Improvements to COEP Moodle

A Project Report

Submitted by

Harshadkumar H. Waghmare	**111003043**
Pankaj A. Bagul	**111003037**
Rahul V. Waghamare	**111003047**

in partial fulfilment for the award of the degree

of

B.Tech. Computer Engineering

Under the guidance of

Prof. Abhijit A. M.

College of Engineering, Pune

DEPARTMENT OF COMPUTER ENGINEERING, COLLEGE OF ENGINEERING, PUNE-5

May, 2014

DEPARTMENT OF COMPUTER ENGINEERING,

COLLEGE OF ENGINEERING, PUNE

CERTIFICATE

Certified that this project, titled "Improvements to COEP Moodle" has been successfully completed by

Harshadkumar H. Waghmare	**111003043**
Pankaj A. Bagul	**111003037**
Rahul V. Waghamare	**111003047**

and is approved for the partial fulfilment of the requirements for the degree of "B.Tech. Computer Engineering".

SIGNATURE	SIGNATURE
Prof. Abhijit A. M.	**Dr. J. V. Aghav**
Project Guide	**HOD**
Department of	**Department of**
Computer Engineering	**Computer Engineering**
and Information Technology,	**and Information Technology,**
College of Engineering Pune,	**College of Engineering Pune,**
Shivajinagar, Pune - 5.	**Shivajinagar, Pune - 5.**

Abstract

Moodle is an open source Learning Management System(LMS). Moodle has rich set of functionalities related to site management, user management and course management. Some typical features of Moodle are Assignment submission, discussion forum, files download, grading, online quiz, online news and announcement,etc. Moodle modular system provides many types of plugins like activities, themes, question types, assignments.

College **O**f **E**ngineering **P**une (COEP) Moodle uses many facilities provided by Moodle like assignment submission, discussion forum, online quiz, etc. While using this system administrators found some special requirements and improvements in existing moodle. Specific requirements of this system are like static bulk management of courses, simple multiple choice question type plugin, grader report enhancement for user files and student gradebook for all the courses she is registered. The implementation of these requirements will help the administrators to use COEP moodle easily and according to their requirements.

Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Moodle

Moodle has many features typically of an e-leaning platform. Moodle's basic structure is organized around courses, these are areas within moodle where teacher can post learning resources and activities to students. A user with the role of Administrator is typically in charge of a Moodle site once it has been installed, although some tasks may be given to others by assigning them a role such as Manager. The M in Moodle stands for modular. The easiest and most maintainable way to add new functionality to Moodle is by writing one of these types of plugin.

1.1.1 COEP Moodle

College **O**f **E**ngineering **P**une (COEP) Moodle uses many facilities provided by Moodle like assignment submission, discussion forum, online quiz, etc. COEP also uses Moodle for managing courses across all the departments of institute. Moodle provides a number of ways of managing authentication, called authentication plugins, whereas COEP Moodle uses LDAP server for authentication.

By the perspective of user like administrator, student and teacher it is required to have the software be user-friendly and customizable according to her use. Therefore some of the requirements observed necessary by administrators of COEP Moodle are selected for implementation.

1.2 Improvements to Moodle

Moodle seems perfect in its design and implementations, but there are some features that needs to be added or some of them are needed to be changed in design to achieve maximum throughput in less work. Some of those corrections were chose and implemented.

1.2.1 Simple Multiple Choice Subplugin

The current scenario of adding Multiple Choice questions[9] to a quiz are pretty vast. So, making it easy, i.e. putting a question and adding required options with one answer correct should add a question and the type will be Simple Multiple Choice questions type.

The thing that makes the current scenario of adding multiple choice questions vast is the feedback. A lot of feedback fields are created, i.e feedback for question, for each answer, and to the overall question. More about the task is discussed in further chapter 3.2 .

1.2.2 Static Management of Courses

Management of courses is the job of Manager or Administrator of Moodle. In the recent versions of course management pages, the single button action is utilized which refreshes the page on each action, which makes the job slow. So a job of designing a plugin was required so that the all of these actions to be done would be recorded and on clicking submit button all the actions should take place.

The current architecture of management of plugins in moodle had to be changed. Now on completion of task, the actions are recorded and can be acted upon in one click.

1.2.3 Grader Report Enhancement for user files

The main task of Teachers in Moodle, is to grant grades to students based on their work. The same is done in moodle in a form that takes input from teachers and on submitting the grades would be assigned. In current Moodle page, Teacher has to look at the students code by going to his submission page then download his work and then give grades on this page. Bulk grading is not possible with this approach as teacher will have to navigate a lot throughout.

So a link was needed in the same input form. A link which would directly download the work submitted by the students.

1.2.4 Student Gradebook

Moodle does not support to get the grade report of each student. i.e. the student's marks of the semester for all the subjects.

So task was to design a page, which will help students to see their current status of each courses as well as could export the semester report.

Chapter 2

Overview Of Moodle API

2.1 Moodle

2.1.1 Introduction

Modular **O**bject **O**riented **D**ynamic **L**earning **E**nvironment (Moodle)[1] as the full-form specifies the Moodle code is Modular and Object Oriented. The modularity is gained by the use of API's in Moodle[2]. There are API's in moodle for nearly everything, and these API's are grouped as core API's and complementary API's. Core API's[2] are the ones which were developed on the base code of Moodle, while complementary API's are developed while writing a plugin for Moodle. These plugins serve the purpose of customization because they can be installed manually and to the need, i.e. installing all the plugins is not mandatory, but depends on need of the user. For example, plagiarism plugin. These complementary API's are also known as Activity module API.

Moodle is very much flexible and customizable in the working environment. The varied weekly activities such as assignments, quiz, adding some study material, adding link etc. makes moodle flexible to use. It is on the verge of becoming the complete platform as the most varied dynamic learning systems. The customizability comes from the installation of required plugins.

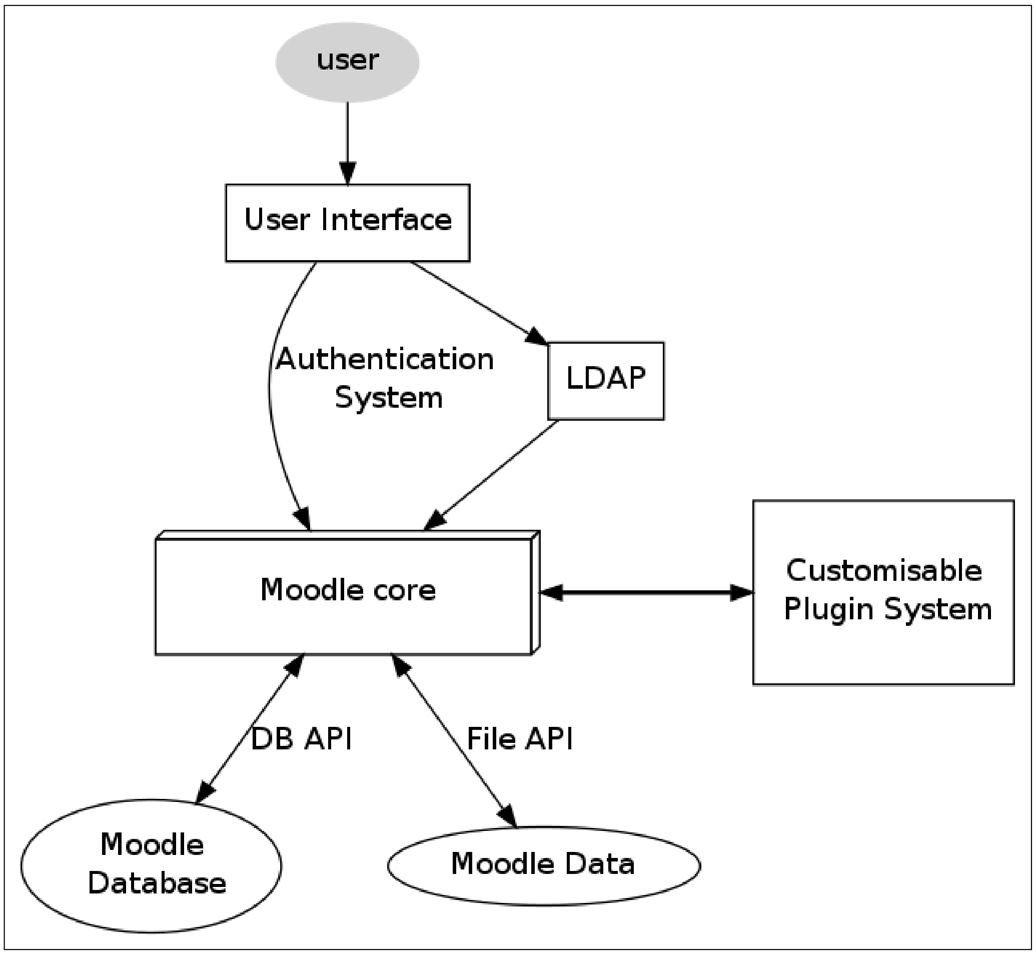


Figure 2.1: **M**odular **O**bject **O**riented **D**ynamic **L**earning **E**nvironment (Moodle) Basic Architecture Diagram

2.2 Moodle API

Introduction

The **A**pplication **P**rogramming **I**nterface (API) are developed in the sense to add modularity, and all of these API's are written Object Oriented. These API provides different tools in moodle scripting and which are then used in moodle plugins to add new feature to enhance the user customizability. The Core API are used everywhere in the moodle code, and thus are used in basic code design. For example, the Access API deals with the current user log-ins, and her capabilities at the particular page.

We found following APIs useful for our work, Access API, String API, Form API, File API, Page API, Output API, Data manipulation API, Gradebook API and more[2].

2.2.1 Access API

Access API[3] deals with granting the access of Moodle features to the logged in user by checking her context and capabilities that are defined while defining the roles. Thus access API works as the means of checking if the user has capability to even view this feature, if yes then allow the access to that feature. Thus though some users can access a particular file that also consists of some administrator features, she'll not be even able to see/sense those features, that's the level of Moodle security which serves the purpose of restricting access. This type of security makes it hard to crack by Mal-users.

The defining of new capabilities is done while writing the code of the particular plugin in its db/ directory in the file named *access.php*, while granting of these capabilities is done at the installation of the plugin. Access API has the sequence of features predefined, i.e. to define the capabilities to fetch the contexts with the help of some functions. Defining of capabilities of done in db/access.php. The various macros that are required to define in this file are defined in moodle/lib/accesslib.php file, which is the library file. Therefore the file would contain all the required functions necessary for this API are in accesslib.php, thus we can say that Access API is written in accesslib.php.

Listing 2.1: code snippet from moodle/mod/assign/db/access.php[4]

```
$capabilities = array(
  'mod/assign:view' => array(
             'captype' => 'read',
             'contextlevel' => CONTEXT_MODULE,
             'archetypes' => array(
             'guest' => CAP_ALLOW,
             'student' => CAP_ALLOW,
             'teacher' => CAP_ALLOW,
             'editingteacher' => CAP_ALLOW,
             'manager' => CAP_ALLOW
             )
           )
         );
```

The capabilities array as defined in db/access.php. The name of the capability is of type *plugintype/pluginname:capabilityname*. The file is run at the installation phase of the plugin, so once the capabilities are defined, those are stored in database table *capabilities*.

The important functions are then defined in accesslib.php library file. Once the context is stored in database, it can be fetched from the use of Moodle functions in various ways. An exception is thrown if context can not be created. The main class written to get context is *class context* defined in lib/accesslib.php. The various inherited classes from this class like *class context_system* from system context, *class context_course* to get course context. All of these classes have a *function instance* which returns the context if already present, else returns a *dml exception*, which means the record is not found in database. The default context is also granted if SYSCONTEXTID is defined and is allowed to be cached.

Defining the context and then fetching the context does not solve the problem of access control. So the various functions are defined which returns Boolean as of the result if the user has that capability or not. For example, *function has_capability()* and *function require_capability()* are the two functions which does the job. *function require_capability()* calls *function has_capability()*, with adjusted arguments.

In this way serving of access control is done by Access API, which is the most used and most important Core API in Moodle design.

2.2.2 String API

On the topic of localization of web, moodle ensures every possibility to add the local language plugin as **L**anguage **P**ackage (langpack), to improve localization. Currently moodle consists of over 90 langpacks and the count is not constant. Thus every locality is ensured to take benefit from Moodle.

This localization is possible because of the string API[5], defined in moodle. The API deals with how you want to display the text strings in User Interface.

There are two types to define the strings, one is to define the strings in main moodle/lang/ directory, and the other one is to create the lang/ directory in the plugin directory. The lang/ directory consists of the lang packs with the acronym-ed names of directories as the language content. The function of the files from these directories is that to store the strings with an identifier as the key of associative array.

Listing 2.2: Language strings from moodle/lang/en/moodle.php[6]

```
$string['courseshown'] = 'Course successfully shown';
```

$string is an associative array with the key as *'courseshown'* and value as *'Course successfully shown'*.

The function that retrieves this value from and plays most important role is *function get_string()*. It is defined in file moodlelib.php, in an *interface string_manager*, which returns the localized string.

For example, *get_string('someidentifier');* will first check if there is any local directory, else will try to search the key identifier in moodle/lang/ directory. The argument can also be given as the language name, to let *get_string()* look for that identifier string in that specified language directory. The default language directory is 'en' (English) language as the default locale.

2.2.3 Page API

The **H**yper**T**ext **M**arkup **L**anguage (HTML) content before rendering need to set up correctly. The PAGE API[7] is very important for theme developers, as the page setups are different for different themes. Thus settings of the information about the HTML page is done by PAGE API, and the information is stored in the Global Variable, $PAGE, which is always checked while setting the page. All this set information is then rendered to the OUTPUT API[8], which then shows the HTML content on the web page.

Listing 2.3: moodle/course/action.php[17]

```
$PAGE->set_url(new moodle_url('/course/action.php'));
$PAGE->set_context(context_system::instance());
require_login();
```

The most important thing to set in the PAGE API, is to set url of the page, which is must. Then setting of page context is done, that will define the context for this page. The importance of setting the page context is that, now every page will have some default required capabilities to view the page. This crosschecking is done by *require_login()* function defined on the next line. The *require_login* also set the basic page settings in $PAGE variable. The page layout can also be set using page API, default layout is 'base', which is normal page without blocks. 'Standard' layout is the one with navigation bars on both the sides of the page, it is recommended for most of the pages with general information.[7]

These settings should be done before starting the output, i.e. before defining the OUTPUT API.

2.2.4 OUTPUT API

OUTPUT API[8] is pure HTML based functions containing API. Whatever the content that we see on any page of moodle, is all rendered by this OUTPUT API. The various classes on this API implements the representation of HTML texts in the web page. The classes contains the rich metadata which is used by rendered by output renderers to generate the output.

The functions which has *html_* as the prefix, are part of this OUTPUT API. Consider the following code snippet, from file moodle/course/smanage.php, which creates a form,

Listing 2.4: moodle/course/smanage.php[16]

```
echo $OUTPUT->paging_bar($totalcount,
        $page, $perpage, $pagingurl);
echo html_writer::
     start_tag('form',
               array('id' => 'movecourses',
               'action' => $actionurl,
               'method' => 'post'));
echo html_writer::
               start_tag('div');
echo html_writer::
               empty_tag('input',
                  array('type' => 'hidden',
                        'name' => 'sesskey',
                        'value' => sesskey()));
foreach ($searchcriteria as $key => $value) {
  echo html_writer::empty_tag('input',
                    array('type' => 'hidden',
                    'name' => $key,
                    'value' => $value));
}
echo html_writer::table($table);
echo html_writer::end_tag('div');
echo html_writer::end_tag('form');
echo html_writer::empty_tag('br');
```

The $actionurl and $pagingurl are set to *new moodle_url('smanage.php')* and *new moodle_url('/course/smanage.php', array('categoryid' => $id, 'perpage' => $perpage) + $searchcriteria).* $table variable is initialized with some more HTML content required to set fields of table. This simple form is created for moving the courses in a particular category, The actionurl given is the same filename thus there is initial check begin done in the same file about the variables passed, i.e. if they are received or not using *optional_param()* function. And then the required action is taken in that function. The session key is passed and is then checked while receiving the variables arguments, this is to avoid any other redirects to the same action, like by using **C**ross **S**ite **S**cripting (XSS) attacks. These attacks are then saved by *optional_param()*, by giving the last argument as the expected type of the passed value, as defined in moodlelib.php library file. Thus the scripts will not be run if given in url context, unless PARAM_SCRIPT is expecting. But there is no such macro defined in moodlelib.php, so XSS attacks[27] are saved.

The rendering of HTML output is done with the help of a global variable $OUTPUT, which is the global object, and for rendering the HTML content. For example, consider the output API content from actioncat.php,

Listing 2.5: output api content from actioncat.php[18]

```
echo $OUTPUT->header();
echo $OUTPUT->notification(
     get_string('categorymoved'), 'notifysuccess');
echo $OUTPUT->continue_button("smanage.php");
echo $OUTPUT->footer();
```

$OUTPUT->header() outputs the header in HTML content, set by $PAGE. $OUTPUT->notification() function is used as notify event. the second argument, 'notifysuccess' will print the notification string in green font color, and 'notifyproblem' will print it in red font color. Adding a continue button is as easy as adding this line $OUTPUT->continue_button(). And finally footer is printed.

2.2.5 Moodlelib API

It is the central library file of miscellaneous general-purpose Moodle functions. It contains the functions like *required_param, optional_param* which are responsible for parameter passing while navigating from one page to other.

require_login this function checks that the current user is logged in and has the required privileges, which is defined in this library file.

get_string this function returns localised string specified by $identifier.

In this way, functions defined in this file are used for user preferences, time, login, plugins and strings.

2.2.6 Gradebook API

This API allows to read and write from gradebook with interface for detailed grading information. The library of this API contains the *grade_grade* class which is mapped to grade_grade table in database and contains detailed information for different grade_items for users. *grade_item* is the class which maps grade item for the user. *grade_tree* is the class representing a complete tree of categories, grade_items and final grades, it generates and stores the hierarchical array of all grade_category and grade_item from course id that is for a course.

2.2.7 File API

This API is for managing all the files stored by Moodle. Storage of files is conceptually in file areas, where file area is described by context id, full component name, itemid. Serving files to user is handled by file serving script, which is pluginfile.php. *moodle_url::make_pluginfile_url()* function is used to generate url for file serving, which takes the arguments of contextid, component, file_area, itemid and filepath with filename. File API is also responsible for managing file upload by user through the use of filepicker, filemanager and editor form elements.

2.2.8 Form API

For submitting data in Moodle, it requires to have web forms, which is managed by this

API. It supports all HTML elements with improved accessibility. formslib.php - library of classes for creating forms in Moodle, which contains abstract class moodleform, which is the base class for all the forms. This library contains the functions like *set_data(), get_data()* which are manage the submitted data for further processing. Generally new form is created by extending moodleform class and override the definition to include form elements which are required. Basic form elements are *button*, *checkbox*, *radio*, *text*, *HTML*, etc. Also it has custom form elements like *date_time_selector, filepicker, filemanager, etc.*. All these form elements are defined in 'lib/form/', where for each element different php file is written.

2.3 Moodle Database

Moodle database consists of many tables of which some are core tables and the tables belonging to each plugin. The Moodle database structure is defined in *install.xml* files inside *db* folder in each plugin. *lib/db/install.xml* defines the tables used in Moodle core. Further the database abstraction layer is introduced in Moodle that leads to,

- Data Definition API
- Data Manipulation API

2.3.1 Data Definition API

The objective of this API is to have well defined set of functions to handle all the DB structure by executing the correct SQL statements required by underlying **R**etational **D**ata**b**ase **M**anagement **S**ystems (RDBMS) used, that are supported by Moodle. Database manager instance is responsible for all database structure modification in the process of installation and upgrade. It consists of functions like *install_from_xmldb_file*, *create_table*, *drop_table*, etc., which are generating SQL statements to execute in database used.

2.3.2 Data Manipulation API

The library of this API contains all data manipulation functions to interact with database, *moodle_database* is the main abstract class for the functions like *get_record*, *get_record_sql*, etc. All these functions are accessed using global $DB object. Functions of the form ****_sql* take SQL query as parameter and provide the appropriate data, while to have cross-db compatibility helper functions are used to build the SQL fragments which are of the type *sql_****, other functions use table names and conditions to formulate SQL query and then operate it on database. Thus for retrieving and updating data from database these data manipulation functions are used accordingly.

Chapter 3

Design And Implementation

3.1 Simple Multiple Choice Questions SubPlugin

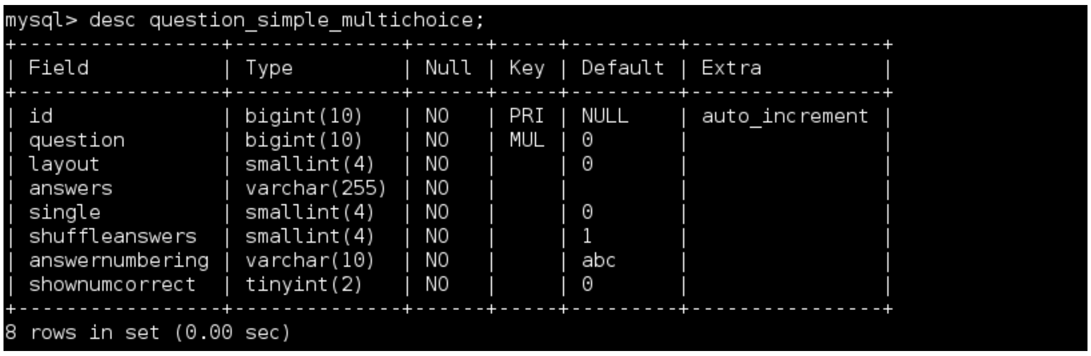
3.1.1 Introduction

The flexibility in Moodle is achieved by adding all types of Teacher-Student co-ordinated tasks in the weekly format like adding a quiz, notes, url link, assignment, etc. Further flexibility is achieved by giving more types of subplugins in each of these tasks. For example, quiz plugin contains the subplugins as yes-no type questions, multiple choice questions[14], essay[12], matching[13], etc. Now Moodle has become so advanced and vast over the time that there are a lot of options while creating such daily activities. Over the time, adding a multiple choice question has become so detailed that it became the cumbersome task to add questions.

So A new subplugin was required to add in the code, so as to accomplish the task of adding multichoice questions as easy as adding a question and putting some options with one answer correct, which serves the same purpose. Thus A simple multiple Choice Plugin was developed in this sense, and in order to make the things easier and sophisticated with less options while putting the questions.

3.1.2 Implementation

This is question type plugin in Moodle, basic structure of this plugin type was same as multichoice question type [11], [14] plugin already present in the question/type/ folder. Basically currently available multichoice question type form contains feedback fields for almost all the form elements like question text, each choice option and extra filed of combined feedback. In new plugin type of question, these feedback fields are removed and single choice correct option is kept, whereas multiple answer choices can be correct is available in existing



```
mysql> desc question_simple_multichoice;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| id              | bigint(10)   | NO   | PRI | NULL    | auto_increment |
| question        | bigint(10)   | NO   | MUL | 0       |                |
| layout          | smallint(4)  | NO   |     | 0       |                |
| answers         | varchar(255) | NO   |     |         |                |
| single          | smallint(4)  | NO   |     | 0       |                |
| shuffleanswers  | smallint(4)  | NO   |     | 1       |                |
| answernumbering | varchar(10)  | NO   |     | abc     |                |
| shownumcorrect  | tinyint(2)   | NO   |     | 0       |                |
+-----------------+--------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

Figure 3.1: Simple Multichoice Database Table

multichoice question type.

Brief Description of Implementation :

- simple_multichoice folder is added in question/type folder.
- In this folder, edit_simple_multichoice_form.php is the file form defining the form for adding question.
- question.php file contains question definition classes and questiontype.php contains the extended class of question_type class defined in questionbasetype.php .
- ***D****atabase (db)* folder contains install.xml for table defined for simple multichoice question and *lang* folder [15] contains the strings defined for this question type.

Database (db) folder This folder contains the *install.xml* file, which has the question_simple_multicho
table defined in it. Fields included in it are

id : primary key of table,

question : foreign key referencing to question.id,

answers : comma separated list of question answers ids,

shuffleanswers : for whether the choices to be randomly shuffled or not.

answernumbering : type of answer numbering.

lang folder : Strings for component 'qytpe_simple_multichoice' and language 'en' .

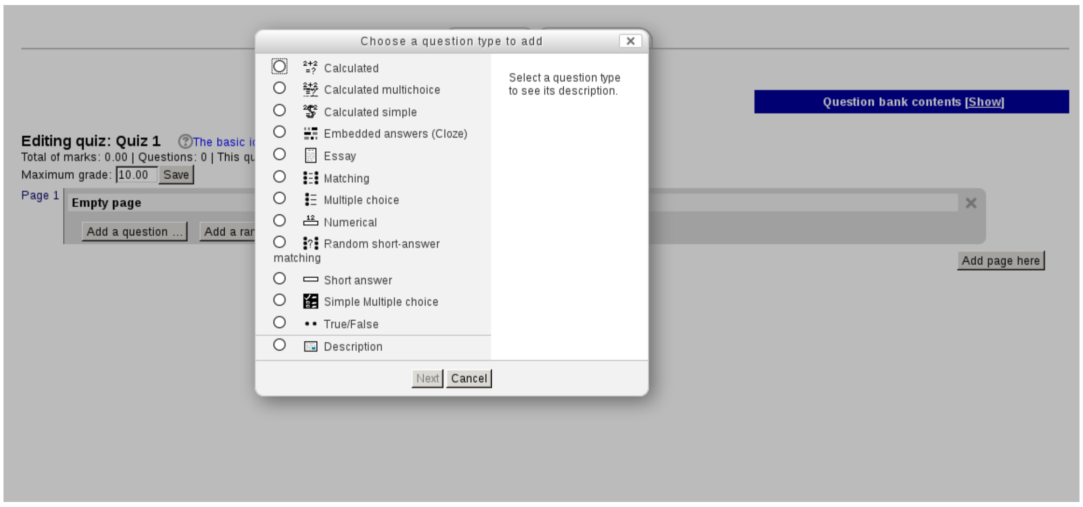
pix folder : icon file to be used in popup window of question type selection.

edit_simple_multichoice_form.php : It contains the class qtype_multichoice_ -edit_form which is extended from question_edit_form class defined in edit_question_form.php, the function definition_inner() is responsible for the form definition of question edit form, it contains the element to be included in question edit form for this particular question type.

Listing 3.1: Code snippet from edit_simple_multichoice_form.php[10]

```
protected function
get_per_answer_fields($mform,
   $label,
   $gradeoptions,
   &$repeatedoptions,
   &$answersoption) {
     $repeated = array();
     $repeated[] =
        $mform->
         createElement('editor',
                       'answer',
                       $label,
                       array('rows' => 1),
                       $this->editoroptions);
     $repeated[] =
        $mform->
         createElement('select',
                       'fraction',
                       get_string('grade'),
                       $gradeoptions);
    $repeatedoptions['answer']
                    ['type'] = PARAM_RAW;
    $answersoption = 'answers';
    return $repeated;
}
```

The function given above defines the per answer fields, which are answer editor, select element for answer, is true or false in terms of score that is 100% or 0%, while the feedback element is not available.



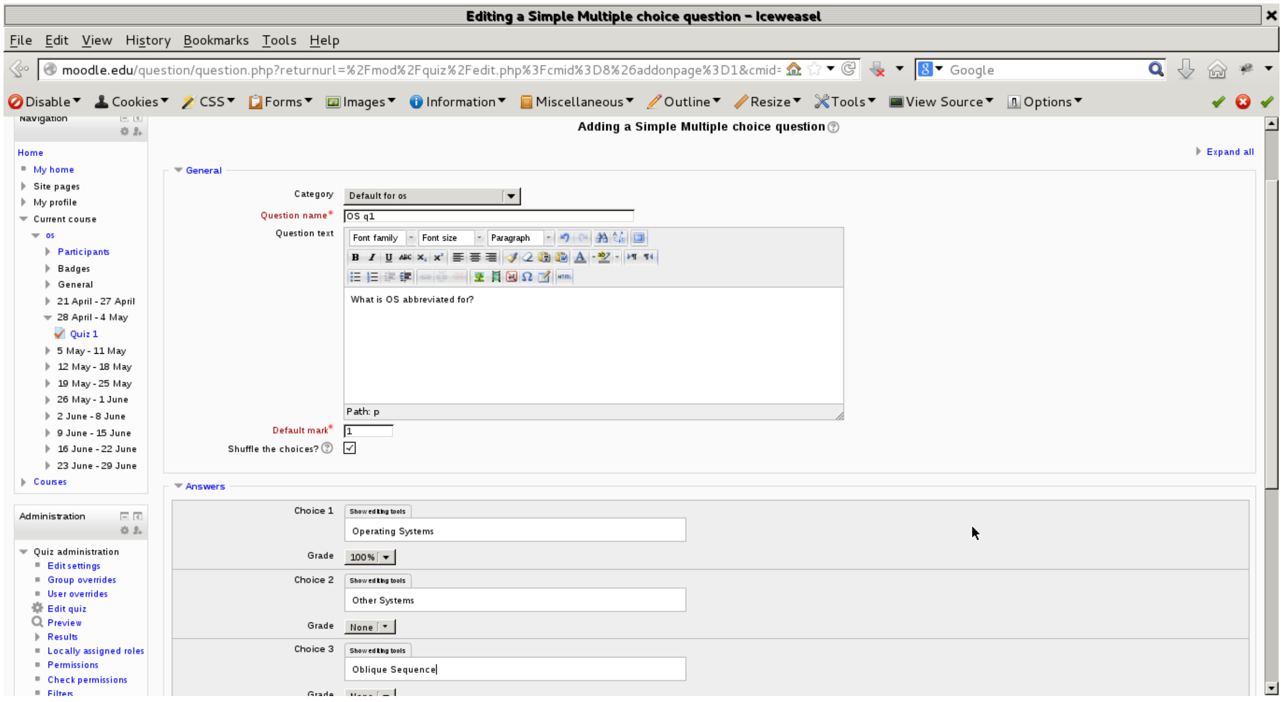
Editing quiz: Quiz 1 The basic i
Total of marks: 0.00 | Questions: 0 | This qu
Maximum grade: 10.00 Save
Page 1 Empty page

Choose a question type to add

- Calculated
- Calculated multichoice
- Calculated simple
- Embedded answers (Cloze)
- Essay
- Matching
- Multiple choice
- Numerical
- Random short-answer matching
- Short answer
- Simple Multiple choice
- True/False
- Description

Select a question type to see its description.

Figure 3.2: Popup window of question type selection after clicking on add question button



Editing a Simple Multiple choice question – Iceweasel

moodle.edu/question/question.php?returnurl=%2Fmod%2Fquiz%2Fedit.php%3Fcmid%3D8%26addonpage%3D1&cmid=

Navigation
- Home
- My home
- Site pages
- My profile
- Current course
  - os
    - Participants
    - Badges
    - General
    - 21 April - 27 April
    - 28 April - 4 May
      - Quiz 1
    - 5 May - 11 May
    - 12 May - 18 May
    - 19 May - 25 May
    - 26 May - 1 June
    - 2 June - 8 June
    - 9 June - 15 June
    - 16 June - 22 June
    - 23 June - 29 June
- Courses

Administration
- Quiz administration
  - Edit settings
  - Group overrides
  - User overrides
  - Edit quiz
  - Preview
  - Results
  - Locally assigned roles
  - Permissions
  - Check permissions
  - Filters

Adding a Simple Multiple choice question

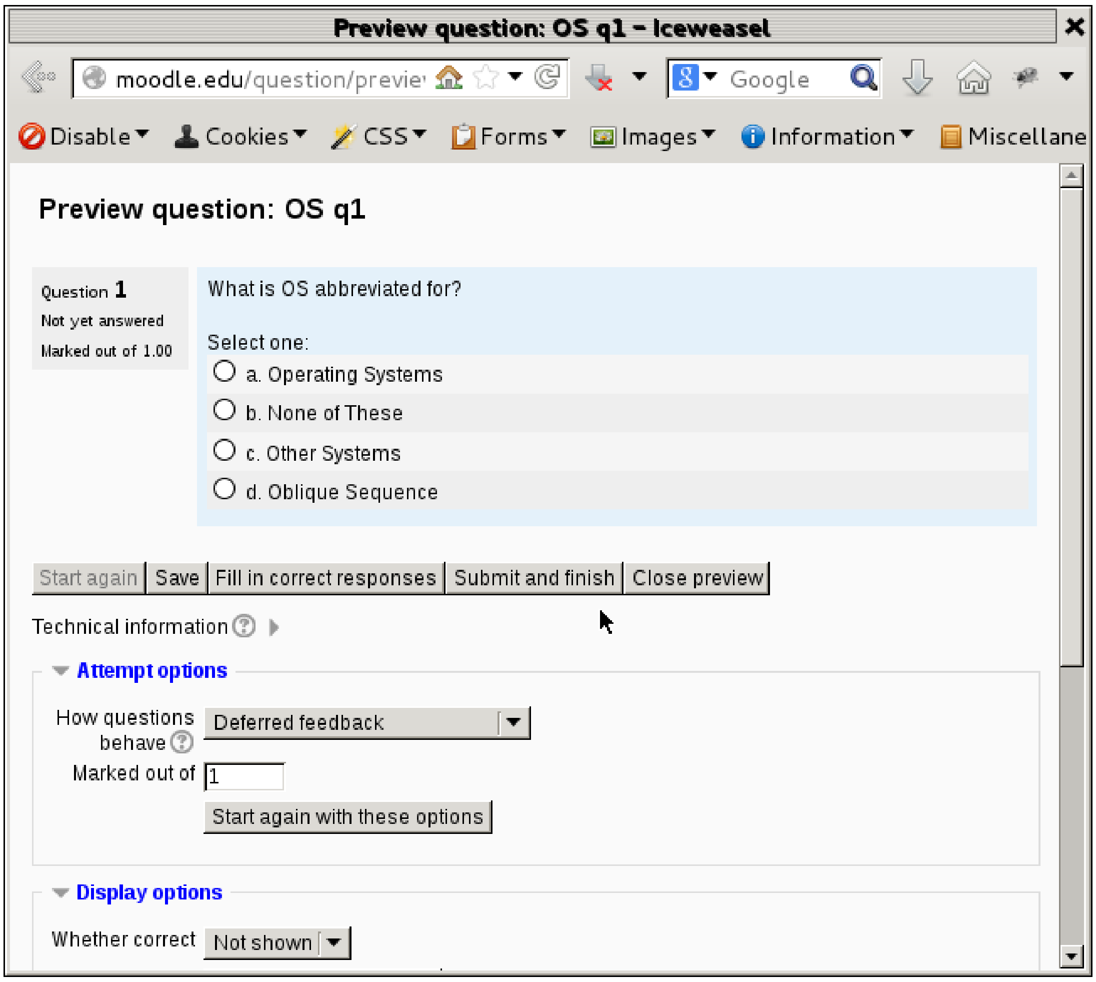
General

Category	Default for os
Question name*	OS q1
Question text	What is OS abbreviated for?
	Path: p
Default mark*	1
Shuffle the choices?	✓

Answers

Choice 1	Operating Systems
Grade	100%
Choice 2	Other Systems
Grade	None
Choice 3	Oblique Sequence
Grade	None

Figure 3.3: Adding Simple Multichoice Question page


Preview question: OS q1 - Iceweasel
moodle.edu/question/previe
Google
Disable
Cookies
CSS
Forms
Images
Information
Miscellane

Preview question: OS q1

Question 1
Not yet answered
Marked out of 1.00

What is OS abbreviated for?

Select one:
- a. Operating Systems
- b. None of These
- c. Other Systems
- d. Oblique Sequence

Attempt options

How questions behave: Deferred feedback

Marked out of: 1

Display options

Whether correct: Not shown

Figure 3.4: Preview of question after completely adding

3.2 Bulk Manage Courses

3.2.1 Introduction

The current scenario of managing the courses somewhat lacks the optimal methods to do the managing tasks easier. In the current context, the use of action buttons, i.e. refreshing the page on each click seems infeasible on slow connections for managers.

Rather than taking immediate action and reloading page, it is more preferred to store the actions and then in one click do all the selected operations. Though this looks easy, but in the current coding styles, as the single_button actions are implemented, it becomes hard to write our code and then merge with the current implementation of Moodle. But the current implementation looks so complete, to make that merge happen. So we created our own files added our action there and then did a small change in current code to complete this task.

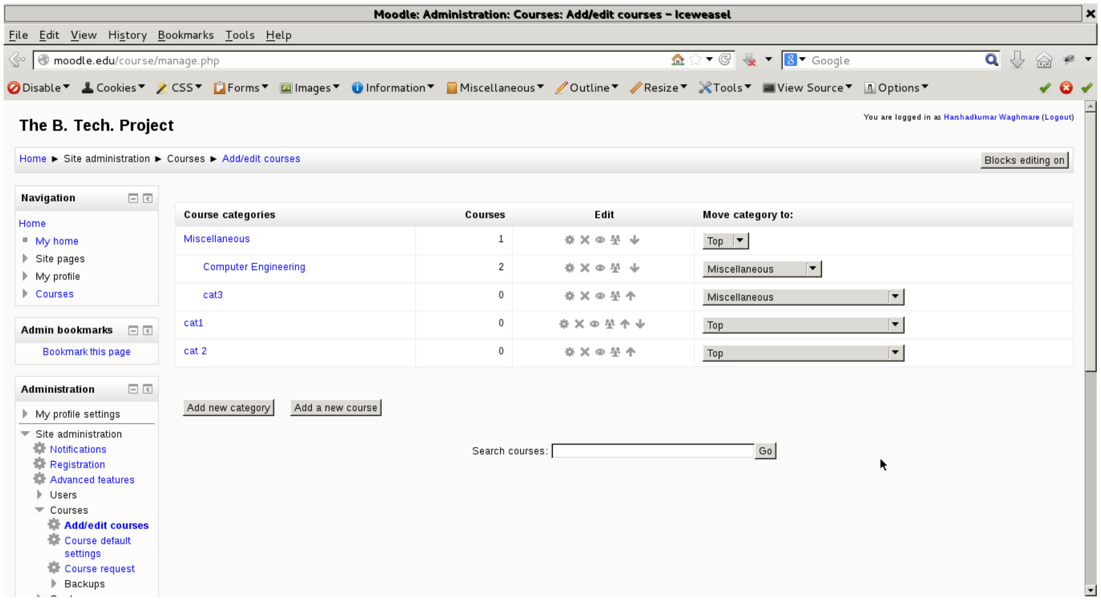
3.2.2 Implementation

The current file that supports management of courses is *manage.php* located at *moodle/-course/manage.php.* The single file is written to manage course categories as well as courses in those categories. So, first the *manage.php* page will show the list of categories in a html_table with the attributes as *course_categories, No. of courses, Edit actions*, and a dropbox menu of *move to courses.* The edit options here are *settings, delete, hide or show, cohorts, moveup or movedown.* Technically each of these edit options are the action button, whose on-click action to to directly take an action and reload the page with acted action. The *move category to* dropdown menu is also spontaneously responding menu, i.e. as soon as the category is selected from the drop down menu, the action of moving the current category to new selected category takes place. And this table is not a form, i.e. it is just a table shown in the HTML page. Now here is the scope of improvement in moodle, we can always store the desired actions and with just one *submit* button it should take action on all the stored choices.

To achieve this goal, *smanage.php* is created, which now contains a form, including above table and a submit button. This submit button takes an action by redirecting to *actioncat.php*, the file in which the actions to be taken on coming input are written. The form in smanage.php has now three more fields of checkbuttons to input from user. The three actions now can be done in groups because of using these checkboxes. The three fields are *Move, Delete, and Hide/Show.* Now just check the actions among the three for a category and get the action done. Now the course is recorded from dropdown menu of 'move category to' and a checkbox 'Move' need to be selected to ensure the action of moving the courses. Delete and Hide checkboxes are regular checkboxes and takes the actions on selecting. Hide is a toggle button with show. Thus a course initially shown will be hidden and vice versa.

Listing 3.2: Code snippet for hiding a category[18]

```
$strhidcatparam = 'chid'.$category->id;
$hidecatparam =
   optional_param($strhidcatparam, 0,
                  PARAM_INT);
   if ($hidecatparam !== 0) {
     if ($category->visible == 1) {
       $hidecat = $category->id;
       $showcat = 0;
     }
     else {
       $showcat = $category->id;
       $hidecat = 0;
     }
     echo $OUTPUT->heading($category->name);
     if ($hidecat and confirm_sesskey()) {
       $cattohide = coursecat::
                    get($hidecat);
    require_capability(
            'moodle/category:manage',
            get_category_or_system_context(
                $cattohide->parent));
            $cattohide->hide();
    echo $OUTPUT->
       notification(get_string('hidecat',
                                  '',
                                  $category->name),
                                 'notifysuccess');
     } else if ($showcat and confirm_sesskey()) {
       $cattoshow = coursecat::get($showcat);
       require_capability('moodle/category:manage',
               get_category_or_system_context(
                    $cattoshow->parent));
       $cattoshow->show();
       echo $OUTPUT->notification(
                get_string('showcat',
                        '', $category->name),
                        'notifysuccess');
        }
    }
```


Moodle: Administration: Courses: Add/edit courses - Iceweasel
File Edit View History Bookmarks Tools Help
moodle.edu/course/manage.php
Google
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

The B. Tech. Project

You are logged in as Harshadkumar Waghmare (Logout)

Home ► Site administration ► Courses ► Add/edit courses

Blocks editing on

Navigation
- Home
- My home
- Site pages
- My profile
- Courses

Admin bookmarks
- Bookmark this page

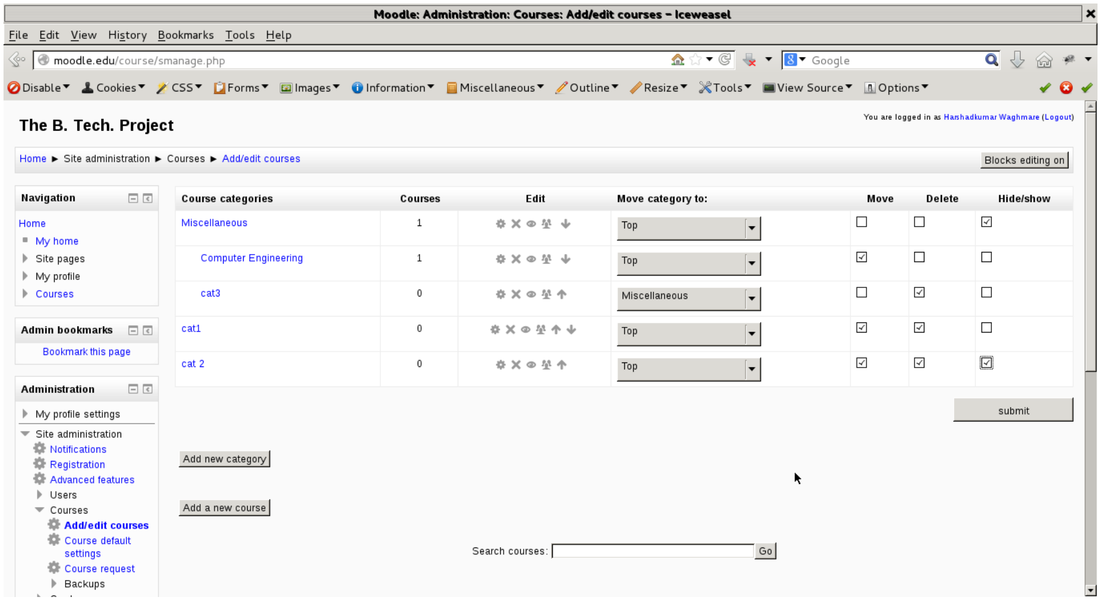
Administration
- My profile settings
- Site administration
  - Notifications
  - Registration
  - Advanced features
  - Users
  - Courses
    - Add/edit courses
    - Course default settings
    - Course request
    - Backups

Course categories	Courses	Edit	Move category to:
Miscellaneous	1		Top
Computer Engineering	2		Miscellaneous
cat3	0		Miscellaneous
cat1	0		Top
cat 2	0		Top

Add new category Add a new course

Search courses: Go

Figure 3.5: The current page for managing the courses, moodle/course/manage.php


Moodle: Administration: Courses: Add/edit courses - Iceweasel
File Edit View History Bookmarks Tools Help
moodle.edu/course/smanage.php
Google
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

The B. Tech. Project

You are logged in as Harshadkumar Waghmare (Logout)

Home ► Site administration ► Courses ► Add/edit courses

Blocks editing on

Navigation
- Home
- My home
- Site pages
- My profile
- Courses

Admin bookmarks
- Bookmark this page

Administration
- My profile settings
- Site administration
  - Notifications
  - Registration
  - Advanced features
  - Users
  - Courses
    - Add/edit courses
    - Course default settings
    - Course request
    - Backups

Course categories	Courses	Edit	Move category to:	Move	Delete	Hide/show
Miscellaneous	1		Top	☐	☐	✓
Computer Engineering	1		Top	✓	☐	☐
cat3	0		Miscellaneous	☐	✓	☐
cat1	0		Top	✓	✓	☐
cat 2	0		Top	✓	✓	✓

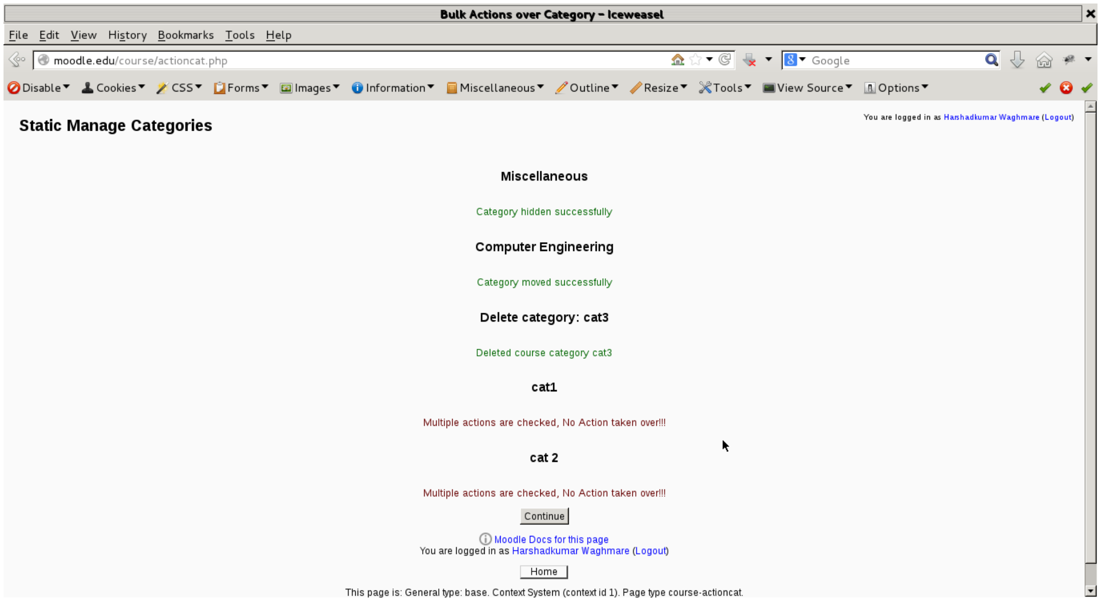
submit

Add new category

Add a new course

Search courses: Go

Figure 3.6: New file for doing bulk operations for categories, moodle/course/smanage.php



Bulk Actions over Category - Iceweasel

moodle.edu/course/actioncat.php

Static Manage Categories

You are logged in as Harshadkumar Waghmare (Logout)

Miscellaneous

Category hidden successfully

Computer Engineering

Category moved successfully

Delete category: cat3

Deleted course category cat3

cat1

Multiple actions are checked, No Action taken over!!!

cat 2

Multiple actions are checked, No Action taken over!!!

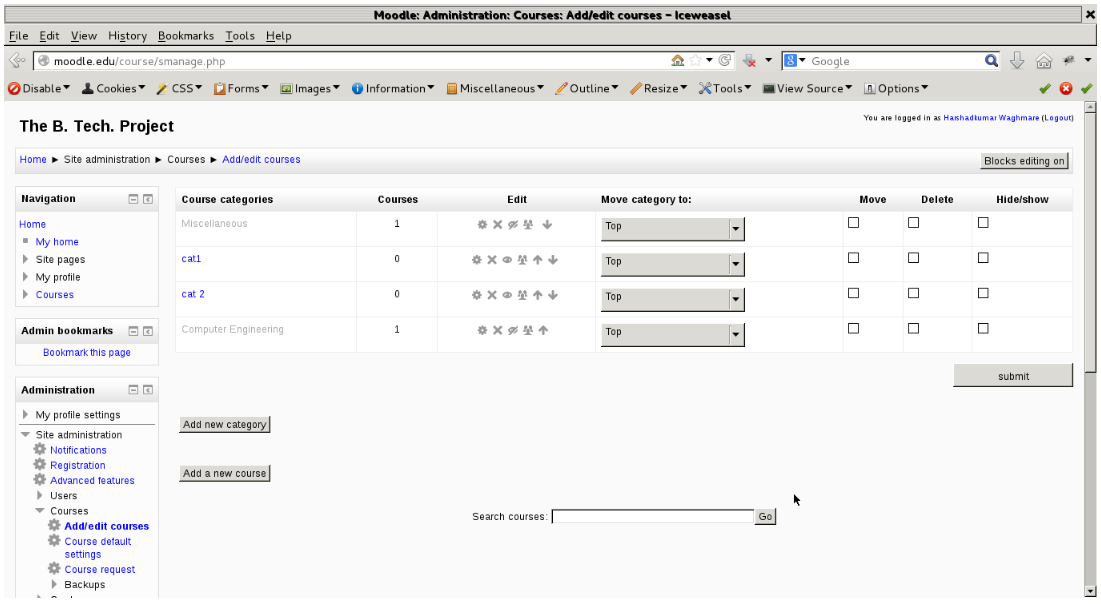
Continue

Moodle Docs for this page
You are logged in as Harshadkumar Waghmare (Logout)

Home

This page is: General type: base. Context System (context id 1). Page type course-actioncat.

Figure 3.7: Summary of actions taken place in bulk for the categories, actioncat.php



Moodle: Administration: Courses: Add/edit courses - Iceweasel

moodle.edu/course/smanage.php

The B. Tech. Project

You are logged in as Harshadkumar Waghmare (Logout)

Home ► Site administration ► Courses ► Add/edit courses

Blocks editing on

Course categories	Courses	Edit	Move category to:	Move	Delete	Hide/show
Miscellaneous	1		Top	☐	☐	☐
cat1	0		Top	☐	☐	☐
cat 2	0		Top	☐	☐	☐
Computer Engineering	1		Top	☐	☐	☐

submit

Add new category

Add a new course

Search courses: Go

Figure 3.8: Categories after the action, by actioncat.php, on page smanage.php

If more than one checkboxes are checked, then no action will takes place and the result page *actioncat.php* will show the summary as *'No action is being taken, selected more than one input'*. A continue button is added before the footer in actioncat.php, which redirects to the our category management form. The actioncat.php page also gives the summary of the actions taken place. These are the only actions needed to be taken over for categories.

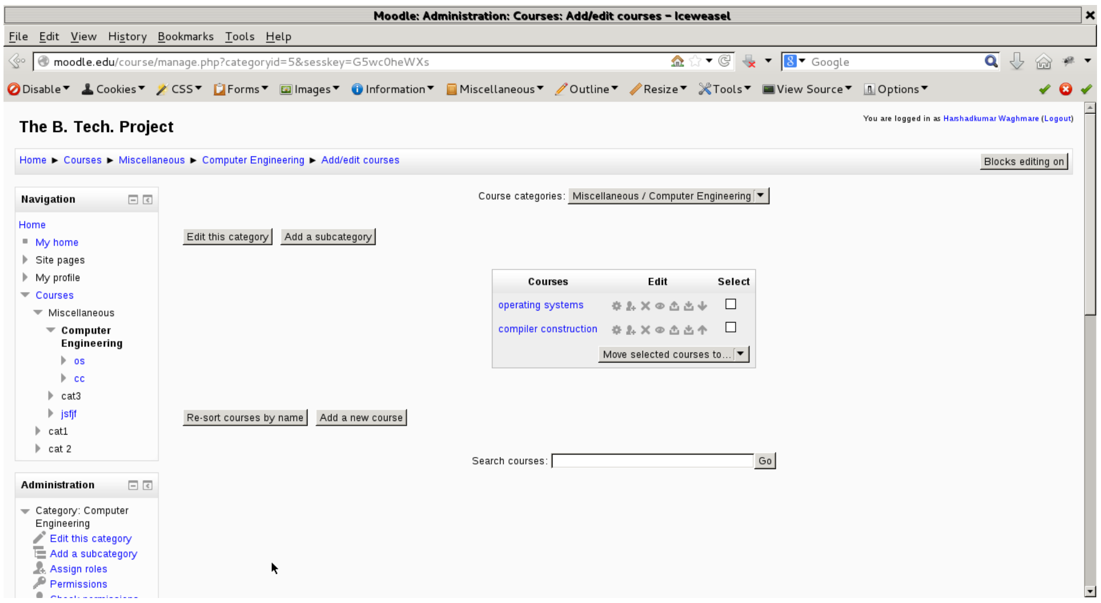
In manage.php, if a category is selected, it'll be redirected to the same page but with varied arguments, and thus will now show the list of courses in that category in the table with three columns as coursename, edit using various single_action buttons, then there is a select menu, selection of courses to move to other category. The Edit actions are settings, Enrolled users in that course, delete, hide, backup, restore. The most common actions that are required to change are delete and hide the courses. Now on this page, the table is enclosed in the form. Thus this form has an action to redirect to the same page and passing the checked arguments to the same page with a moveto category from dropdown menu. The dropdown menu is actually a *autosubmit* button, which takes action as well as it is clicked. Thus we can conclude that only one action can be takes place at a time.

Now as there is already a form with the specified option, it is not feasible to give it more than one actions for the current instance. So a new form on the new page was created, with some static options to store the actions from users about the courses. The staticoperations page was created, which includes the form with three fields of course name, delete menu, hide menu. Hide menu is a toggle button between hide and show courses. Now the actions are stored in the form variable, and are then passed to take actions. The submit button is added to submit the values of form and transfer these values to a new actionurl, *action.php*, which stores the various actions to be acted upon. This page also shows the summary of action that took place after the actions.

Listing 3.3: Code snippet for deleting a course[17]

```
if ($dcourseid !== 0) {
  if (!can_delete_course($course->id)) {
    echo 'course'.$course->fullname.
         ' was not deleted due to restricted
           permissions';
         continue;
  }
  if (!confirm_sesskey()) {
    print_error('confirmsesskeybad', 'error');
  }
```

```
    add_to_log(SITEID,
               "course", "delete",
               "view.php?id=$course->id",
               "$course->fullname
               (ID $course->id)");
    echo $OUTPUT->heading($course->fullname);
    delete_course($course);
    fix_course_sortorder();
}
```


Moodle: Administration: Courses: Add/edit courses - Iceweasel
File Edit View History Bookmarks Tools Help
moodle.edu/course/manage.php?categoryid=5&sesskey=G5wc0heWXs
Google
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Blocks editing on
Edit this category
Add a subcategory
Move selected courses to...
Re-sort courses by name
Add a new course
Go

The B. Tech. Project

You are logged in as Harshadkumar Waghmare (Logout)

Home ▸ Courses ▸ Miscellaneous ▸ Computer Engineering ▸ Add/edit courses

Navigation
- Home
  - My home
  - Site pages
  - My profile
  - Courses
    - Miscellaneous
      - Computer Engineering
        - os
        - cc
      - cat3
      - jsfjf
    - cat1
    - cat 2

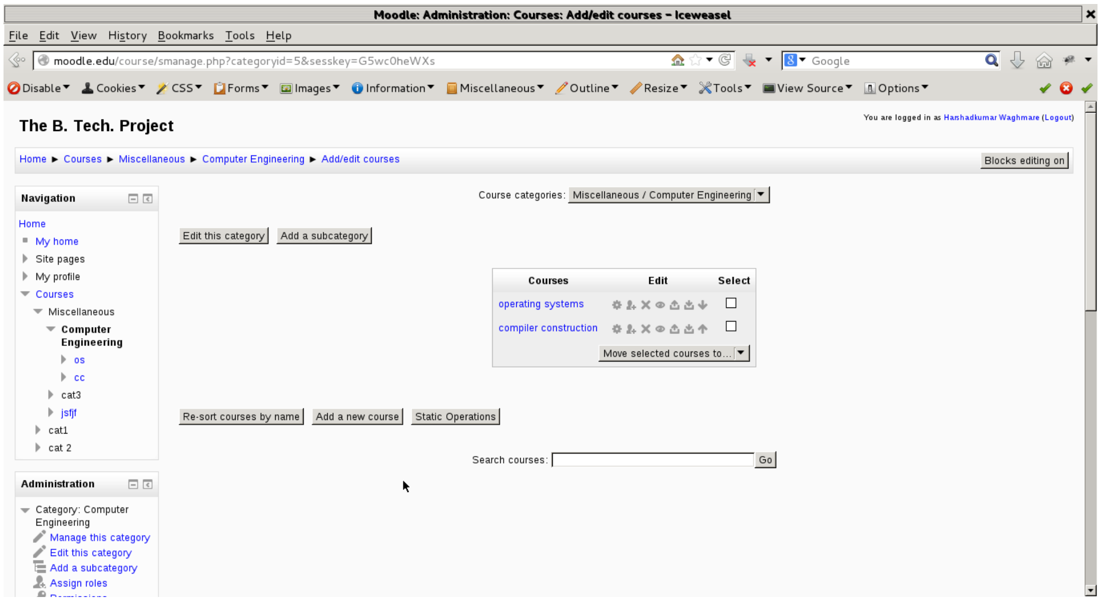
Administration
- Category: Computer Engineering
  - Edit this category
  - Add a subcategory
  - Assign roles
  - Permissions

Course categories: Miscellaneous / Computer Engineering

Courses	Edit	Select
operating systems		☐
compiler construction		☐

Search courses:

Figure 3.9: The current manage.php file, for managing courses


Moodle: Administration: Courses: Add/edit courses - Iceweasel
File Edit View History Bookmarks Tools Help
moodle.edu/course/smanage.php?categoryid=5&sesskey=G5wc0heWXs
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Blocks editing on
Edit this category
Add a subcategory
Move selected courses to...
Re-sort courses by name
Add a new course
Static Operations
Go

The B. Tech. Project

You are logged in as Harshadkumar Waghmare (Logout)

Home ► Courses ► Miscellaneous ► Computer Engineering ► Add/edit courses

Navigation
- Home
- My home
- Site pages
- My profile
- Courses
  - Miscellaneous
    - Computer Engineering
      - os
      - cc
    - cat3
    - jsfjf
  - cat1
  - cat 2

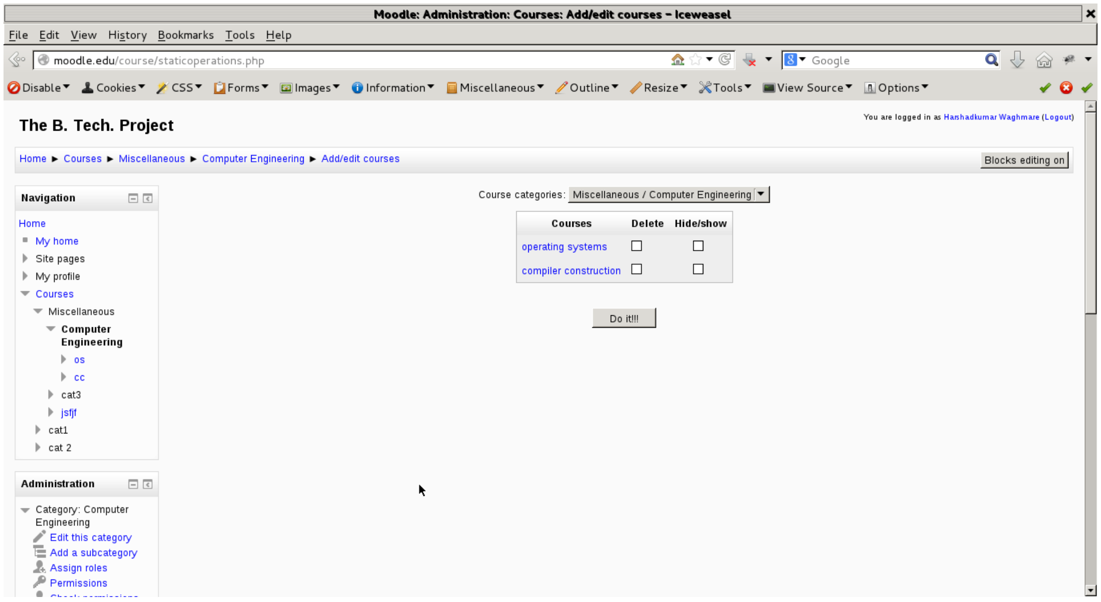
Administration
- Category: Computer Engineering
  - Manage this category
  - Edit this category
  - Add a subcategory
  - Assign roles

Course categories: Miscellaneous / Computer Engineering

Courses	Edit	Select
operating systems		☐
compiler construction		☐

Search courses:

Figure 3.10: smanage.php, for courses in a selected category


Moodle: Administration: Courses: Add/edit courses - Iceweasel
File Edit View History Bookmarks Tools Help
moodle.edu/course/staticoperations.php
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Blocks editing on
Do it!!!

The B. Tech. Project

You are logged in as Harshadkumar Waghmare (Logout)

Home ► Courses ► Miscellaneous ► Computer Engineering ► Add/edit courses

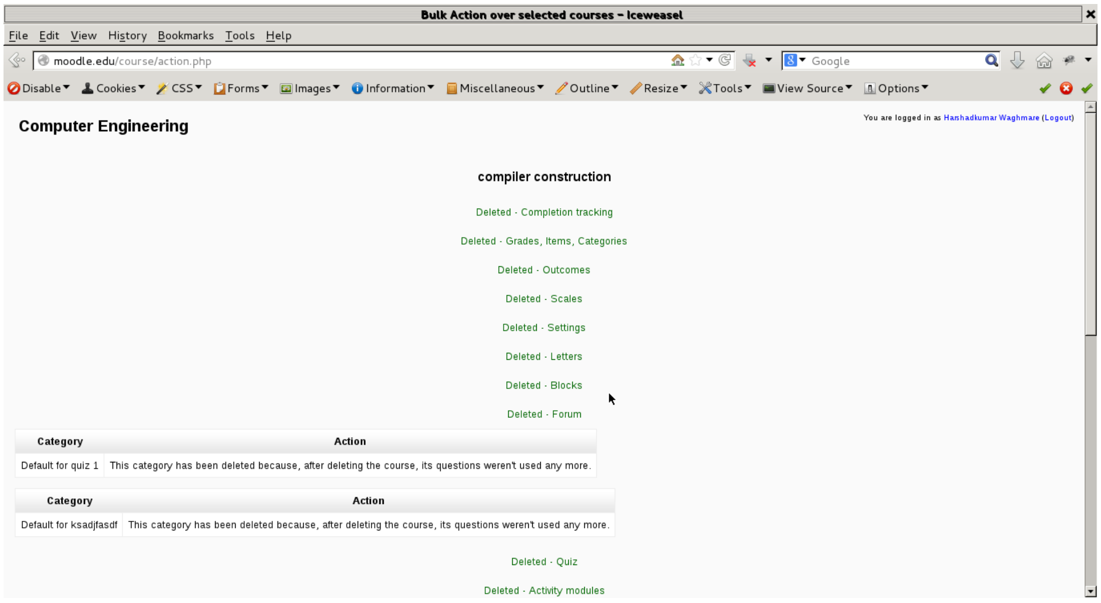
Navigation
- Home
- My home
- Site pages
- My profile
- Courses
  - Miscellaneous
    - Computer Engineering
      - os
      - cc
    - cat3
    - jsfjf
  - cat1
  - cat 2

Administration
- Category: Computer Engineering
  - Edit this category
  - Add a subcategory
  - Assign roles
  - Permissions

Course categories: Miscellaneous / Computer Engineering

Courses	Delete	Hide/show
operating systems	☐	☐
compiler construction	☐	☐

Figure 3.11: staticoperation.php, the page dedicated to make bulk actions over courses in selected category


Bulk Action over selected courses – Iceweasel
File Edit View History Bookmarks Tools Help
moodle.edu/course/action.php
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

Computer Engineering

You are logged in as Harshadkumar Waghmare (Logout)

compiler construction

Deleted - Completion tracking

Deleted - Grades, Items, Categories

Deleted - Outcomes

Deleted - Scales

Deleted - Settings

Deleted - Letters

Deleted - Blocks

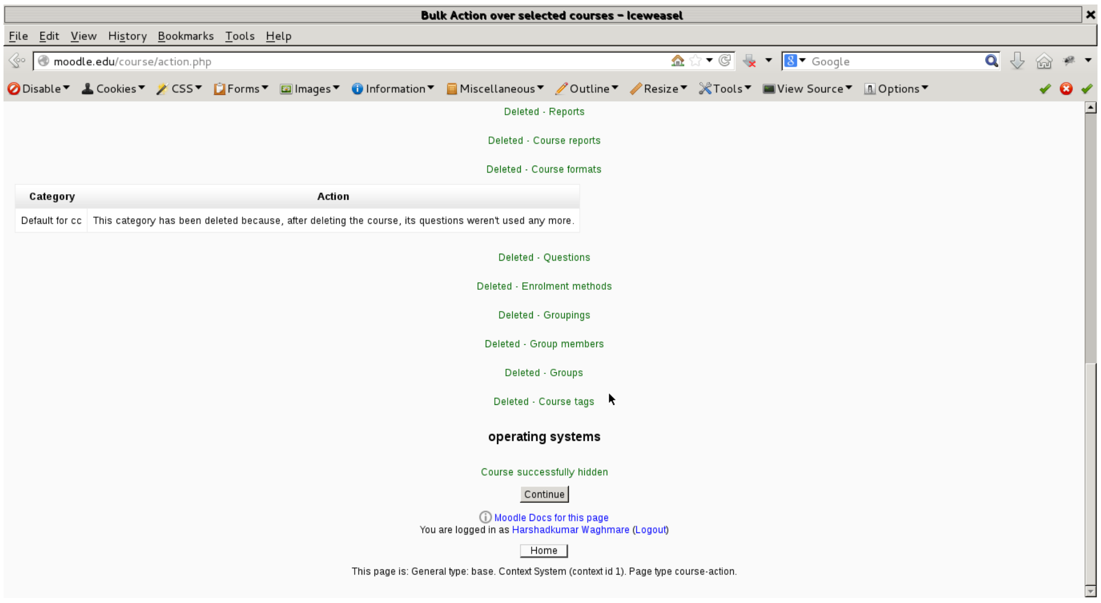
Deleted - Forum

Category	Action
Default for quiz 1	This category has been deleted because, after deleting the course, its questions weren't used any more.

Category	Action
Default for ksadjfasdf	This category has been deleted because, after deleting the course, its questions weren't used any more.

Deleted - Quiz

Deleted - Activity modules


Bulk Action over selected courses – Iceweasel
File Edit View History Bookmarks Tools Help
moodle.edu/course/action.php
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

Deleted - Reports

Deleted - Course reports

Deleted - Course formats

Category	Action
Default for cc	This category has been deleted because, after deleting the course, its questions weren't used any more.

Deleted - Questions

Deleted - Enrolment methods

Deleted - Groupings

Deleted - Group members

Deleted - Groups

Deleted - Course tags

operating systems

Course successfully hidden

Continue

Moodle Docs for this page

You are logged in as Harshadkumar Waghmare (Logout)

Home

This page is: General type: base. Context System (context id 1). Page type course-action.

Figure 3.12: action.php, the summary page of actions applied to the courses

3.3 Grader Report Enhancement for user files

3.3.1 Introduction

Moodle supports different grade reports like User Report, Overview Report and Grader Report[23]. Grader Report is used in moodle for user activity grading interface through which a teacher can view and manipulate grades of gradable course activities such assignments, quizzes etc. It provides features to view and edit grades for different gradable activities of a course.

Instead of having all these features grader report lacks behind in some operations. Teachers or course administrators are required to navigate to various pages in order to grade a student submission and to check files submitted by students that require grading. To view the student's submission teacher have to redirect to download submission page, which slows down their grading process. Bulk grading is not possible with this approach as teacher will have to navigate a lot throughout.

Grader Report Enhancement is designed to resolve this issue and provides an efficient grading interface for teacher/admin which handles all gradable activities and its submissions. It provides an in-line link for students submissions for particular activity in grader report so that teachers need not navigate different pages and can view submitted files.

3.3.2 Description

- Problem:
    1. To give grade to a specific assignment teacher has to navigate through different web pages. First go to that assignment page, then click on View/Grade all submissions, click on Grade Edit Icon from grading table and then it navigates to a different page which views Submission Status page of a particular user, on this page teacher can view submitted files and can grant grade for that assignment.

       In this way, teacher has to grade all the assignments of users individually. This slows down the process of grading and makes it very lengthy.
    2. There is another option to grade assignments through Grader Report, here teacher can grade to assignment submissions of all users of a course but to view/get submitted files of particular assignment teacher has to navigate through different pages or teacher has to follow same procedure as stated above.
- Suggested Solution: To resolve this issue in the grader report an In-line Download link of files of respective assignment submissions should be added in Grader Report Grading

Table.[29]

3.3.3 Implementation

To implement this task we have modified get_right_rows() function from file moodle/grade/report/grader/lib.php.

moodle/grade/report/grader/lib.php : Function get_right_rows() Builds and returns the rows that will make up the right part of the grader report.

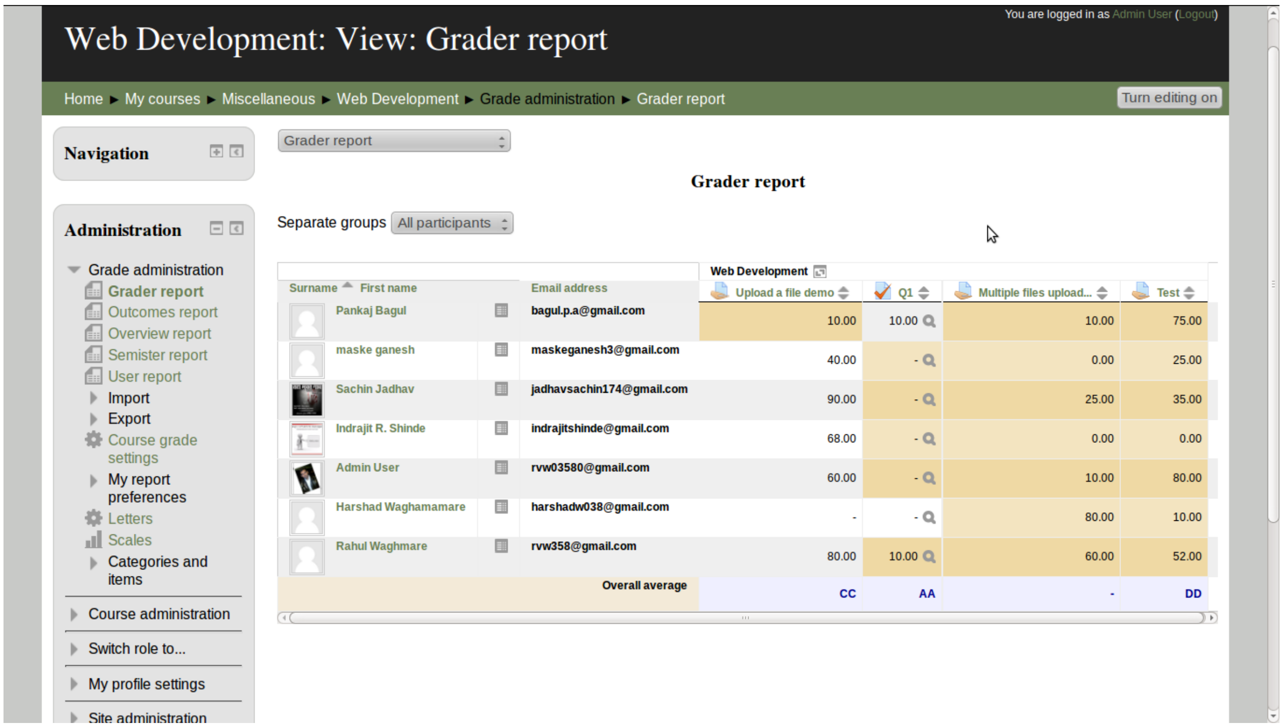
We need course module object, context of module to create object of class assign (Assignment) which is defined in moodle/mod/assign/locallib.php through which we can access different parameters of assign class. We get object of course module by using function get_coursemodule- _from_instance() and context from context_module:: instance()[19]. After getting object of assignment we get user submissions by calling function get_user_submissions() (defined in moodle/mod/assign/locallib.php).

File download : Moodle saves its all submitted files in a different directory other than web directory for security purposes. So to get location of stored file we can call function get_file_storage() (defined in moodle/lib.php). Now to get all submitted files we call function get- _area_files().[20]

Now to create in-line download link of submitted file we used function moodle_url::make_pluginfile_u in moodle/lib.php), the required parameter for this function are file component, file area, submission id, context id, file path, file name and forcedownload. Display created filelink in the item-cell text. When we click on the links on file name displayed in grading table cell a popup window pops on screen having options of download and view file.

Listing 3.4: Getting files and making url for each obtained file[21]

```
$files = $fs->get_area_files(
        $context->id,
        'assignsubmission_file',
        'submission_files',
        $submission->id);
foreach($files as $file) {
  $filelink = moodle_url::
     make_pluginfile_url($context->id,
                        $file->get_component(),
                        $file->get_filearea(),
                        $submission->id,
                        $file->get_filepath(),
                        $file->get_filename(),
                        true);
```



You are logged in as Admin User (Logout)

Web Development: View: Grader report

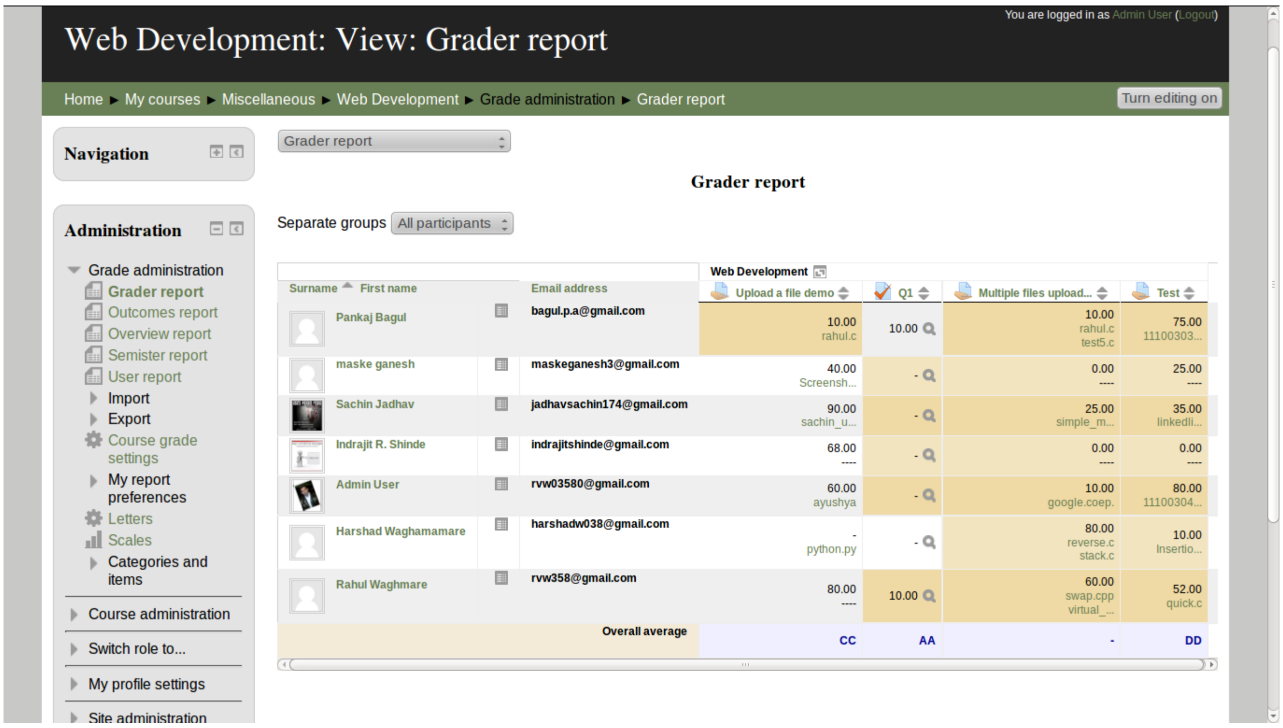
Home ► My courses ► Miscellaneous ► Web Development ► Grade administration ► Grader report

Grader report

Separate groups: All participants

Surname ▲ First name	Email address	Web Development: Upload a file demo	Q1	Multiple files upload...	Test
Pankaj Bagul	bagul.p.a@gmail.com	10.00	10.00	10.00	75.00
maske ganesh	maskeganesh3@gmail.com	40.00	-	0.00	25.00
Sachin Jadhav	jadhavsachin174@gmail.com	90.00	-	25.00	35.00
Indrajit R. Shinde	indrajitshinde@gmail.com	68.00	-	0.00	0.00
Admin User	rvw03580@gmail.com	60.00	-	10.00	80.00
Harshad Waghamamare	harshadw038@gmail.com	-	-	80.00	10.00
Rahul Waghmare	rvw358@gmail.com	80.00	10.00	60.00	52.00
	Overall average	CC	AA	-	DD

Figure 3.13: Before Grader Report Enhancement



You are logged in as Admin User (Logout)

Web Development: View: Grader report

Home ▸ My courses ▸ Miscellaneous ▸ Web Development ▸ Grade administration ▸ Grader report

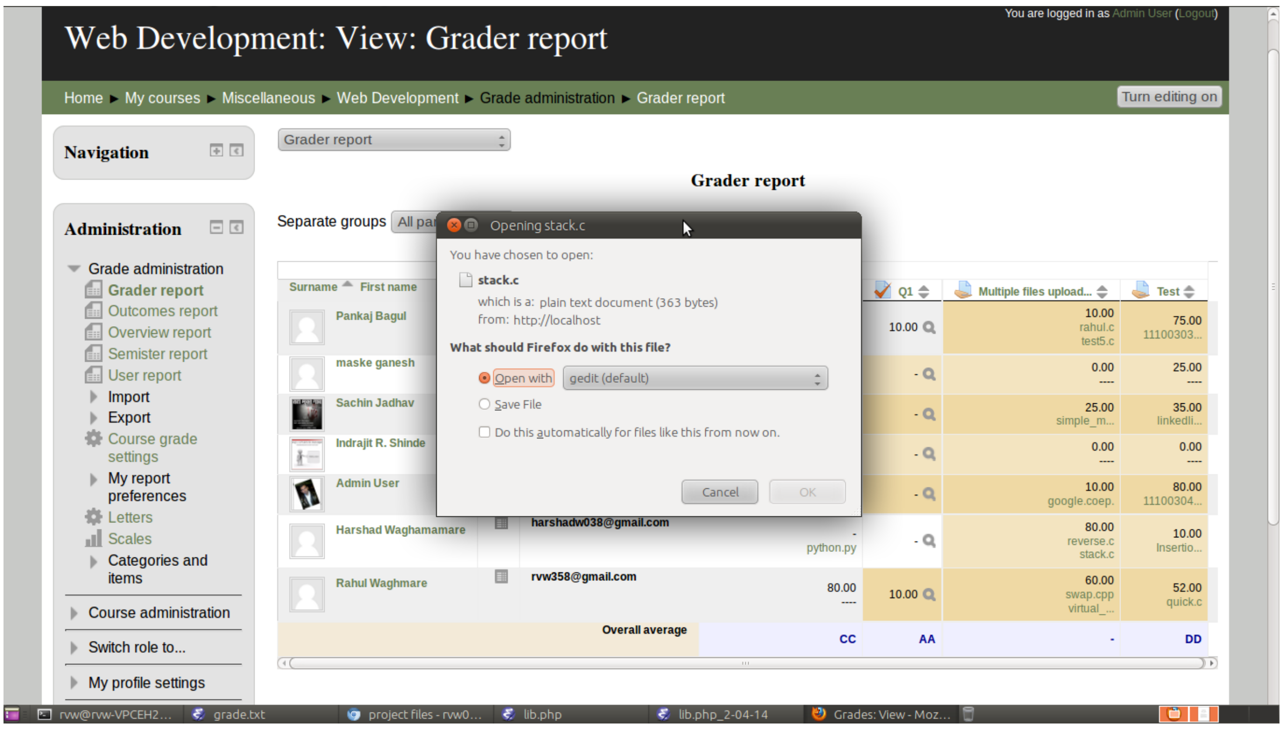
Turn editing on

Grader report

Separate groups: All participants

Surname / First name	Email address	Web Development: Upload a file demo	Q1	Multiple files upload...	Test
Pankaj Bagul	bagul.p.a@gmail.com	10.00 rahul.c	10.00	10.00 rahul.c test5.c	75.00 11100303...
maske ganesh	maskeganesh3@gmail.com	40.00 Screensh...	-	0.00 ----	25.00 ----
Sachin Jadhav	jadhavsachin174@gmail.com	90.00 sachin_u...	-	25.00 simple_m...	35.00 linkedli...
Indrajit R. Shinde	indrajitshinde@gmail.com	68.00 ----	-	0.00 ----	0.00 ----
Admin User	rvw03580@gmail.com	60.00 ayushya	-	10.00 google.coep.	80.00 11100304...
Harshad Waghamamare	harshadw038@gmail.com	- python.py	-	80.00 reverse.c stack.c	10.00 Insertio...
Rahul Waghmare	rvw358@gmail.com	80.00 ----	10.00	60.00 swap.cpp virtual_...	52.00 quick.c
Overall average		CC	AA	-	DD

Figure 3.14: After Grader Report Enhancement



You are logged in as Admin User (Logout)

Web Development: View: Grader report

Home ▸ My courses ▸ Miscellaneous ▸ Web Development ▸ Grade administration ▸ Grader report

Turn editing on

Grader report

Separate groups: All pa

Opening stack.c

You have chosen to open:

stack.c

which is a: plain text document (363 bytes)

from: http://localhost

What should Firefox do with this file?

[✓ Open with] gedit (default)

Save File

Do this automatically for files like this from now on.

Cancel OK

Surname / First name	Email address	Q1	Multiple files upload...	Test
Pankaj Bagul		10.00	10.00 rahul.c test5.c	75.00 11100303...
maske ganesh		-	0.00 ----	25.00 ----
Sachin Jadhav		-	25.00 simple_m...	35.00 linkedli...
Indrajit R. Shinde		-	0.00 ----	0.00 ----
Admin User		-	10.00 google.coep.	80.00 11100304...
Harshad Waghamamare	harshadw038@gmail.com	-	80.00 reverse.c stack.c	10.00 Insertio...
Rahul Waghmare	rvw358@gmail.com	10.00	60.00 swap.cpp virtual_...	52.00 quick.c
Overall average		AA	-	DD

Figure 3.15: Pop window for Download

3.4 Grade Report for Student

3.4.1 Introduction

A user, who is student can see her current grades of a particular course in present Moodle environment, but is unable to see the grades of all registered courses on one page. To facilitate this we have added a grade report page for user, which has courses and final grades that user has got in those courses.

To implement this enhancement, we looked into gradelib.php, which is the library for gradebook API[22], where the grade_grade class is defined which contains the detailed information for different grade_items for users. By using Moodle PDF library[28] option of exporting this report is made available for the user in PDF format.

3.4.2 Implementation

Brief Description of Implementation :

- Created a new file for fetching courses and their grades at /user/profile/ named grade_report.php
- Created export.php file in same folder, to export this report.
- A link is added on profile page of user to grade_report file.

grade_report.php By using HTML output class, a simple form is written to get the semester type and semester start date as well as semester end date, by using function select_time(). By using enrol_get_all_users_courses() function, list of courses the user is enrolled is obtained.

Listing 3.5: Function enrol_get_all_users_courses()[24]

```
$usercourses = enrol_get_all_users_courses(
               $user->id, true, NULL,
               'visible DESC,
               sortorder ASC');
```

For each of the courses obtained the grade_item, which in turn gives the stored grade object for the user from grade_grade class , where the final grade for that user for that course is calculated.

Listing 3.6: Getting final grade for that course[24]

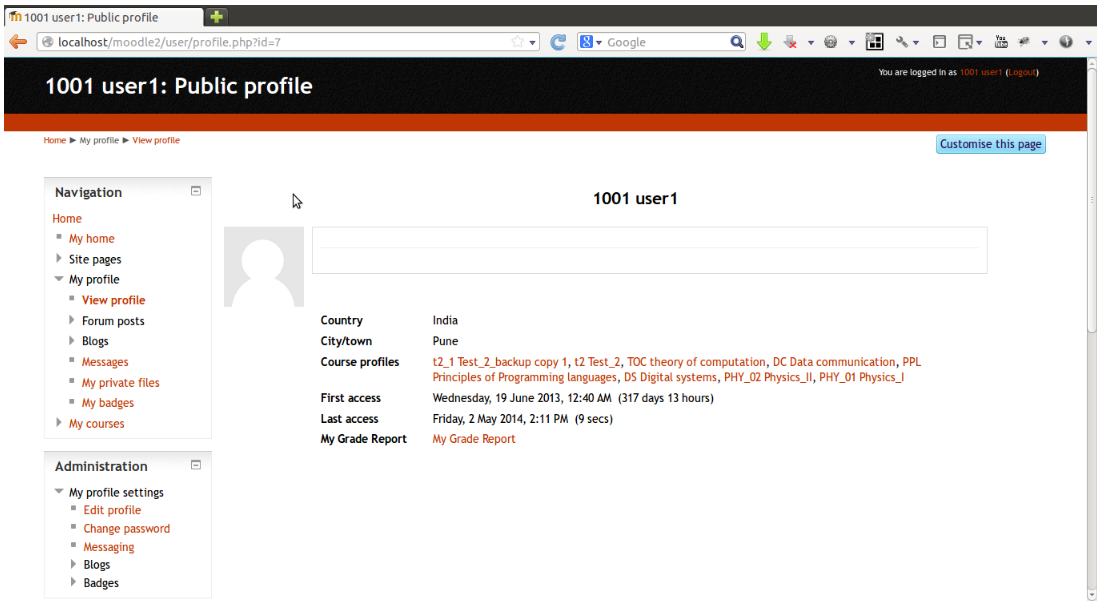
```
// Get usercourse grade_item
$usercourse_item = grade_item::
   fetch_course_item($usercourse->id);
// Get the stored grade
$usercourse_grade = new grade_grade(array(
   'itemid'=>$usercourse_item->id,
   'userid'=>$user->id));
$finalgrade = $usercourse_grade->finalgrade;
```

To show this content obtained, table element of HTML output class[26] is used.

export.php This file gets the HTML content of grade report and using Moodle PDF library this content is exported to PDF file as shown below :

Listing 3.7: Generate PDF file and download in browser[25]

```
//pdf file generation
$doc = new pdf;
$doc->setPrintHeader(false);
$doc->setPrintFooter(false);
$doc->AddPage();
$doc->writeHTML($exporthtml,
                true, false,
                false, false);
$downloadfilename = clean_filename(
                "user_grade_report.pdf");
$doc->Output($downloadfilename,'I');
```


1001 user1: Public profile
localhost/moodle2/user/profile.php?id=7
Google
Customise this page

You are logged in as 1001 user1 (Logout)

1001 user1: Public profile

Home ▶ My profile ▶ View profile

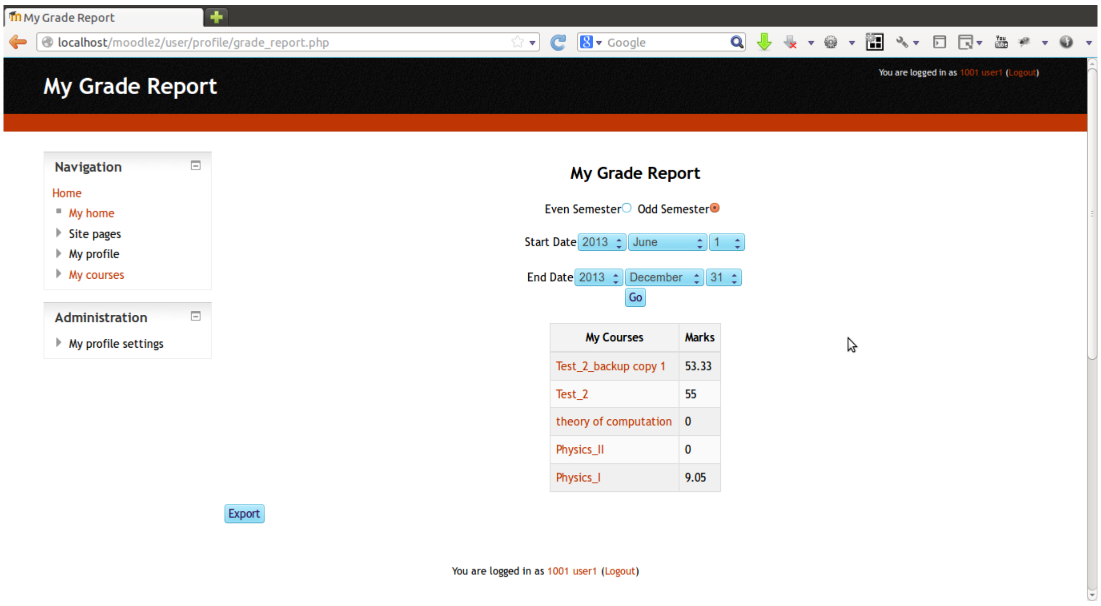
Navigation
- Home
  - My home
  - Site pages
  - My profile
    - View profile
    - Forum posts
    - Blogs
    - Messages
    - My private files
    - My badges
  - My courses

Administration
- My profile settings
  - Edit profile
  - Change password
  - Messaging
  - Blogs
  - Badges

1001 user1

Country	India
City/town	Pune
Course profiles	t2_1 Test_2_backup copy 1, t2 Test_2, TOC theory of computation, DC Data communication, PPL Principles of Programming languages, DS Digital systems, PHY_02 Physics_II, PHY_01 Physics_I
First access	Wednesday, 19 June 2013, 12:40 AM (317 days 13 hours)
Last access	Friday, 2 May 2014, 2:11 PM (9 secs)
My Grade Report	My Grade Report

Figure 3.16: Link for grade report on profile page:



You are logged in as 1001 user1 (Logout)

My Grade Report

Navigation
- Home
  - My home
  - Site pages
  - My profile
  - My courses

Administration
- My profile settings

My Grade Report

Even Semester ○ Odd Semester ●

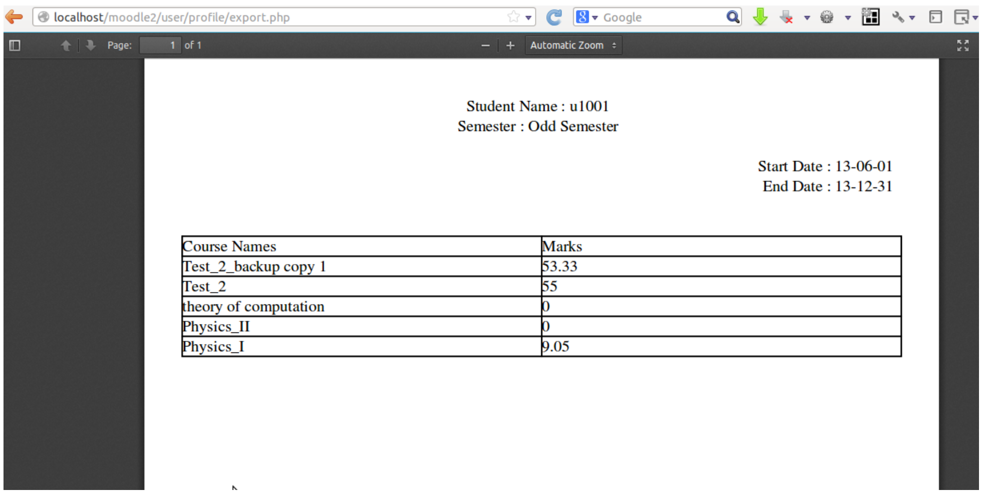
Start Date 2013 June 1

End Date 2013 December 31

My Courses	Marks
Test_2_backup copy 1	53.33
Test_2	55
theory of computation	0
Physics_II	0
Physics_I	9.05

You are logged in as 1001 user1 (Logout)

Figure 3.17: Grade report page :



Student Name : u1001
Semester : Odd Semester

Start Date : 13-06-01
End Date : 13-12-31

Course Names	Marks
Test_2_backup copy 1	53.33
Test_2	55
theory of computation	0
Physics_II	0
Physics_I	9.05

Figure 3.18: Pdf export of grade report :

Chapter 4

Conclusion and Future Scope

4.1 Conclusion

COEP is one of the institutes which uses Moodle. COEP is using Moodle for almost 5 years. Being that experienced, the Administrators had encountered a need for modification or new features, which will make COEP Moodle, more easy to use. The tasks which are completed, are the points of improvement that are suggested by Administrators of COEP Moodle. The need to make these implementations is because of the current implementation of these tasks does not meet the satisfaction of Administrators at COEP, who manage COEP Moodle. Making these improvements had saved a lot of their work and made COEP Moodle easy to use.

4.2 Future Scope

Apart from making improvements to Moodle by choosing tasks, there is a need to write an administration module. The Administration Module will consists of automation of all the academic administrative transactions like credit registration, grade allocation, etc. The Module will also be able to store student grades for all semesters, her individual information like name, address, Date of Birth etc. And the most important task of this module will be to manage all this information at the given instance. As Moodle is now course based system, to write this administrative module, semester based system has to develop, which seems a rather tedious job. This is the Future Scope of the Project.

Appendix A

Acronyms

Moodle	**M**odular **O**bject **O**riented **D**ynamic **L**earning **E**nvironment
COEP	**C**ollege **O**f **E**ngineering **P**une
IRC	**I**nternet **R**elay **C**hat
API	**A**pplication **P**rogramming **I**nterface
db	**D**ata**b**ase
langpack	**L**anguage **P**ackage
HTML	**H**yper**T**ext **M**arkup **L**anguage
XSS	**C**ross **S**ite **S**cripting
RDBMS	**R**etational **D**ata**b**ase **M**anagement **S**ystems
LDAP	**L**ightWeight **D**irectory **A**ccess **P**rotocol
LAMP	**L**inux **A**pache **M**ysql **P**HP

Appendix B

Installation of Moodle

The installation of moodle on Linux machine can be done in following steps, but before the setup, environment is to be fixed for moodle. Installation of **L**inux **A**pache **M**ysql **P**HP (LAMP) is necessary. As setup is done in Linux environment, installation of Apache2, mysql5, and php5 can be done using either apt manager, or could be install manually using dpkg or manual compilation is also possible.

Listing B.1: The apt commands to install all the required packages

```
//shell
$ sudo apt-get update
$ sudo apt-get install apache2 mysql-server php5
//some complementary packages required are
$ sudo apt-get install libapache2-mod-php5 \\
      libapache2-mod-auth-mysql php5-mysql
```

Now at this point the setup of environment is done.

- Download moodle code from `http://download.moodle.org/`. Latest stable version is recommended.
- Move the contents to your www-root directory, here /var/www/. Open Mysql, and create a database for storing moodle tables.
- Open a browser and open the moodle/index.php file, which should redirect you to install.php script, and your installation should start.
- Follow the guidelines given to complete the setup.

Once the setup is complete, you are allowed to do experimentation with the newly set website at your host.

While publishing this website on your localhost, configure your apache server tentatively to avoid directory access.
The setup is complete.

Bibliography

[1] Moodle official website. http://moodle.org.

[2] Moodle api. http://docs.moodle.org/dev/Core_APIs.

[3] Access api. http://docs.moodle.org/dev/Access_API.

[4] Capabilities array. https://github.com/harshadwaghmare/ImprovementsToMoodle/blob/master/mod/assign/db/access.php.

[5] String api. http://docs.moodle.org/dev/String_API.

[6] String example. https://github.com/harshadwaghmare/ImprovementsToMoodle/blob/master/lang/en/moodle.php.

[7] Page api. http://docs.moodle.org/dev/Page_API.

[8] Output api. http://docs.moodle.org/dev/Output_API.

[9] Multiple choice questions. http://docs.moodle.org/25/en/Multiple_Choice_question_type.

[10] edit_simple_multichoice_form.php. https://github.com/harshadwaghmare/ImprovementsToMoodle/blob/master/question/type/simple_multichoice/edit_simple_multichoice_form.php.

[11] Question type multichoice. http://docs.moodle.org/dev/Multiple_Choice_question_type.

[12] Essay question type. http://docs.moodle.org/25/en/Essay_question_type.

[13] Matching question type. http://docs.moodle.org/25/en/Matching_question_type.

[14] Multichoice question type documentation. http://docs.moodle.org/25/en/Multiple_Choice_question_type.

[15] Places to search for lang strings. http://docs.moodle.org/dev/Places_to_search_for_lang_strings.

[16] smanage.php. https://github.com/harshadwaghmare/ImprovementsToMoodle/blob/master/course/smanage.php.

[17] action.php uploaded on github. https://github.com/harshadwaghmare/ImprovementsToMoodle/blob/master/course/action.php.

[18] actioncat.php. https://github.com/harshadwaghmare/ImprovementsToMoodle/blob/master/course/actioncat.php.

[19] Context module. http://docs.moodle.org/dev/Roles_and_modules#Context.

[20] File api internals. http://docs.moodle.org/dev/File_API_internals#File_browsing_API.

[21] Code for adding download link. https://github.com/harshadwaghmare/ImprovementsToMoodle/blob/master/grade/report/grader/lib.php.

[22] Gradebook api. http://docs.moodle.org/dev/Gradebook_API.

[23] Gradebook. http://docs.moodle.org/25/en/Grader_report.

[24] grade_report.php. https://github.com/harshadwaghmare/ImprovementsToMoodle/blob/master/user/profile/grade_report.php.

[25] export.php. https://github.com/harshadwaghmare/ImprovementsToMoodle/blob/master/user/profile/export.php.

[26] html_writer::table. http://docs.moodle.org/dev/Output_API#html_table.

[27] Cross site scripting attacks. http://en.wikipedia.org/wiki/Cross-site_scripting.

[28] Nicola Asuni. Moodle pdf library. http://www.tcpdf.org/.

[29] Jason Hardin. Activity grading interface specification. http://docs.moodle.org/dev/Activity_Grading_Interface_Specification, April 2013.