Django Project Report

Computer science (Nigerian Turkish Nile University)

PROJECT REPORT OF BUILDING COURSE MANAGEMENT SYSTEM BY DJANGO FRAMEWORK

by

Yiran Zhou

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

APPROVAL

Name:	Yiran Zhou
Degree:	Bachelor of Science
Title of Report:	PROJECT REPORT OF BUILDING COURSE MANAGEMENT SYSTEM BY DJANGO FRAMEWORK

Examining Committee:

Dr. Qianping Gu, Supervisor

Dr. Ramesh Krishnamurti, Examiner

Date Approved:

Abstract

This report introduces the process of creating part of a course management system which is a data-driven website used by the instructors and students. This website has four major components: grade, marking, group, and submission. The major part of the group component and minor part of marking component are implemented by me and the implementation details will be introduced in the report. The implementation uses a tool called Django Framework which is an excellent open source web application frame work for complex data-driven website development. The major part of this report will introduce how to use Django to create a database table, web page user interface and inner logic to handle user request by going through the group component implementation process.

Contents

Chapter 1

Introduction

This project report will introduce how to build part of a course management system using the Django framework[1]. Django is an open source web application frame work which is written in Python[2]. This course management system built using Django has four major components each of which has different functionality but similar architecture. In the project report I will demonstrate details of using Django to build one major component of this system: the group component, which is my major contribution to the whole system. Also the technique and process which is showed here can be applied to build the other three components in the course management system as well as other complex database-driven websites.

1.1 Background

SFU has several disparate systems for course management. They are: grade book, which is used to allow students to see their grades in every activity in their course; online marking system, which is used by teaching assistants and instructors to calculate the grade of their students; submission web service, which is by students to submit their assignments. These three systems are logically related but in reality they are implemented as individual systems and located on different servers. So the technical maintenance team needs to spend a lot of time and energy to support these three different systems. Also these systems were built a long time ago and it is hard for the technical team to maintain and add new functionalities which are required as time goes by. So the purpose of this project is to create a new integrated scalable course management system with all the existing functionalities in these three systems, as well as new functionalities if required.

1.2 Significance

The new course management system will integrate all existing functionalities of the three systems as well as other new functions. Thus the technical team only needs to maintain one system instead of three. The scalability of the new system will allow them to add more features to the system in future. Students and professors only need to log into one system to accomplish what they did on the three systems before. This will be more convenient and significantly improve their efficiency.

1.3 Method used

This system is built using Django web application framework. Django was originally developed for the news-oriented site of the world company in Lawrence, Kansas[3]. It simplifies the development process of complex, data-base driven web applications like a news-oriented site. Its well-designed framework includes three major parts: model, view and template[4]. Our course management system consists of four components which are grades, marking, group and submission. Each component contains those three parts. When we develop the course management system, we first design the model of the relative component for data architecture, then the template for user interface, at last we implement the view which includes all the functions.

1.4 Limitation

Complex websites such as the course management system usually take some time to test and validate. The system may have some potential bugs or flaws because of the development time constraint. However because of the flexibility and powerful functionality of Django, these bugs or flaws can be fixed.

1.5 Report Organization

In next Chapter, which is Chapter 2, we give the preliminaries of the report. Chapter 3 is an overview of the whole project. From Chapter 4 to Chapter 6, we will give the details of each component in Django to implement one process in the group component of the system, namely creating a new group for some group activities. Chapter 4 describes how to deal with the database to create database tables for groups. The user interface for input information for a new group is discussed in Chapter 5. Chapter 6 introduces the view function for processing the create group request. Chapter 7 contains all the functionalities that I have implemented.

Chapter 2

Preliminary

2.1 Django framework

Django is an open source web application frame work written in Python. The primary goal of Django is to make the development of complex, data-based websites easier. Thus Django emphasizes the reusability and pluggability of components to ensure rapid developments. Django consists of three major parts: model, view and template[4].

2.1.1 Model

Model[4] is a single, definitive data source which contains the essential field and behavior of the data. Usually one model is one table in the database. Each attribute in the model represents a field of a table in the database. Django provides a set of automatically-generated database application programming interfaces (APIs) for the convenience of users.

2.1.2 View

View[4] is short form of view file. It is a file containing Python function which takes web requests and returns web responses. A response can be HTML content or XML documents or a "404 error" and so on. The logic inside the view function can be arbitrary as long as it returns the desired response. To link the view function with a particular URL we need to use a structure called URLconf which maps URLs to view fucntions.

2.1.3 Template

Django's template[4] is a simple text file which can generate a text-based format like HTML and XML. The template contains variables and tags. Variables will be replaced by the result when the template is evaluated. Tags control the logic of the template. We also can modify the variables by using filters. For example, a lowercase filter can convert the variable from uppercase into lowercase.

2.2 Python

Python[2] is the language used to build the Django framework. It is a dynamic scripting language similar to Perl[5] and Ruby[6]. The principal author of Python is Guido van Rossum[7]. Python supports dynamic typing and has a garbage collector for automatic memory management. Another important feature of Python is dynamic name solution which binds the names of functions and variables during execution[2].

Chapter 3

Project overview

3.1 An overview of the project

Our desired result of the course management system is an integrated web-based system combining several current systems in use. It will include the following four major components:

- Grade component
  This component is basically associated with course assignments and exams. In this component the instructor can define the assignment/exam details such as assignment/exam name, percentage, due date and so on. The instructor can also view the grades of every student in the course or change the formula used for calculating the final grade. A student can view his/her own grades in all the courses he/she has taken. It also provides functionality such as import/export student grades from/to an excel sheet.

- Submission component
  This component is used to replace the submission system we currently have. It deals with the student submissions for assignments. It can validate some commonly used file extensions to see if the submission file is in the correct form and associate the correct submission with the assignment.

- Marking component
  This component is used to mark student assignments and exams. The instructor or the teaching assistant can mark all questions in an assignment and record some

common problems or mistakes made by the students. The instructor or the teaching assistant can also mark the assignment associated with a student or with a group if it is a group assignment. This component also provides functionalities for the instructor to copy the course setup from one course to another if they are the same course in different semesters or different sections, thereby saving time. I have been in the marking component development team for a while and implemented the copy course setup function and functionalities associated with "common problems". However these functions have been changed significantly after I switched to the group component development team. For details of the marking component, please refer to the project report of Li Tan[8].

- Group component
  The Group component is used by instructors and students to manage groups in the class. Sometimes there are group assignments, so students can form a group to complete these assignments. This component allows the instructors and students to create groups. Instructors can assign students to a group and students can invite other students to the group they belong to. I completed the majority of the work for this component and will introduce this component in detail in Chapter 7.

Besides these four major components, there is another component separate from all the above four. This component is called core data which contains information such as courses, students and semesters. This data is used in the above four components.

3.2 An overview of the Django framework development process

To build such a complicated web system, we need three major parts for each component: database, user interface and the functions to interact in between. Django framework provides sufficient functionalities to implement these three parts. Corresponding to database, user interface and functions in between, Django has model, template and view components to deal with each part respectively. Django's model component helps programmer to define and maintain tables in the database, while its template component helps to write html files using a combination of both html syntax and Django syntax. For those functions in between, Django provides a view component which reads the input from user interface and makes corresponding changes in the database.

Chapter 4

Create the database tables

To get started, we need to create a backend of the system which is the database. All the tables in the database for this course management system include information for courses, instructors, students, assignments, groups and so forth. These tables are created initially when the course management system is deployed. Some information is input into the database at the beginning, such as semester information and course information. However, most information will be inserted or updated in the database dynamically (For example, creating a group). Every time we want to create a new group, we will insert a tuple into the Group table to make the database consistent with the real world.

To have a Group table in the database, we first need to choose which database we are going to use. Django supports almost all popular databases such as mysql, sqlite3, and oracle. The one we used for this course management system is sqlite3. We only need to write one sentence to setup the database:

```
DATABASE_ENGINE = 'sqlite3'
```

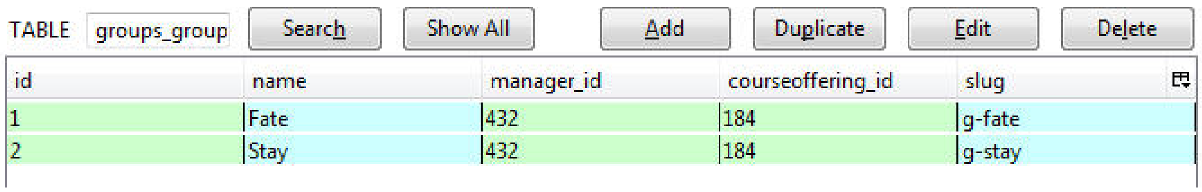
Next, we create the Group table in the database. Django uses a class called model[R] to represent the database table schema. To create a table we just need to write a new class derived from model class. Here is an example:

```
class Group(models.Model):
    """
    General group information in the courses
```

```
"""
name = models.CharField(max_length=30, help_text='Group name')
manager = models.ForeignKey(Member, blank=False, null=False)
courseoffering = models.ForeignKey(CourseOffering)
slug = AutoSlugField(populate_from=autoslug, null=False, \
editable=False, unique_with='courseoffering')

class Meta:
    unique_together = ("name", "courseoffering")
```

This piece of code creates a table called group which stores the information of groups in courses. The table has three columns: name corresponding to the name of the group; manager corresponding to the group manager; courseoffering corresponding to the course that the group is in. Manager and courseoffering are foreign keys that refer to two other tables: Member and CourseOffering. The structure of these two tables is very similar with the group table. The actual Group table in the database looks like this:



id	name	manager_id	courseoffering_id	slug
1	Fate	432	184	g-fate
2	Stay	432	184	g-stay

Figure 4.1: Group table in database

The group component has two tables of its own, both of which are used in the creating group process. They are Group table which has already been introduced above, and GroupMember table which represents the memberships of students in the group. I will skip the details of the GroupMember table since it has a similar structure to the Group table. Member table and CourseOffering table are the foreign keys of Group tables and are defined in the core data component, which is separate from the four sub-systems and provides information used all over the course management system.

These Group and GroupMember tables contain all group specific information for operating the group component including operation of creating group. Once we create a group, we insert a tuple representing the new group into the Group table, and insert several tuples representing the members into the GroupMember table. If a student quits or moves to another group, we can update the corresponding tables to match.

Chapter 5

Create the User Interface

Although we have created the Group table and GroupMember table and can update them using the Django model component, we should not allow the user to manipulate the database directly. Otherwise it would be a disaster for both the users and the technical maintenance team. Instead, we create a user interface to let users interact with the data indirectly. Django provides the template component to create the user interface for users.

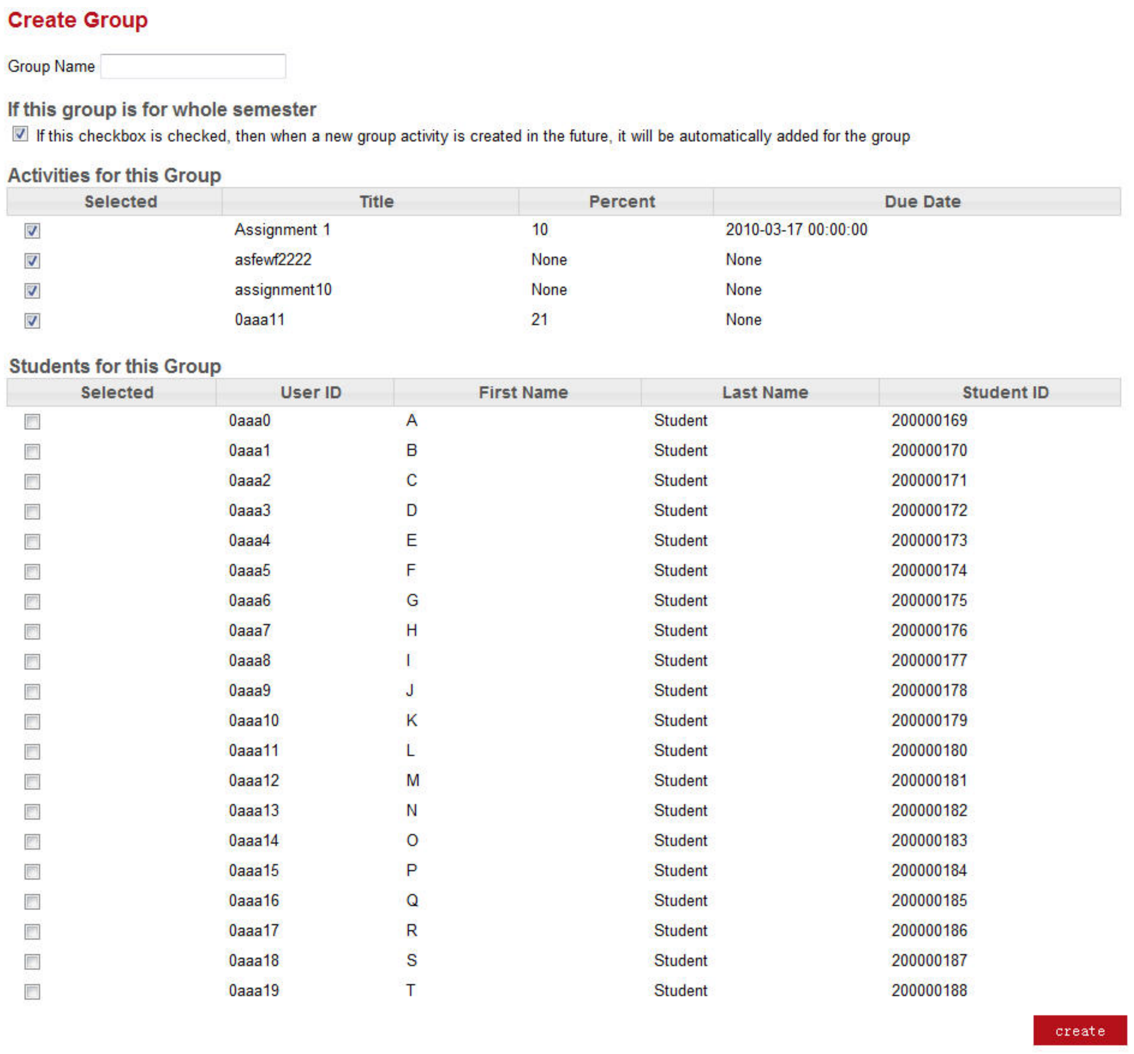
A template is just a simple html file with some Django syntax mixed in. Every template corresponds to a web page which the users will use to interact with the system. Here is the template for creating a group:

```
{% extends "base.html" %}
{% load course_display %}

{% block content %}
    <h1>Create Group</h1>
    {% if user %}
        <form action="{% url groups.views.submit course_slug=course.slug  %}"\
                 method="post">
            <div class="group_name">
                <label for="name">Group Name </label>
                <input type="text" name="GroupName" id="name" />
            </div>
            <div class = "group_for_semester">
                <h2>Group is for whole semester?</h2>
                {{ groupForSemester.selected }}
                <label for="id_selected">This group will stay together for \
                            any newly-created activities in this course</label>
            </div>
            <div class = "datatable_container">
                <h2>Activities for this Group</h2>
                <table class="display" id="activities">
                <thead>
                    <tr>
                        <th>Selected</th>
                        <th>Title</th>
                        <th>Percent</th>
                        <th>Due Date</th>
                    </tr>
                </thead>
                <tbody>
                {% for act in activityList %}
                    <tr>
                        <td>{{ act.activityForm.selected }}</td>
                        <td>{{ act.name }}</td>
                        <td>{{ act.percent }}</td>
                        <td>{{ act.due_date }}</td>
                    </tr>
                {% endfor %}
                </tbody>
                </table>
            </div>
            <div><input class = 'submit' type="submit" value="create"/></div>
        </form>
        {% else %}
        <p> Student does not exist </p>
```

```
    {% endif %}
{% endblock content %}
```

Every line surrounded by "% %" is a Django sentence. Most of them deal with simple "if-else" and "for" loop. The line surrounded by double curve brackets " " is a variable, whose value is passed from a view function whenever this template has to be displayed. This will be discussed in the next chapter. The rest of the code is just simple html sentences. The actual web page of this template looks like this:



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
✓	Assignment 1	10	2010-03-17 00:00:00
✓	asfewf2222	None	None
✓	assignment10	None	None
✓	0aaa11	21	None

Students for this Group

Selected	User ID	First Name	Last Name	Student ID
☐	0aaa0	A	Student	200000169
☐	0aaa1	B	Student	200000170
☐	0aaa2	C	Student	200000171
☐	0aaa3	D	Student	200000172
☐	0aaa4	E	Student	200000173
☐	0aaa5	F	Student	200000174
☐	0aaa6	G	Student	200000175
☐	0aaa7	H	Student	200000176
☐	0aaa8	I	Student	200000177
☐	0aaa9	J	Student	200000178
☐	0aaa10	K	Student	200000179
☐	0aaa11	L	Student	200000180
☐	0aaa12	M	Student	200000181
☐	0aaa13	N	Student	200000182
☐	0aaa14	O	Student	200000183
☐	0aaa15	P	Student	200000184
☐	0aaa16	Q	Student	200000185
☐	0aaa17	R	Student	200000186
☐	0aaa18	S	Student	200000187
☐	0aaa19	T	Student	200000188

Figure 5.1: Group table in database

This template first displays a text box for inputting the group name. Second, it shows a check box to ask the group creator if this group is for all group assignments in this course. Third, it displays a table (not a database table) containing assignments, so the group creator can choose the assignments that the group is for. Next, the template displays a student table for the creator to choose the students belonging to this group. Last, there is a “create” button at the bottom of the page. Once the creator clicks the “create” button, the group name, the assignments and students chosen by the creator will be packaged in a http request object and sent to the corresponding view function for processing.

Chapter 6

Implement the view function

So far we have our backend database and the frontend web page user interface. What we need now is the logic in between to deal with the user requests and maintain the database. Django view component provides a set of application programming interfaces to fulfill our need and help us implement the logic.

The Django view file is where we write our function to achieve the above two goals. First, it is used to pass parameters to the template and call the right template for the user. Every time we input a URL in the address bar or click a hyperlink in the system, Django will call the right view function based on that URL. Then the function will return a template as well as the corresponding parameters. Thus we can see the actual web page displaying the information we need. Second, if we submit something such as create group, the function will have an http request as its input parameter. Based on that parameter the database is updated or the user is provided the required information. The view function for creating a group is given below:

```
def submit(request,course_slug):
    #TODO: validate activity?
    person = get_object_or_404(Person,userid=request.user.username)
    course = get_object_or_404(CourseOffering, slug = course_slug)
    member = Member.objects.get(person = person, offering = course)
    error_info=None
    name = request.POST.get('GroupName')
```

```
group = Group(name=name, manager=member, courseoffering=course, \
    groupForSemester = groupForSemester)
group.save()

if is_course_student_by_slug(request.user, course_slug):
    for activity in selected_act:
        groupMember = GroupMember(group=group, student=member, \
            confirmed=True, activity=activity)
        groupMember.save()

    messages.add_message(request, messages.SUCCESS, 'Group Created')
    return HttpResponseRedirect(reverse('groups.views.groupmanage', \
        kwargs={'course_slug': course_slug}))

elif is_course_staff_by_slug(request.user, course_slug):
    students = Member.objects.select_related('person').filter(\
        offering = course, role = 'STUD')
    for student in students:
        studentForm = StudentForm(request.POST, \
            prefix = student.person.userid)
        if studentForm.is_valid() and \
            studentForm.cleaned_data['selected'] == True:
            for activity in selected_act:
                groupMember = GroupMember(group=group, student=student,\
                     confirmed=True, activity=activity)
                groupMember.save()

    messages.add_message(request, messages.SUCCESS, 'Group Created')
    return HttpResponseRedirect(reverse('groups.views.groupmanage',\
         kwargs={'course_slug': course_slug}))
```

This function deals with the group created by the user. In the last chapter, the user inputs the group name and chooses the assignments and students for the group. The user then clicks the ”create” button. As soon as the user clicks the button, the request is sent to this function as a parameter. The first thing this function does is gather the group information including the group name which is input by the user, the course, and the user who created this group. These three pieces of information is exactly what we defined in the Group table schema in the model chapter. They form a tuple regarding this new group and is inserted it into the database by the group.save() function. After this, the function gathers information to create group member tuples to insert into GroupMember table. Once the database is updated, a message is created to tell the user that the group was successfully created. Finally the web page is redirected to the group component home page, which displays a list of groups for this course including the newly created group.

Chapter 7

Functionalities implemented in group component

In the last three chapters we introduced the basic work flow of implementing the functionality of creating a group. First we create the database tables for storing the group and group member information. Then we create the user interface for the user to input the group name, activities (include both assignments and exams) and student information. Finally we add the view function to get the user input, process it and store it into the database. In this chapter I will cover all the group functionalities implemented by me. All the implementation processes of these functionalities follow the last two steps for each of the three described above (we only need to create the database tables once).

7.1 An overall view of group component

As mentioned in Chapter 3, group component has two data tables, one is the Group table and the other is the GroupMember table. All the functionalities of group component operate on these two tables; they either insert new tuples into these two tables or update existing tuples.

The group component currently has more than ten templates including that of create group. Other templates are for instructors and students to view the group list, inviting group members, confirming an invitation, edit group information, etc. These templates constitute the major user interface of group component and provide sufficient support for users to perform their desired operations on the web pages created by these templates.

Group component also has a bunch of view functions for processing the user request. These functions basically match the purpose of the templates. All the functions, such as viewing group list, creating a group, inviting a student to a group, and confirming invitation, have their corresponding templates. With these functions, the group component can deal with users' requests and retrieve data from the database to show the desired information to users, as well as update the database with the change the user made through the web page user interface.

7.2 Functionalities in the project scope

The following functions in the view file with their corresponding templates are implemented to fulfill the project requirement. With those functions the following requirements in the project scope can be accomplished: an instructor/ student can view the groups that already have been created; a instructor can create a group; a student can invite other students to the group; a student being invited can confirm/refuse the invitation; an instructor can assign a student (who is not in any group) to an existing group.

- Group manage function and its template
  This function is used to display the group list for instructors and students. It first checks the user who has logged into the course management system, based on the ID of the user. It determines whether this user is a student or an instructor. Secondly, if the user is a student, the function will use a SQL query to find all the groups which the student belongs to and all the group members in those groups. Then this information is passed to a template dedicated to display group list for student. Then the web page created by this template will show a list of groups the student is in and all the group members in each group. There is also an invite button in each group for this student to invite other students to join his group, and a create button for this student to create a new group if the student does not belong to any group. If the user is an instructor then the instructor can see all the groups in this class, and have more buttons to manage every group including creating new groups, removing students in one group, assigning students to a group and so on.

- Create group function and its template
  This function is the example I used in the previous three chapters. Please see previous three chapters for details.

- Invite students function and its template
  Once a student creates a group, he can invite other students to his group. As I mentioned in the section of group manage function, there is a button called "invite" in the group manage web page. If the student clicks the invite button, it will call the invite student function. At this stage the invite student function just simply directs the student to an invite student web page which is created by the function's corresponding template. The student inputs the user id of the one he/she wants to invite in a text box and click the invite button. Then the user id will be passed back to the invite student function. The function will check if this user id is valid. If it is, the function will create a new group member tuple and insert it into the GroupMember table. However, because this invitation has not been confirmed by the other student, one Boolean field in the group member tuple which is called confirmed will be false.

- Confirm invitation function and its template
  After a student is invited to join a group, he/she can see the group which he/she is invited to in the group manage web page. There is a confirm button and a refuse button under the group name. If the student wants to join that group, he/she can just click the confirm button and the confirm invitation function will be called. This function will get the previously created group member tuple and change the Boolean field confirmed to be true. If the student chooses to refuse this invitation, then the group member tuple will be deleted.

- Assign student function and its template
  The instructor of the course can also assign a student to a group if the student does not belong to any group. The instructor can just click the assign student button in the group manage web page and it will call the assign student function. This function will first find all the students that are not in any group and direct the instructor to the assign student web page created by the function's corresponding template. In the web page the instructor chooses a group that the students need to be assigned to, selects the students and click the assign button. This request will then go back

to the assign student function. The function will create group member tuples for the assigned students and insert all the newly created tuples into the GroupMember table.

7.3 Functions out of project scope

The previous functions provide basic functionalities for group management. Besides these I also added some other functions which are not in the project scope but will make the group component easier to use.

- Switch group functions and its template
  An instructor can switch a group member from one group to another group. This situation may not happen frequently but it is still good to have this functionality. Once a student requests to switch a group and the instructor approves it, then the instructor can go to group manage web page, click the switch group button that is under the group that the student currently belongs to. It will call the switch group function and this function will find all the group members belonging to that group and all the groups that the student can switch to. Then the function will direct the instructor to a switch group web page showing the information found by the function. The instructor just selects the student who wants to switch the group, and chooses the new group from a drop down menu and clicks submit. It will then go back to the function and update the group member tuples belonging to that student. With the GroupMember table updated, the next time the student logs in, he/she will see himself/herself in the new group.

- Add activity to group function
  If a course has several group activities, and when a group is created at the beginning of the semester only some of the group activities are associated with the group, then this function come into play. As I mentioned in the create group example, when a group is created, the user chooses activities that this group is for. As the semester goes on, there will be new group activities created by the instructor. So when a new activity is created it will call this function to add the newly created activities to all existing groups. By doing this the user no longer has to worry about the new activity since this function will take care of the relationship between the new activities and the already existing groups automatically.

Chapter 8

Conclusion

The Django framework gives us a simple and reliable way to create the course management system. It provides powerful functionalities and concise syntax to help programmers deal with the database, the web page and the inner logic. The experience of developing the group component in the system also helped me learning a lot of website development with Django. Within the Django framework, we have successfully accomplished the requirements of the system. Once this system passes the testing phase, it can be used to serve students and instructors and substitute several systems currently in service. It will make the work for instructors to manage the course much easier. It also can simplify the operations for students with grade book, submission, and group management all in one system. In short, this system will bring great user experience to both instructors and students. The only limitation for this course system is that although the developers have been testing it with various use cases, it may still encounter problems during real time use. However, even if that happens, the flexibility of Django would provide a simple way to fix the problem, as well as add new features into the system.

Bibliography

[1] Django homepage. http://www.djangoproject.com/.

[2] Python documentation. http://www.python.org/doc.

[3] Django(web framework). http://en.wikipedia.org/wiki/Django.

[4] Django documentation. http://docs.djangoproject.com.

[5] The perl programming language. http://www.perl.org/.

[6] Ruby programming language. http://www.ruby-lang.org/en/.

[7] Python(programming language). http://en.wikipedia.org/wiki/Python.

[8] Li Tan. Course management web system, capstone project report, computing science, simon fraser university, 2010.