wikipedia.org/wiki/Django_(web_framework)).
- [10] Wikipedia. Html. <http://en.wikipedia.org/wiki/HTML>.
- [11] Wikipedia. Html5. <http://en.wikipedia.org/wiki/HTML5>.
- [12] Wikipedia. Javascript. <http://en.wikipedia.org/wiki/JavaScript>.
- [13] Wikipedia. Scrum (development). [http://en.wikipedia.org/wiki/Scrum\\_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development)).