Documentation on the stages of development of the My Progress Moodle Plugin

Jean Crudden

2467739

Submitted in partial fulfillment for the Higher Diploma of

Science in Web Development

Griffith College Dublin

June, 2015

Under the supervision of Osama Abushama

Disclaimer

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the Higher Diploma of Science in Web Development at Griffith College Dublin, is entirely my own work and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Signed: ______________________ **Date: ____________________**

Table of Contents

List of Figures

Abstract

My Progress was developed as a plugin for Moodle’s platform. The rationale behind the development was to provide students with a tool to track their overall progress for courses they are enrolled in on Moodle. There are many plugins that are available on Moodle that allow a student to track their progress, however these are only administered at a course level. Therefore if a student is enrolled in more than one course it is difficult to have a holistic view of all the courses they are assessed in. This in turn prevents a student from quickly assessing what work assignments are due, what grades they received or how many work assignments they have coming up. My Progress allows students to quickly access this information which in turn enables them to prioritise their work.

Chapter 1. Introduction

<u>Overview</u>:

This chapter first addresses the rationale and motivation behind the development of the My Progress plugin. Once this has been established we then move on to discuss the main objectives of the My Progress plugin. Then we briefly discuss the overview of the approach that was taken to achieve these main objectives. This chapter then ends with a discussion detailing the structure of this document for reference.

1.1 Rationale and motivation behind the My Progress plugin

Initially this project was designed as a stand-alone program that allowed a user to keep track of their records by entering details in a 'fake' university database. This information would be taking from form fields and entered in a relational database with for example, 'student', 'teacher' and 'course' entities. However it was realised that this stand-alone program would have little use outside of this development. The university database would be created with unique specifications and relations that would be hard to implement elsewhere. Therefore it was decided to develop a plugin for Moodle that could be implemented elsewhere across different universities. As a user of Moodle it was decided that this system would be ideal to develop a program for. Moodle is an open source Learning Management System. Its design and development is guided by social constructionist pedagogy which focuses on 'learning from the learner's point of view'. My Progress was designed and developed as a plugin for Moodle with this social constructivist philosophy in mind. It centres the student as an active agent allowing them to have more engagement with their studies.

1.2 Objectives

There are many plugins that are available to use in Moodle but due to their design they are based at a ‘course’ level. This means that if these plugins were installed and enabled to use by teachers, they can set tasks, assignments, grade marks, give feedback. However while this can be very useful there is a major drawback – a teacher can have many students but a course can only have one teacher. Therefore even if a teacher implements the use of these plugins, it is unlikely that all the teachers in all of a student’s courses will do the same. Therefore the plugin may work really well for one of the student’s courses but they have no way of gaining a holistic picture of all the work assignments that are due or graded for all their courses. This was the fundamental objective of this project – to allow students to have a holistic view of their work items and provide a quick means to do so. An overview of how this was achieved is discussed briefly below.

1.3 Overview of Approach

After discovering that a stand-alone system would not suffice I began to look for alternatives. When I discovered that Moodle uses MySQL database and PHP as a scripting language I decided that it would be best to develop a program for Moodle. As this was at a late stage in the project, with a two week deadline to complete, I quickly had to learn about Moodle’s system. I downloaded the latest core version and began some administration, such as adding users and courses as I knew I would need these to work with later. From here I chose what type of plugin I was going to use. I decided on a block as it suited the idea of the progress tracker best. Blocks are items which may be added to the left or right or centre column of any page in Moodle. Once a block was decided upon I wanted to understand how it was implemented, so I started to create a simple block to have an overall look of it and what was possible to create in it. This entailed building the basic files necessary for a block, including a block class that extends from a block_list class that was available from Moodle’s core, a language file to hold the strings that I would be using, an access file with standard rules to access Moodle databases and a version file which holds the current version of Moodle’s core and the version of block that has been created. From here I was concerned about Moodle’s core database, which has over 200 tables and was worried that a lack of understanding would hinder the project later down the line. However I started to build a proof of concept outside of Moodle to test my idea and to understand what features could be added or would work well. After the proof of concept I began to work within the Moodle's directory again. I had already enrolled users on the system upon initial installation so I began to create a table to hold the work items. From here I began to implement the proof of concept files that I had created and started the testing and implementation of the plugin. There were many challenges along the way which will be discussed throughout later chapters.

1.4 Document Structure

The rest of this document takes the following structure. Chapter Two provides a literature review of the area of the benefits of e-learning tools such as My Progress and the sources consulted in accomplishing my project. It then moves on to discuss the related work of the progress tracker that I created. This area focuses on existing plugins that are available in Moodle that are similar to my own. Chapter Three describes the methodology. Here a brief look over the steps that were necessary to achieve the project objectives and some of the challenges that were faced along the way. In Chapter Four, I discuss the system design and requirements and specifications including hardware that was used. Also in this chapter I discuss the data modelling used and described the main PHP scripts and their functionalities. Chapter Five provides the implementation details of the My Progress plugin from an administrator and user point of view.

In Chapter Six, details of the working prototype of the My Progress plugin are provided, including the testing and evaluation. I also discuss results including the revisions to the overall design and implementation that were deemed necessary. Finally, Chapter Seven discusses how some of the issues raised throughout the chapters could have been avoided and ends with a brief conclusion and discussion of future work.

Chapter 2. Background

Overview:

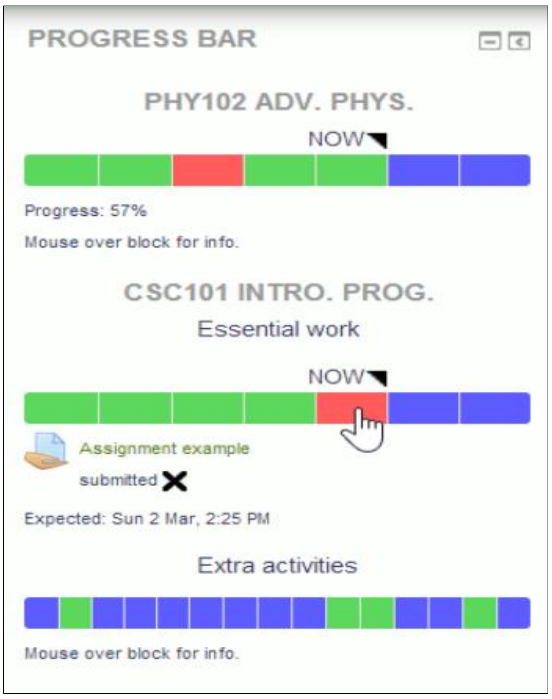
This chapter discusses some of the research around the benefits of e-Learning tools that encourage and facilitate a student's engagement with their studies. This literature review briefly looks at the literature surrounding e-Learning and the technology that is used to facilitate its adoption as an additional learning method. Herein lies the research that has shown how technology such as progress trackers and time management tools can facilitate a positive effect on a student's experience in their studies. This chapter then turns focus to related plugins, their usage and technical drawbacks and ends with comments on these plugins.

2.1 Literature Review

[1] Garrison (2011) defines e-learning as electronically mediated asynchronous and synchronous communication for the purpose of constructing and confirming knowledge. At the core of this is the idea that students should be actively engaged in sustainable communities of inquiry. Likewise [2] e-learning can come under a constructivist paradigm where the core assumptions are active engagement in learning and learner responsibility for the management of learning. An OECD report [3] suggests that the ability to track a student's progress toward objective learning goals is more effective than comparison with peer's progress. Reference to an individual's progress and opportunities to improve work based on feedback can help counter negative impact of social comparisons where a student absorbs the idea that they lack ability compared to their peer's work. The design and development of Moodle is guided by a social constructionist pedagogy [4]. Constructionism asserts that learning is particularly effective when constructing something for others to experience. It was found from this brief research that a tool that enabled students to keep track of their progress through their studies would prove beneficial.

2.2 Related Work

A core element of Moodle is the ability to track progress. Many tools have been developed that facilitate this. Built into Moodle's core platform are four components that facilitate progress tracking grades, conditional activities, activity completion and course completion. From these four components many additional plugins have been created. The most closely related plugins are the Progress Bar [5] and FN-My Progress [6] plugins. They are both block type plugins that allow a student to track their progress to some degree.


PROGRESS BAR

PHY102 ADV. PHYS.

NOW

Progress: 57%

Mouse over block for info.

CSC101 INTRO. PROG.

Essential work

NOW

Assignment example

submitted

Expected: Sun 2 Mar, 2:25 PM

Extra activities

Mouse over block for info.

Figure 1 Progress Bar

Figure 1 Progress Bar above shows how a student can see that for their class CSC101 that some work has been submitted. Work items they have due are marked in red and work items they have submitted and received a grade in 57%. A good feature about this is that it links with the assignment that a teacher has uploaded to complete and also links with the grade that a teacher has given for that assignment.

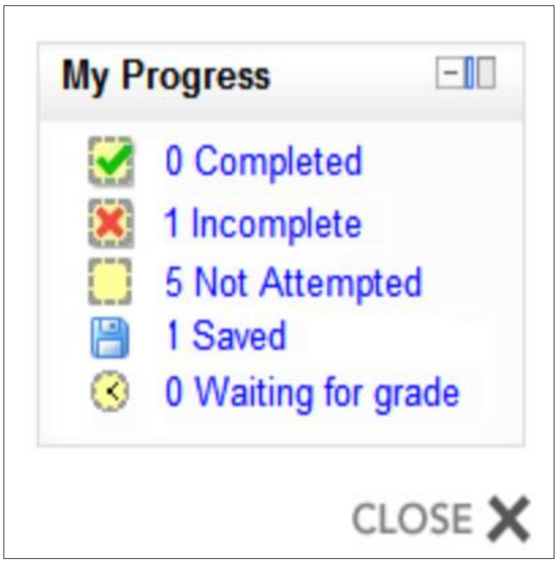

My Progress
0 Completed
1 Incomplete
5 Not Attempted
1 Saved
0 Waiting for grade
CLOSE

Figure 2 FN-My Progress

Figure 2 depicts the FN-My Progress plugin. This gives students an opportunity to see what work items they have upcoming, what files they have saved and work items that are due to submit but have not been attempted yet. A good feature of this is that it allows students to save files that are related to the assignments that teachers have put up and allows graded items to show in the block. However to use this plugin you need to install an additional plugin, Tabs Course formats. This is a drawback of this plugin because of the extra installation that is required.

The key difference between My Progress and the Progress Bar and FN-My Progress plugins are connected to the grade, conditional activities, activity completion and course completed components in some manner or other. This is due to the fact that these plugins are assumed to work at a course level. However working at a course level assumes that there is one teacher per course. The key issue that I found with this approach is that it limits the possibility for a student to have a holistic picture of their progress. I will explain how this manifests in the two scenarios below.

2. 2. 1 Scenario 1: No Guarantee Standardised Use

A student is enrolled in 5 courses (CourseA, CourseB, CourseC, CourseD, CourseE). However it is not guaranteed that each of the student's teachers will chose to use either of the plugins. For example the teacher in CourseA decides to use one of the plugins, however the remaining teachers may not chose to use it. Therefore the student is unable to view their progress for the remaining courses and may seek to do their own evaluation elsewhere.

2.2.2 Scenario 2: Enforced Standardisation

A student is enrolled in 5 courses (CourseA, CourseB, CourseC, CourseD, CourseE). All the teachers of the student use either of the plugins. However it is unlikely that each of the teachers will use the plugins in the exact same way. There are different approaches to how these plugins can be used and they each give flexibility to the teacher. A standardised way of implementing these plugins would need to be enforced for a student to have a holistic point of view of their progress. This I believe would be an unlikely scenario as these plugins are designed to give flexibility to the teacher, so enforcing them in a standardised way would reduce the plugins functionalities and therefore reduce the likelihood of teachers adapting them.

2.2.3 Other issues with the My Progress Bar and FN-MyProgress plugins

Another issue that I think is important is the usability. I wanted this plugin to be easy to use and have as little integration as possible. For the FN-My Progress plugin you are required to implement an additional plugin for it to work. This I believe doubles the workload and may discourage people from using it. Also as they work with the four components previously mentioned there are additional role settings that an administrator would need to maintain. Although my plugin is simpler it also reduces the administration upkeep. You can find the installation details in section Chapter 5 on Implementation. Additionally students are unable to have a holistic picture of their progress as these plugins are based at course level, however a student has many courses.

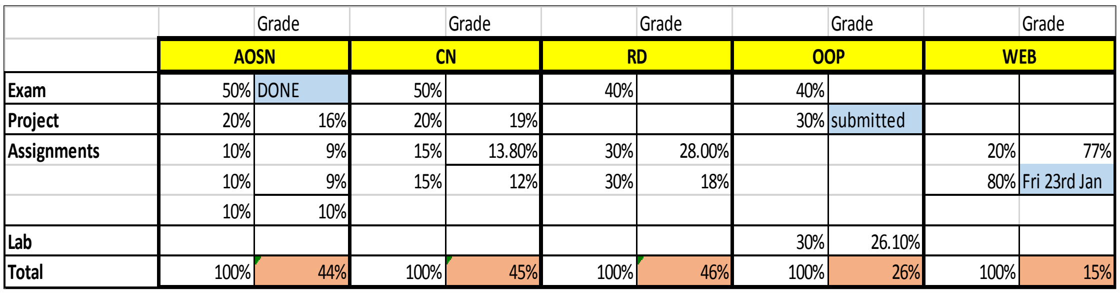
Chapter 3. Methodology

<u>Overview</u>:

As a single developer it was difficult to follow a software methodology that would see the main idea of the project through. The methodology that most closely suited the project reflected some of the principles of Agile methodology. This included prototyping and developing user stories. This approach was favoured over the Waterfall model which focuses largely on documentation where coding only takes place in the later stages in development. The methodology I chose to use allowed me to develop a proof of concept outside of Moodle while simultaneously building a prototype of a block plugin within Moodle. In this chapter I describe my approach to the development that is taking the initial idea from concept through to prototype through to a working solution.

3.1 Initial Idea

The rationale for developing a progress tracker came from creating one myself to keep track of the assignments that I submitted for college and the grades that I received.



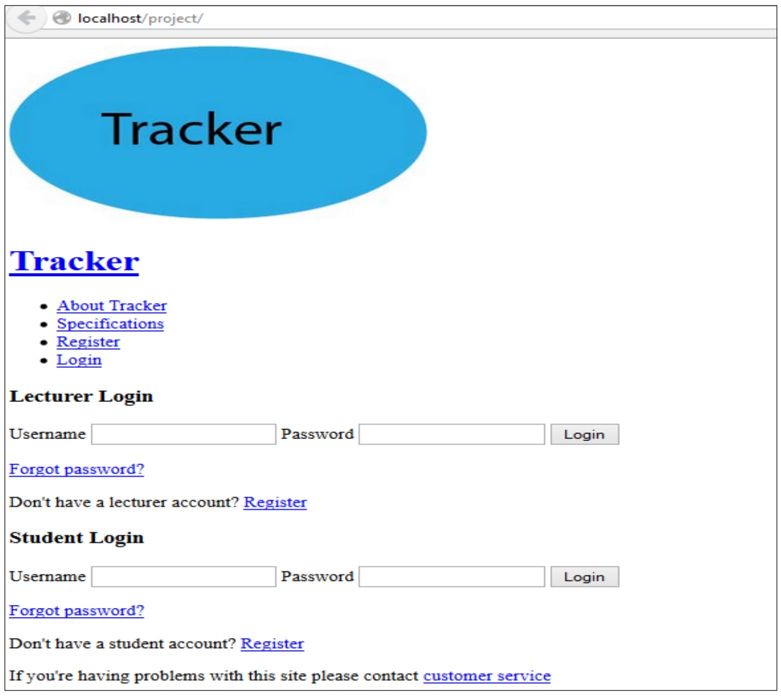
		Grade		Grade		Grade		Grade		Grade
	AOSN		CN		RD		OOP		WEB	
Exam	50%	DONE	50%		40%		40%			
Project	20%	16%	20%	19%			30%	submitted		
Assignments	10%	9%	15%	13.80%	30%	28.00%			20%	77%
	10%	9%	15%	12%	30%	18%			80%	Fri 23rd Jan
	10%	10%								
Lab							30%	26.10%		
Total	100%	44%	100%	45%	100%	46%	100%	26%	100%	15%

Figure 3 Initial idea captured in MS Excel

Figure 3 above depicts the first instance of the progress tracker I kept in a MS Excel sheet. Those marked in light red are the grades that I received before entering into the exams. As the exams made up 50% of each of the courses (except for the Web course) I could see what areas I needed to prioritise, what needed to be submitted and the dates that they were due. I found this really helpful as there were a lot of assignments due, with this simple time management tool it allowed me to keep track of it.

3.2 Initial Stand-Alone Program

My initial response to develop a program that replicated this was to design a stand-alone program that I would create courses, teachers, students.



localhost/project/

Tracker

Tracker

- About Tracker
- Specifications
- Register
- Login

Lecturer Login

Username Password Login

Forgot password?

Don't have a lecturer account? Register

Student Login

Username Password Login

Forgot password?

Don't have a student account? Register

If you're having problems with this site please contact customer service

Figure 4 Index page of stand-alone program

Figure 4 demonstrates the initial standalone program. A lecturer and student could register for accounts. Upon registration a lecturer would choose what faculty they belong to, what courses they teach, what modules in those courses they teach and personal details, such as name and email. The same registration process would be for students. This led me to the realisation that if this program was to be used again there was little possibility of code refactoring.

3.3. Deciding on Moodle

Once passed the initial concept in excel and stand-alone program I considered that I was duplicating data that already existed in Moodle outside of the Moodle system. I considered that it would be best to leverage Moodle's extensibility and open source community based features to provide my solution. I considered that a Moodle solution resulted in the least amount of data and functionality duplication and offered a solution that could be made available not only for my own use but for the entire Moodle community with few dependencies involved. For example if I had chosen the proceed with the stand-alone program I would need to consider dependencies such as hosting, whereas by developing within Moodle's framework the plugin would be available to standard Moodle users which has a base of over x amount of users worldwide. Also Moodle was chosen as I had baseline knowledge of how Moodle operated as a user in Griffith College.

3.4 Deciding on a Moodle block

After reading documentation on Moodle's plugins it was decided that a block would be the best possible choice of plugin to implement. A block is a small information-display that can sit at the centre or side of a Moodle page. Questions were asked such as - what information could be added to blocks? How does information pass from one page to the next? How will the navigation remain consistent to Moodle's look when passing from one page to another? It was decided that the best way to answer these questions was to start building a simple block to understand the possible functionalities that could be contained within a block.

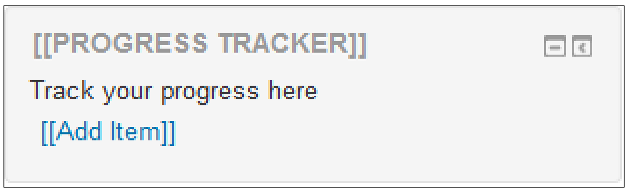
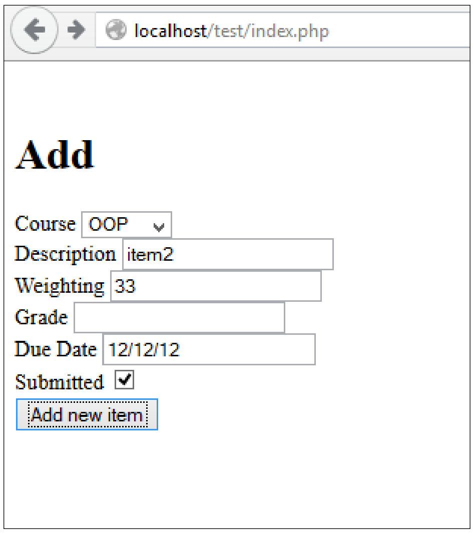

[[PROGRESS TRACKER]]
Track your progress here
[[Add Item]]

Figure 5 First view of a block

Figure 5 above shows the first build of the block. This gave me an indication of what content was possible to build inside a block and how it should be implemented.

3.5 Developing a proof of concept

While the documentation available for building Moodle blocks was extensive, some of the features I was unfamiliar with. For example the use of html writers was a technology I had not come across before. At this stage it was still unclear how the main functionalities of the block could be implemented. It was decided here that the quickest way to see how these functionalities could be realised was to build a prototype outside of Moodle using a html form and PHP scripts as I was familiar with these technologies.


localhost/test/index.php

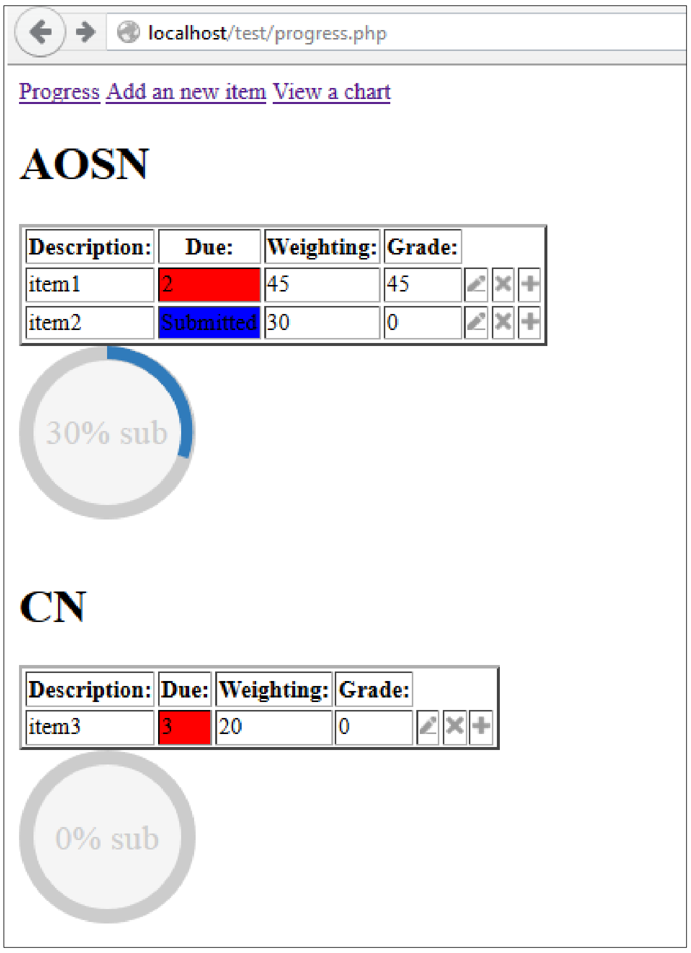
Add

Course	OOP
Description	item2
Weighting	33
Grade	
Due Date	12/12/12
Submitted	✓

Add new item

Figure 6 Proof of concept HTML Form

Figure 6 above depicts the initial HTML form that would take in values of a course, a description for a work item, a weighting of this item, a date submitted. A database table was created to capture the values of these items. When ‘Add new item’ was clicked, it ran a script to reflect the values added and provided a graphic to the user using a CSS radial progress indicator.


localhost/test/progress.php

Progress Add an new item View a chart

AOSN

Description:	Due:	Weighting:	Grade:
item1	2	45	45
item2	Submitted	30	0

CN

Description:	Due:	Weighting:	Grade:
item3	3	20	0

Figure 7 Proof of concept result page

Figure 7 depicts the result page of the proof of concept. I detail how this was achieved specifically in the next chapter. By building this proof of concept outside of Moodle I was able to understand the steps that were necessary to see the concept of the block implemented within Moodle.

Chapter 4. System Design and Specifications

Overview:

This chapter describes the specific technologies that were used and necessary in order for the My Progress block to be developed. It then discuss why some technologies were favoured over others

4.1 Required Technologies

A WAMP stack which provides an Apache Web server, MySQL database components and PHP dynamic scripting language. A Moodle installation comprises the Moodle code executing in a PHP-capable web server, a database managed by MySQL and a file store for uploaded and generated files (the moodledata folder) [7]. Moodle self-installs once the code has been copied to the web server and a blank database created. [8] Moodle was installed in a Windows OS running with Apache version 2.4.9, PHP version 5.5.12, MYSQL version 5.6.17. It is noted that from PHP 5.5 onwards Visual C++ Redistributable for Visual Studio 2012 must also be installed. The latest version (2.9) of Moodle's platform was downloaded which requires PHP 5.4.4 and MySQL 5.5.31 onwards. All three parts can run on a single server or they can be separated with many load-balanced web-servers, a database cluster, and a file-server; or anywhere between those extremes. For this project they were run a single Apache server from a Windows 8 Operating System.

4. 2 Hardware

Disk space of 160MB free (min) plus as much as you need to store your materials. Moodle suggest that 5GB is a realistic minimum. Processor speed of 1GHz (min) however 2GHz dual core is recommended. A memory of 256MB (min), 1GB or more is strongly recommended. The general rule of thumb as determined by Moodle is that Moodle can support 10 to 20 concurrent users for every 1GB of RAM, but this will vary depending on your specific hardware and software combination and the type of use. [8].

4.3 Browser Compatibility

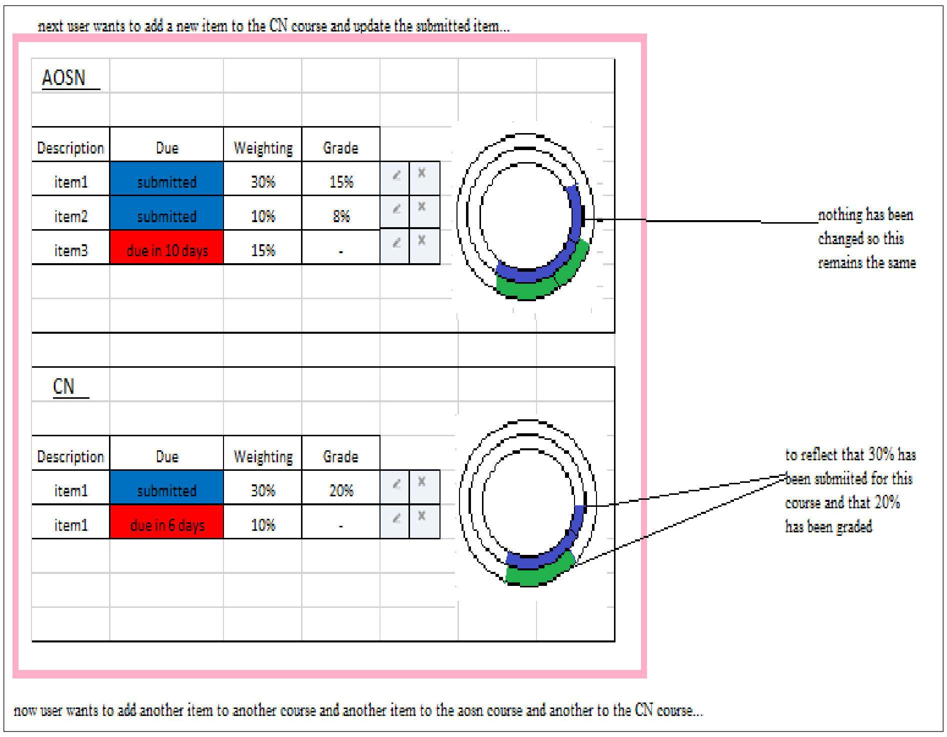
Another advantage of building within in Moodle was the client side support. All core Moodle platforms support the major browsers such as Firefox (minimum version 25.0), Chrome (minimum version 30.0), Apple Safari (minimum version 6), IE (minimum version 9).

4.4 The use of CSS over JavaScript and AMD

Initially JavaScript was considered to demonstrate the block progress circles. However there were serious drawbacks to this. Moodle is currently undergoing a transition from YUI (Yahoo User Interface) to AMD (Asynchronous Module Definition) for writing JavaScript modules. I downloaded NodeJS and its package manager NPM. After spending a considerable amount of time trying to understand how NPM works without success I discovered a way that the progress circles could be implemented with CSS alone.

4.5 Wireframe

I began to design a wire frame to visualise the end result and being to understand where on the Moodle page the content would be shown.



next user wants to add a new item to the CN course and update the submitted item...

AOSN

Description	Due	Weighting	Grade
item1	submitted	30%	15%
item2	submitted	10%	8%
item3	due in 10 days	15%	-

nothing has been changed so this remains the same

CN

Description	Due	Weighting	Grade
item1	submitted	30%	20%
item1	due in 6 days	10%	-

to reflect that 30% has been submiited for this course and that 20% has been graded

now user wants to add another item to another course and another item to the aosn course and another to the CN course...

Figure 8 Initial Design Concept

Figure 8 above is the initial wireframe created in MS paint. Work items are ordered by their associative course. A radial progress bar reflects the % of graded and submitted items. Days remaining and submitted are captured in the due field. Links are provided to edit or delete each work item that has been added.

Also it was important to see how this would be viewed on a Moodle page. To do this I took screen shots of Moodle's form page I had created (details on this form are provided below). This can be viewed in Figure 9 below.

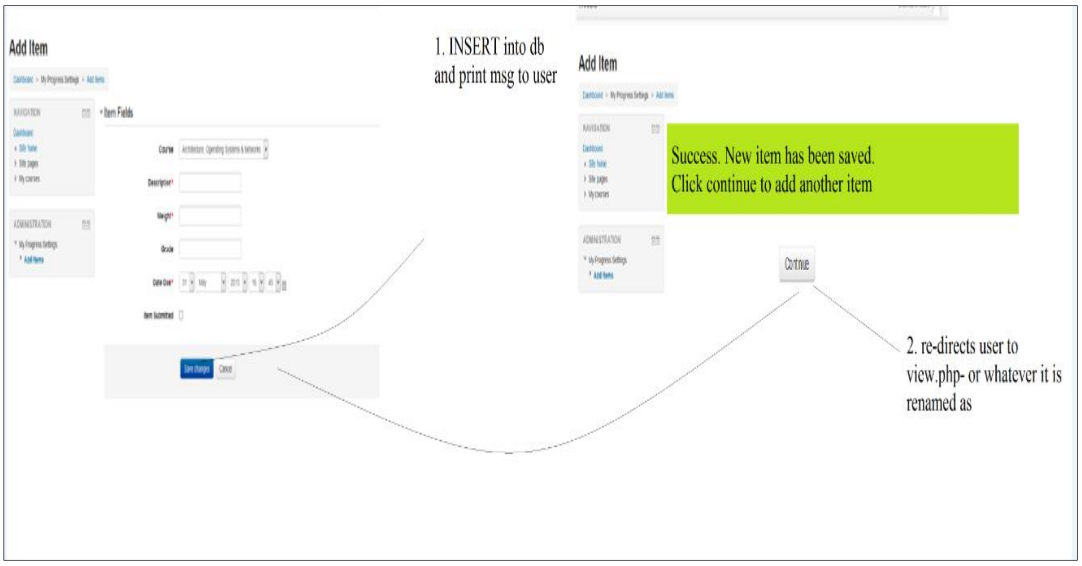

Add Item
1. INSERT into db and print msg to user
Add Item
Success. New item has been saved. Click continue to add another item
Continue
2. re-directs user to view.php- or whatever it is renamed as

Figure 9 Initial Wire frame on Moodle Page

4. 6 Data Modelling

As mentioned in previous chapters I started building a basic block using the documentation available online [9] which implements an Object Oriented Programming approach. In this section I discuss the files that are required for building a basic block in Moodle and then continue to discuss the files that were needed to implement the functionalities of the My Progress block.

4.6.1 Moodle Blocks: Required Files

There are four files necessary to build a basic block in Moodle. The function of each of these files are described briefly below.

1. block_myprogress

block_myprogress is the main block class which is used both to manage it as a plugin and to render it onscreen. As this block required a list of links to be included in the block type this class was required to extend the block_list specifically. block_list allows multiple content to appear in the block as opposed to the generic block_base class that only simple html text. A public function init() was created as it is required for all block definitions, its purpose is to give values to any class member variables that need instantiating.

2. db/access.php

This file holds the new capabilities created by the block. Moodle 2.4 onwards introduced the capabilities addinstance and myaddinstance for core blocks. These capabilities make it possible to control the use of individual blocks.

3. lang/en/block_myprogress.php

This is the English language file for the block. All language files for blocks must go under the /lang subfolder of the block's installation folder. Moodle 2.0 and above require a name for the plugin to show in the upgrading page, this was set to My Progress in the file. Other strings were set here included header names for pages and names given to links defined within the block.

4. version.php

Prior to Moodle 2.0, version details for blocks were stored as class fields; as of Moodle 2.0 these are stored in a file called version.php, stored under */blocks/myprogress/**version.php***. The version file contains the version of the Moodle core platform you are developing in and the version of the plugin created.

4.6.2 My Progress Block: Additional Files

The four files described above allowed the build of a basic block, however to implement the functionalities and incorporate the proof of concept described in the previous chapter it was necessary to create additional files. I describe each of these files below.

1. myprogress_form

myprogress_form requires Moodle's form API, here form elements are described with rules enforced with them, such as client side validation.

2. view.php

view.php refactors myprogress_form by requiring it once and controls the three states that a form can be in, cancelled, loaded for the first time and changes saved. It redirects the user based on each of these states. It also uses Moodle’s global $PAGE variable to set the URL for the page, set the standard Moodle page layout, set the heading and provide the navigation breadcrumbs for the user. Issues with navigation will be discussed in the final chapter.

3. **progress.php**

progress.php is the most complex script. This returns all the items relating to a user and orders them by course.

The pseudo code below gives a description of how this is achieved:

```
$sql = SELECT * FROM TABLE
foreach (element as row){
        if(course is not equal to the previous course){
                if( not firstime in loop){
                        $allitems = end table and radials
                }//ends if not first time in loop condition
                //set firstime in loop to false here
                firsttime in loop =  false
                //constants for every course
                $allitems = course heading, table headers
        }// ends if course is not equal to previous course condition

        $allitems = table row values
}//ends foreach loop
$allitems = end table and radials
echo $allitems;
```

When the SQL statement is run to the database to retrieve all elements in the row, we want to separate them into tables having the course as the header. The first iteration of the loop ends the table and returns the progress radials, until the next iteration where the next course header is found. Ideally this script should be re-written as the design was based on the proof of concept it was hard to implement with the rest of the block that was already created. The implications of this are discussed in the final chapter in more detail.

3. myprogress_edit_form.php

myprogress_edit_form is exactly the same as myprogress_form. Upon reflection, it was not necessary to create this form again. This is discussed below.

4. edit.php

When a user clicks on the edit icon beside a work item on the View Progress page. The PHP $_GET associative array variable was used on the query string to access the unique id of the work item. This page refactors the myprogress_edit_form.php code by requiring it once. As mentioned above this was unnecessary to create the myprogress_edit_form page, instead edit.php could just have required myprogress_edit form and would have achieved the same results. This duplication of code was unfortunately overlooked. Similar to view.php it controls the three states of the form, loaded for the first time, cancelled or changes saved and redirects the user based on these states. It also uses Moodle's global $PAGE variable to set the URL for the page, set the standard Moodle page layout, set the heading and provide the navigation breadcrumbs for the user. Issues with navigation will be discussed in the final chapter.

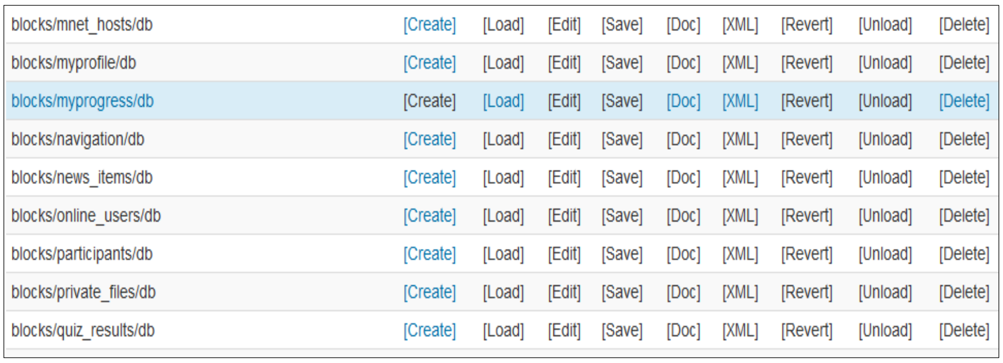
5.delete.php

When a user clicks on the delete icon beside a work item on the View Progress page. The PHP $_GET associative array variable was used on the query string to access the unique id of the work item. It accessing a confirmation page before deletion of a work item to allow a user to choose to either continue with the deletion or to cancel the deletion and return to the progress.php page. There were issues with the look of this page which will be discussed in the final chapter.

4. 7 Database design and considerations

There was only 1 table created that associates with two other tables that relate to Moodle’s core schema - the user table (which stores all the information of any users that are enrolled in a course) and the course table (which holds all the information about the course, such as ID, fullname, shortname, start and end dates). Moodle uses a XMLDB editor that defines the database structure and contains comments on the table and database fields to document their purpose. In site administration you can control this if you go to site administration->development-> XMLDB editor. The XMLDB editor is a tool for making the .xml files that specify how Moodle should set up its database tables. Previously, developers had to make separate .sql install files for MySQL and postgres, but now only database-neutral file is needed, which supports many more databases. To be able to handle files properly, the web server needs write access to all *db* directories where the *install.xml* files reside [10].
Once in the XMLDB editor you can design and create the database to hold the information taken from the form fields. As a folder named DB has already been created in the block directory this is where the install.xml file holding the database information will be saved to when saving in the XMLDB editor.

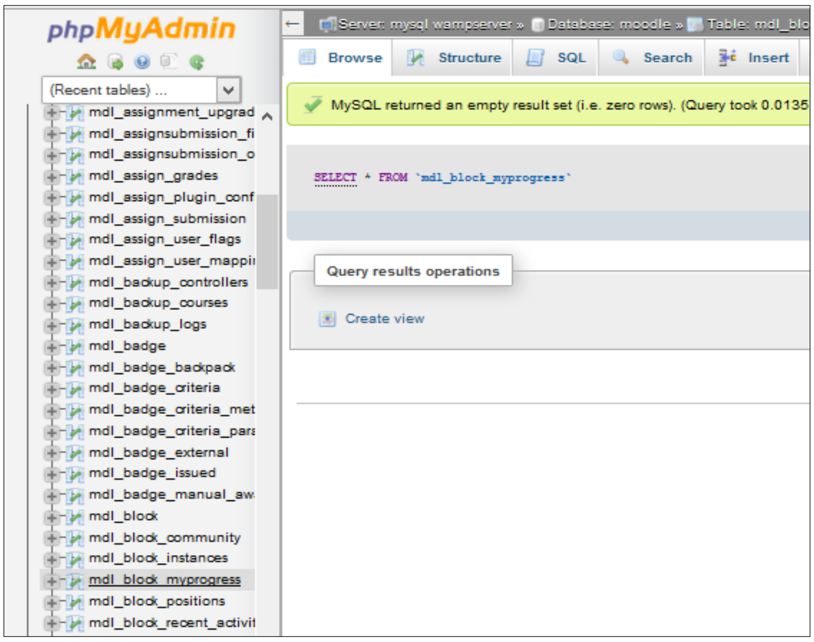
Figure 7 below shows the XMLDB editor and the links to the steps that need to be carried out.



blocks/mnet_hosts/db	[Create]	[Load]	[Edit]	[Save]	[Doc]	[XML]	[Revert]	[Unload]	[Delete]
blocks/myprofile/db	[Create]	[Load]	[Edit]	[Save]	[Doc]	[XML]	[Revert]	[Unload]	[Delete]
blocks/myprogress/db	[Create]	[Load]	[Edit]	[Save]	[Doc]	[XML]	[Revert]	[Unload]	[Delete]
blocks/navigation/db	[Create]	[Load]	[Edit]	[Save]	[Doc]	[XML]	[Revert]	[Unload]	[Delete]
blocks/news_items/db	[Create]	[Load]	[Edit]	[Save]	[Doc]	[XML]	[Revert]	[Unload]	[Delete]
blocks/online_users/db	[Create]	[Load]	[Edit]	[Save]	[Doc]	[XML]	[Revert]	[Unload]	[Delete]
blocks/participants/db	[Create]	[Load]	[Edit]	[Save]	[Doc]	[XML]	[Revert]	[Unload]	[Delete]
blocks/private_files/db	[Create]	[Load]	[Edit]	[Save]	[Doc]	[XML]	[Revert]	[Unload]	[Delete]
blocks/quiz_results/db	[Create]	[Load]	[Edit]	[Save]	[Doc]	[XML]	[Revert]	[Unload]	[Delete]

Figure 10 XMLDB Editor View

Once Create has been click on it brings you to a page where you can start adding the fields required for your database. In the table we wanted to take note of all the fields in form plus a hidden field that captures the userid. This was declared in the myprogress_form mentioned above. (The fields and schema are addressed in the next section). Once all the fields have been added to the database, you save the XML file. This should automatically save to the /db folder. It holds the XML schema of your database. A check was made to ensure that that table was added in phpmyadmin.



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0135

```
SELECT * FROM `mdl_block_myprogress`
```

```
mdl_assignment_upgrad
mdl_assignsubmission_fi
mdl_assignsubmission_o
mdl_assign_grades
mdl_assign_plugin_conf
mdl_assign_submission
mdl_assign_user_flags
mdl_assign_user_mappi
mdl_backup_controllers
mdl_backup_courses
mdl_backup_logs
mdl_badge
mdl_badge_backpack
mdl_badge_criteria
mdl_badge_criteria_met
mdl_badge_criteria_para
mdl_badge_external
mdl_badge_issued
mdl_badge_manual_aw
mdl_block
mdl_block_community
mdl_block_instances
mdl_block_myprogress
mdl_block_positions
mdl_block_recent_activit
```

Figure 11 phpMyAdmin view of table

Figure 11 above shows the table that was created through the XMLDB editor. It was noted that an mdl_ prefix was added to the table name, this was important to note when referring to the table in SQL statements.

4.7.1 ER-Diagram

It was important to understand how the work items captured could be identified by course and by user which are captured in separate core Moodle tables.

The ER-Diagram from Moodle's documentation proved to be hard to understand[1]. Instead built in Moodle functions were used to call all the courses that a user is enrolled in and a global accusative array variable $USER was used to access

[1] An image of a close representation of Moodle's E-R diagram can be found here https://docs.moodle.org/dev/images_dev/5/5a/Moodle2erd.png

information about a student. The E-R diagram in Figure 5 describes the entity of work item, what its attributes are and how they related to Moodle’s grand schema.

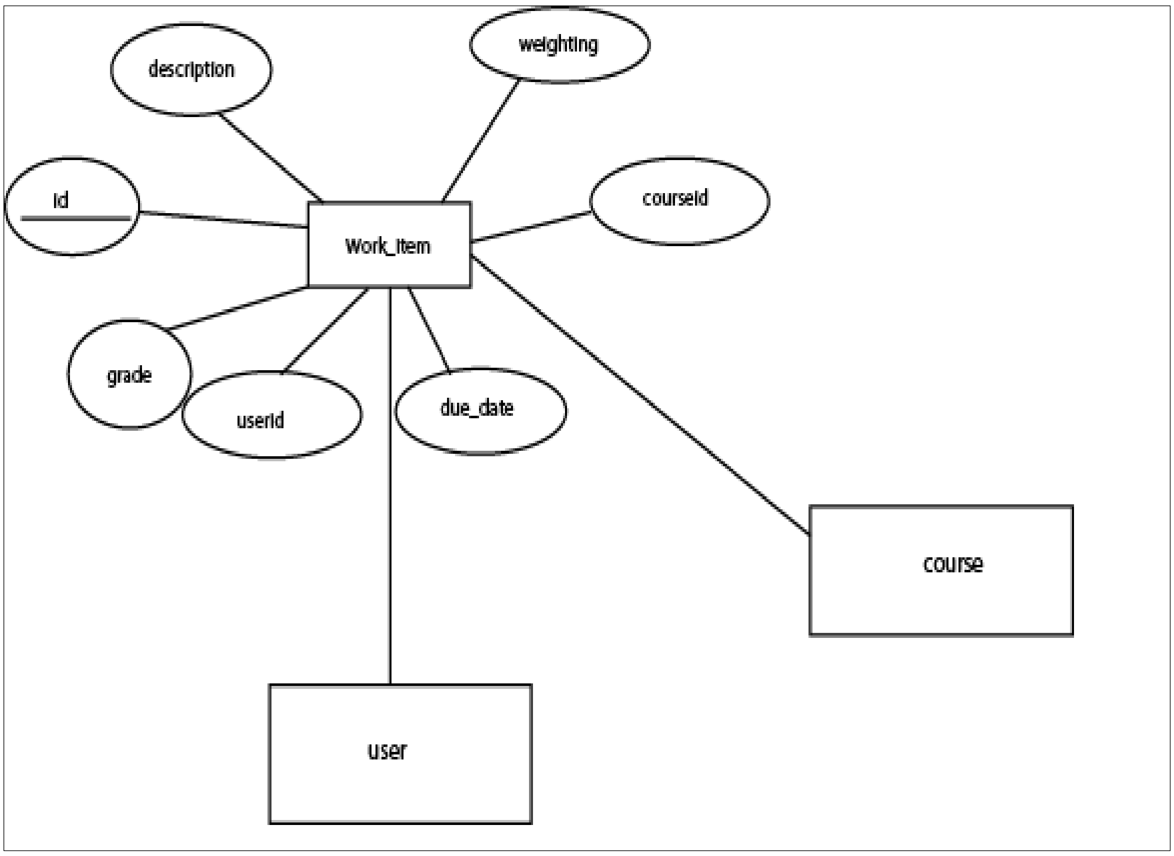
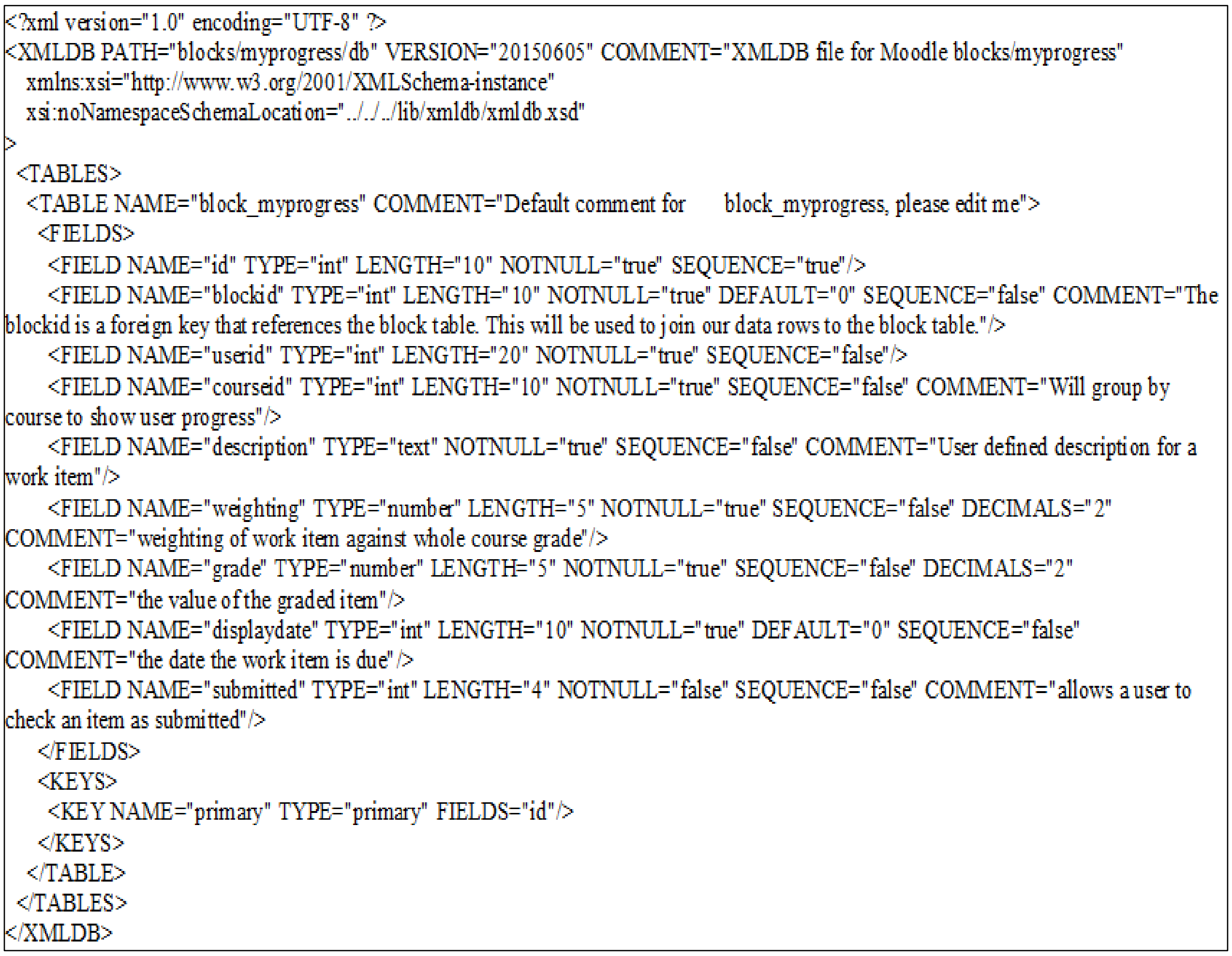


Figure 12 E-R Diagram

4.7.2 Relational Schema



```
<?xml version="1.0" encoding="UTF-8" ?>
<XMLDB PATH="blocks/myprogress/db" VERSION="20150605" COMMENT="XMLDB file for Moodle blocks/myprogress"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../../../lib/xmldb/xmldb.xsd"
>
 <TABLES>
  <TABLE NAME="block_myprogress" COMMENT="Default comment for      block_myprogress, please edit me">
   <FIELDS>
    <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE="true"/>
    <FIELD NAME="blockid" TYPE="int" LENGTH="10" NOTNULL="true" DEFAULT="0" SEQUENCE="false" COMMENT="The
blockid is a foreign key that references the block table. This will be used to join our data rows to the block table."/>
    <FIELD NAME="userid" TYPE="int" LENGTH="20" NOTNULL="true" SEQUENCE="false"/>
    <FIELD NAME="courseid" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE="false" COMMENT="Will group by
course to show user progress"/>
    <FIELD NAME="description" TYPE="text" NOTNULL="true" SEQUENCE="false" COMMENT="User defined description for a
work item"/>
    <FIELD NAME="weighting" TYPE="number" LENGTH="5" NOTNULL="true" SEQUENCE="false" DECIMALS="2"
COMMENT="weighting of work item against whole course grade"/>
    <FIELD NAME="grade" TYPE="number" LENGTH="5" NOTNULL="true" SEQUENCE="false" DECIMALS="2"
COMMENT="the value of the graded item"/>
    <FIELD NAME="displaydate" TYPE="int" LENGTH="10" NOTNULL="true" DEFAULT="0" SEQUENCE="false"
COMMENT="the date the work item is due"/>
    <FIELD NAME="submitted" TYPE="int" LENGTH="4" NOTNULL="false" SEQUENCE="false" COMMENT="allows a user to
check an item as submitted"/>
   </FIELDS>
   <KEYS>
    <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
   </KEYS>
  </TABLE>
 </TABLES>
</XMLDB>
```

Figure 13 XML Schema

Figure 13 above shows the xml schema of the database. This file allows for more compatibility across RDMS's.

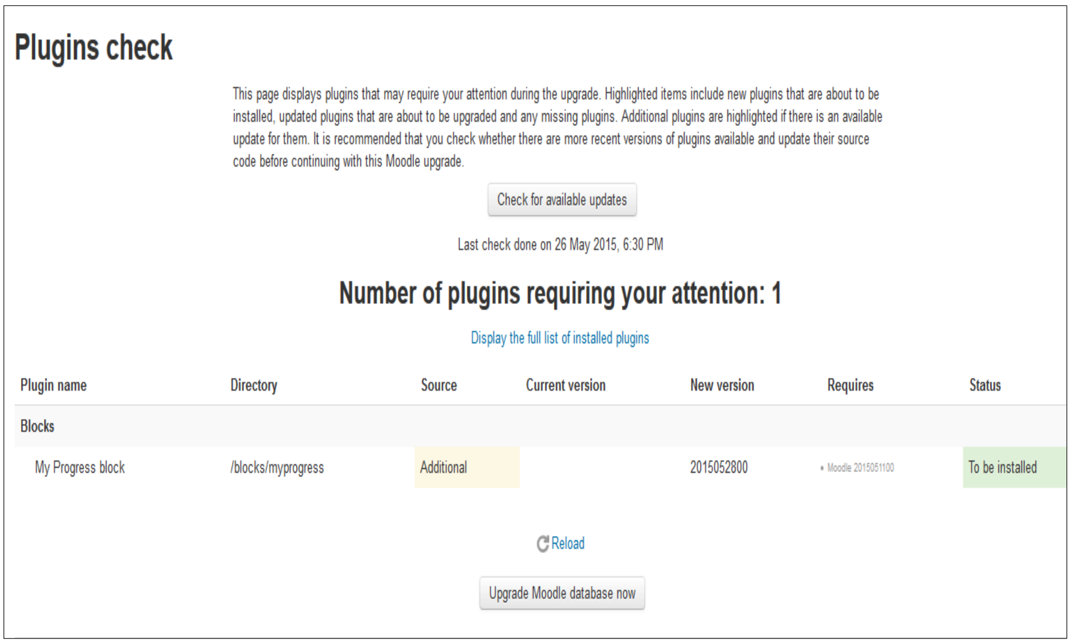
Chapter 5. Implementation

Overview:

This chapter discusses specifically how a working version of the plugin was implemented. It focuses on the necessary requirements for the block to be used. What steps need to be taken from an administrative and student user point of view. It reflects the simple process is required to have the block working within a Moodle environment.

5.1 Block installation

This block is not available to download from Moodle at present. However once it available to download it can be used by copying the folder myprogress within Moodle's directory within the blocks folder. An admin user is then notified with any additional folders that are placed within Moodle's directory by default by navigating to the administration block→notifications.


Plugins check

This page displays plugins that may require your attention during the upgrade. Highlighted items include new plugins that are about to be installed, updated plugins that are about to be upgraded and any missing plugins. Additional plugins are highlighted if there is an available update for them. It is recommended that you check whether there are more recent versions of plugins available and update their source code before continuing with this Moodle upgrade.

Check for available updates

Last check done on 26 May 2015, 6:30 PM

Number of plugins requiring your attention: 1

Display the full list of installed plugins

Plugin name	Directory	Source	Current version	New version	Requires	Status
Blocks						
My Progress block	/blocks/myprogress	Additional		2015052800	• Moodle 2015051100	To be installed

Reload

Upgrade Moodle database now

Figure 14 Install New Plugin Notification

Figure 14 above shows that the My Progress block has been successfully copied into Moodle’s block directory and the correct notification has appeared. The name of the block ‘My Progress block’ is defined in the language file described in the previous chapter. The folder location is described, blocks/myprogress. Also the new version is described, this is the version of the block declared in version.php (mentioned in the previous chapter) and the system version that it requires. This block requires version 2.9+. The limits of this are described in the final chapter. By clicking on Upgrade Moodle database now Moodle’s core is updated along with the installation of the plugin.

5.2 Site Administration

There are predefined roles in Moodle’s core platform. Once a Moodle core installation has been made it is possible for an admin user to create a profile. Site is managed by an administrator user and is defined during setup. Defaults can be edited during setup or globally accepted. Site can be modified by a robust Site administration block. Using Moodle’s platform those with administration control the use of blocks. To enable the use of this block, block editing must be enabled by an admin user.

The administration block is accessible to admin users by default upon Moodle’s core installation.

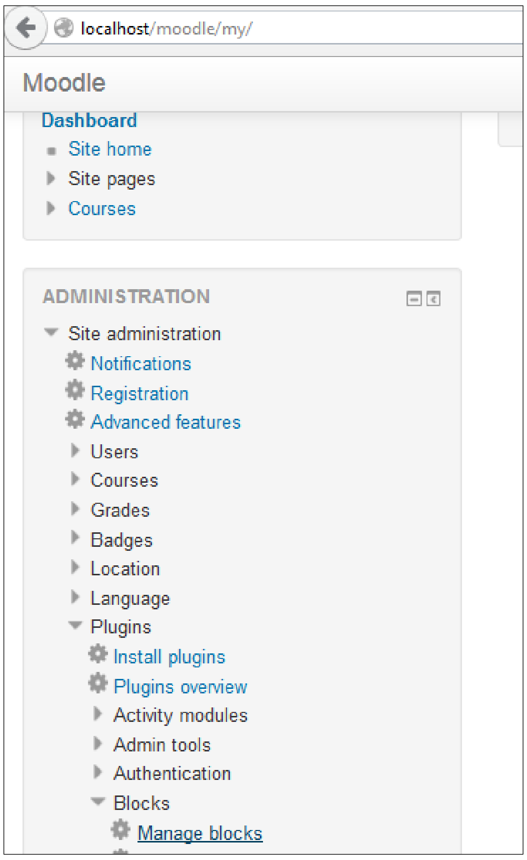


Figure 15 Administration Block

Figure 14 above shows the administration block that is enabled by default once an admin user is defined upon installation. Navigate to Site Administration→Plugins→Blocks→Manage Blocks and click on Manage Blocks. An admin person should ensure that the Blocks editing on button is as shown. This ensures that blocks editing is on. A full list of the blocks installed on the Moodle system are generated. From here it is possible to hide the block, uninstall the block, view how many instances of the block there are and protect the instances of the block. For this block the default option was chosen, therefore the block is accessible by all users.

5.3 User Adds Block

Once the two steps above have been carried out, it is easy for a user to implement the use of the block. By default it is possible for each user to customise their page on Moodle. A user click on the Customise this page button on the right hand top corner of their home page. Once this has been click add a block appears the 'Add A Block' block appears with a list of the blocks available for the user to add, as shown in figure 16 below.

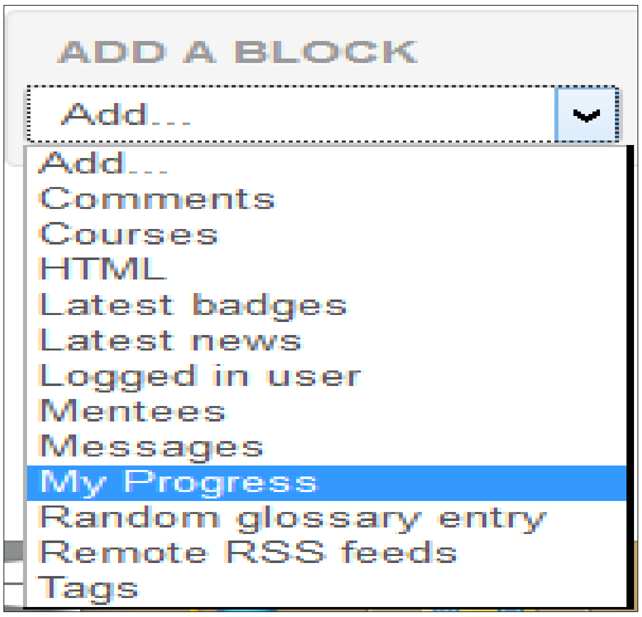


Figure 16 User page customisation: Add My Progress Block

Once My Progress has been clicked on it appears above the 'Add A Block' block as shown in figure 17 below.

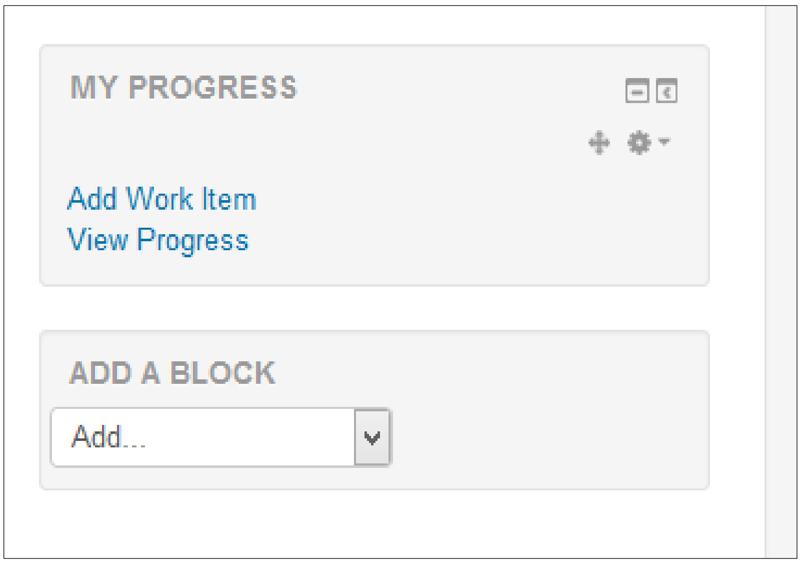
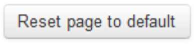
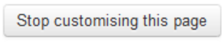


Figure 17 First View of the My Progress Block

The user then clicks on 

which will disable the use of the block or 

which will take the user out of customisation mode and allow them to start using the block.

5.4 Comments

By following the three simple steps above it is easy for the administration user to add the block and control its accessibility. As this block does not work in tandem with any other existing features of Moodle it is not required to reduce access to student users. This I believe has an advantage over the existing plugins that track progress on Moodle that were previously mentioned as default settings apply so the block should be accessible to student users once they click on the

button. Also unlike FN-MyProgress no additional plugins are required for the My Progress block to function.

Chapter 6. Testing and Evaluation

<u>Overview</u>:

This chapter includes a description of the processes you used to test and evaluate the My Progress block by describing some user case scenarios. It then discusses the evaluation of the testing and briefly discusses some of the issues that were raised in the testing stage.

6.1 Debugging Feature

Initially Debugging was enabled through a site administration feature on Moodle to allow errors to be shown. However there were conflicts with headers when debugging was switched on which resulted in page redirects not working. Therefore as the block reached a stable position at this stage, debugging was switched off. There are obvious drawbacks to switching debugging off that will be discussed in the next chapter.

6.2 User Scenarios

Below we discuss some of the user scenarios that were carried out during the testing stage.

User Scenario 1: Charmaine adds an item

Charmaine already has the block enabled by following the steps described in the previous chapter. She wants to add a new item so that she can keep track of her work items. She goes to the block and sees the Add New Item link.

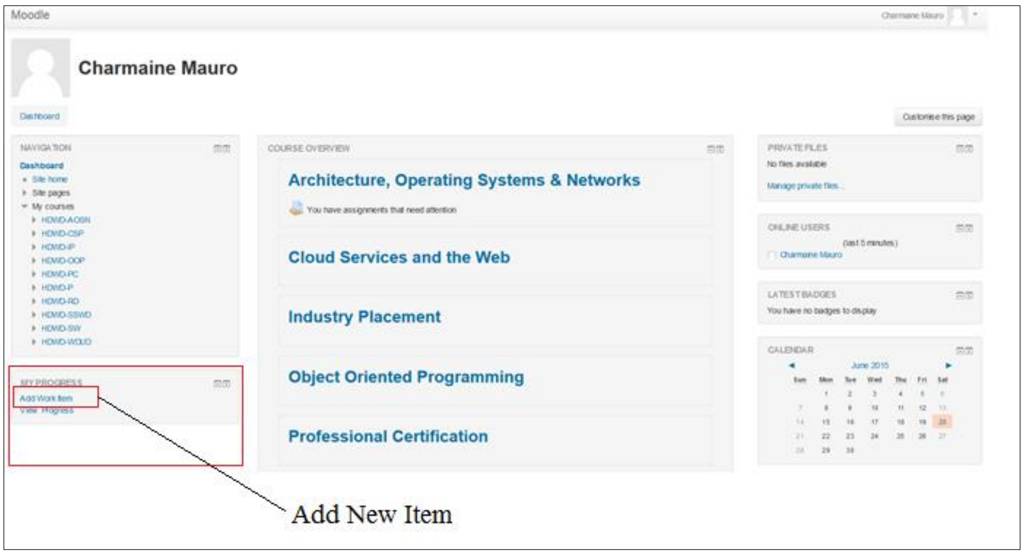
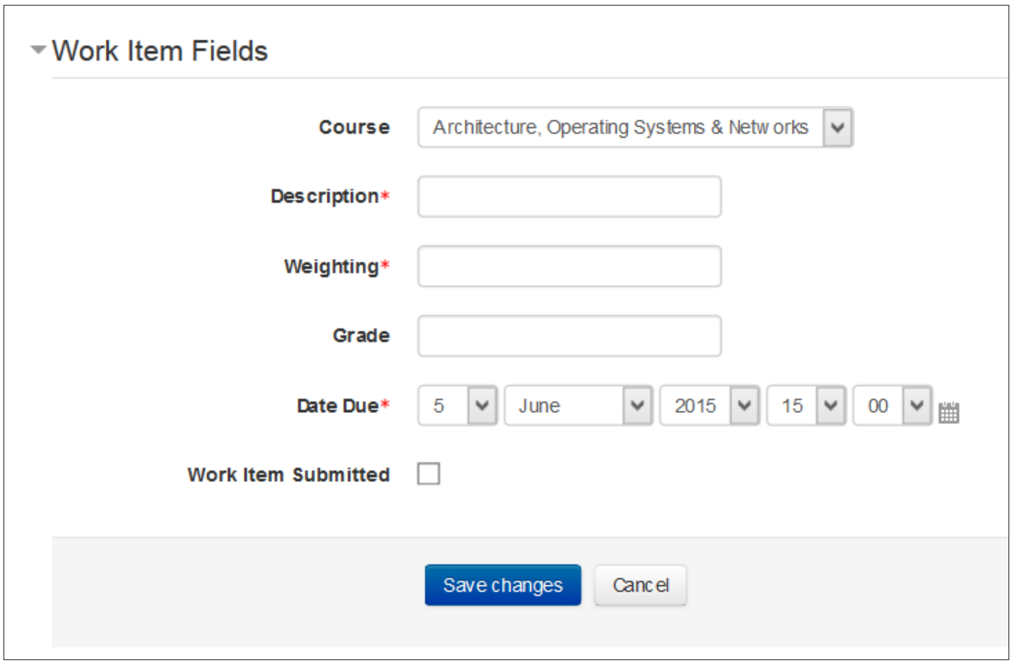


Figure 18 Add New Item

Once Add New Item is clicked it brings up the form below



Work Item Fields

Course	Architecture, Operating Systems & Networks
Description*	
Weighting*	
Grade	
Date Due*	5 June 2015 15 00
Work Item Submitted	☐

Figure 19 New Item Form

Figure 8 shows the ‘Add New Item’ form, it lists the courses that a student is enrolled in, and requires a description, weighting and due date. Here is where the form controls become important in view.php. If the user clicks cancel they should be redirected back to the view page. If the user hits save changes the form entries are saved to the database.

Charmaine hits cancel and is brought back to /my which is what we should expect.

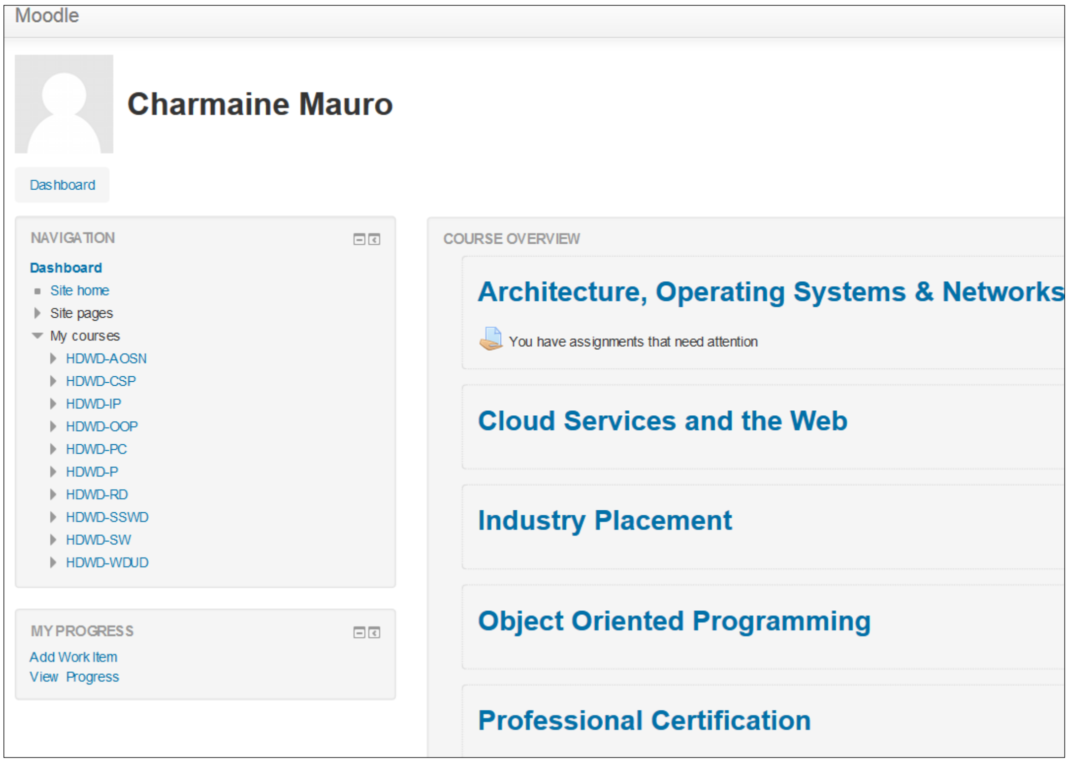
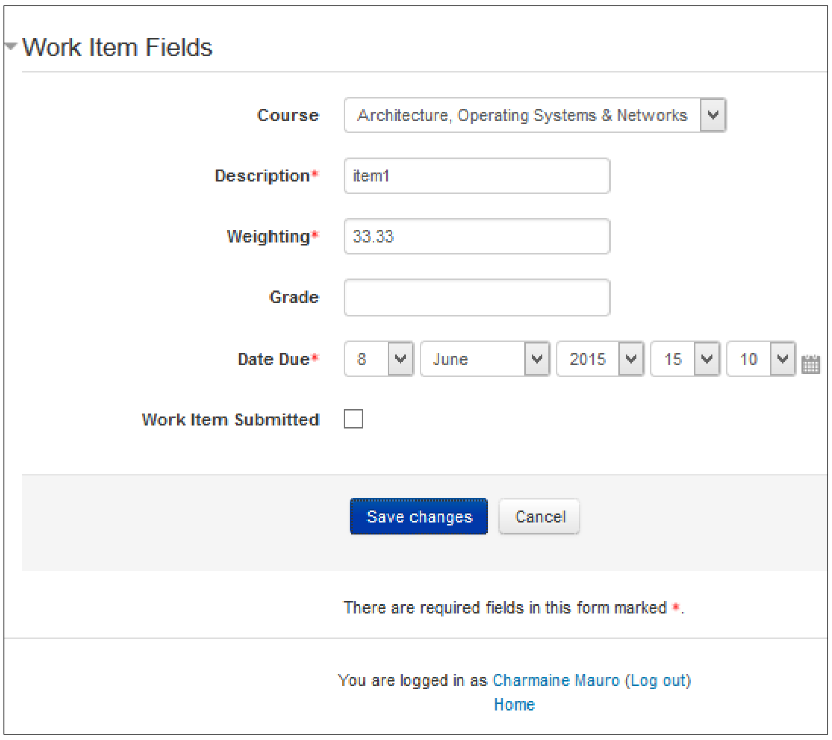


Figure 20 Re-direct to /my page

Next Charmaine adds an item. She clicks on the Add Work Item link again and the form populates. This time she saves the changes made to the form.


Work Item Fields

Course	Architecture, Operating Systems & Networks
Description*	item1
Weighting*	33.33
Grade	
Date Due*	8 June 2015 15 10
Work Item Submitted	☐

There are required fields in this form marked *.

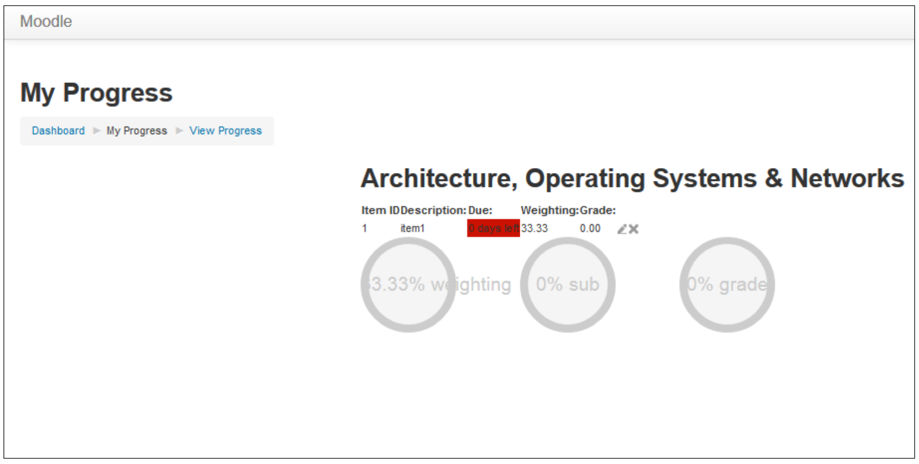
You are logged in as Charmaine Mauro (Log out)
Home

Figure 21 Save Changes

As you can see from figure 10 above Charmaine has chosen to add a work item to here Architecture, Operating Systems and Networks course, she has given a description of ‘item1’, she has given a weighting[2] of 33.33 that is due on 8th June, item submitted has been left unchecked.

Once Save changes has been clicked, Charmaine is re-directed to progress page where she can see the item she has just added.

[2] weighting is noted as an ambiguous term that would need to change. it is meant to refer to the % of the overall grade for a given course.


Moodle

My Progress

Dashboard ▸ My Progress ▸ View Progress

Architecture, Operating Systems & Networks

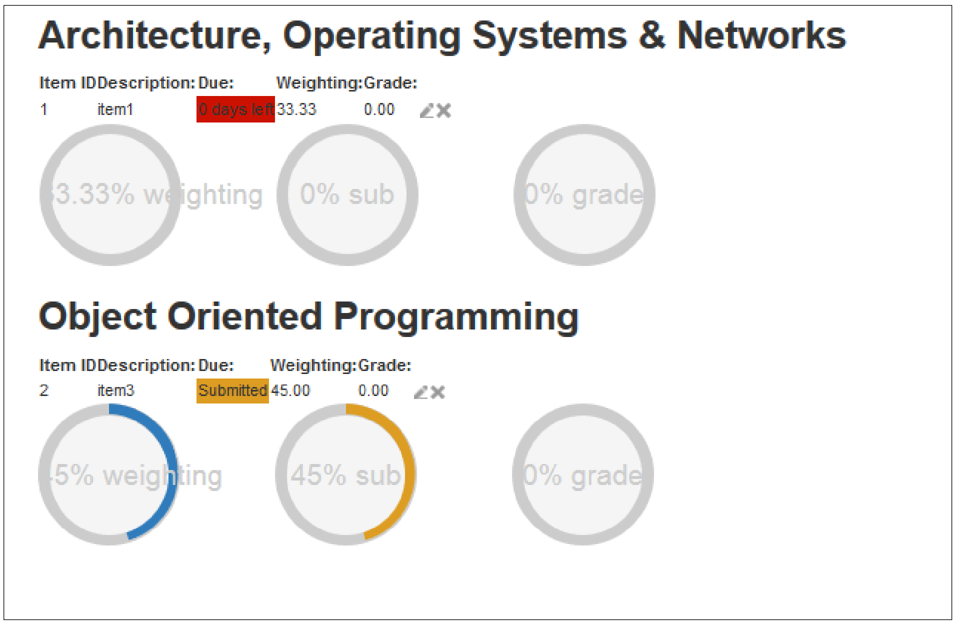
Item ID	Description:	Due:	Weighting:	Grade:
1	item1	0 days left	33.33	0.00

Figure 22 First view of my Progress

As figure 11 above denotes 3 radial circles, one that depicts % of weighting. In the centre of the first circle 33.33% weighting has been shown. However this should also be reflected by the radial filled with 33.33% of the chosen colour. Also as there were conflicts with the headers the footer is not shown in this page. A knock on effect of this is that Charmaine cannot logout from this page. The implications of this will be discussed in the next section.

User scenario 2: charmaine adds another item

Charmaine adds another item, this time for a different course with a weighting of 45% and has ticked the item off as submitted. Now we can see how this reflects below


Architecture, Operating Systems & Networks

Item ID	Description:	Due:	Weighting:	Grade:
1	item1	0 days left	33.33	0.00

Object Oriented Programming

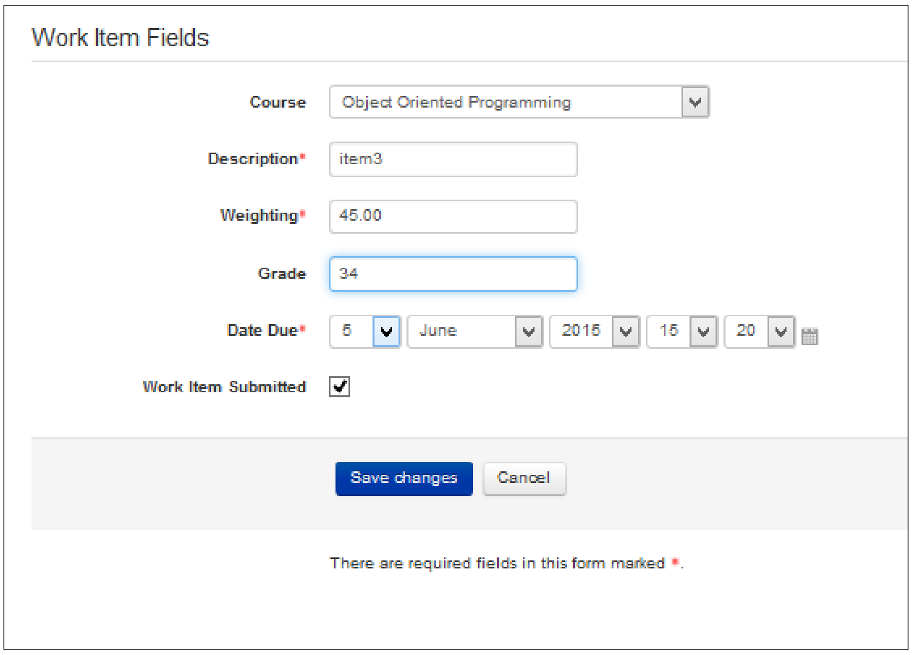
Item ID	Description:	Due:	Weighting:	Grade:
2	item3	Submitted	45.00	0.00

Figure 23 Radial Progress View

Notice from Figure 12 that the radials move in relation to the percentage that has been entered. Also submitted in the due field is now orange to reflect the colour of the submitted radial.

User Scenario 3: Charmaine edits an item

Charmaine edits the item to update with a grade that she has received. She is redirected to edit.php which populates the field names



Work Item Fields

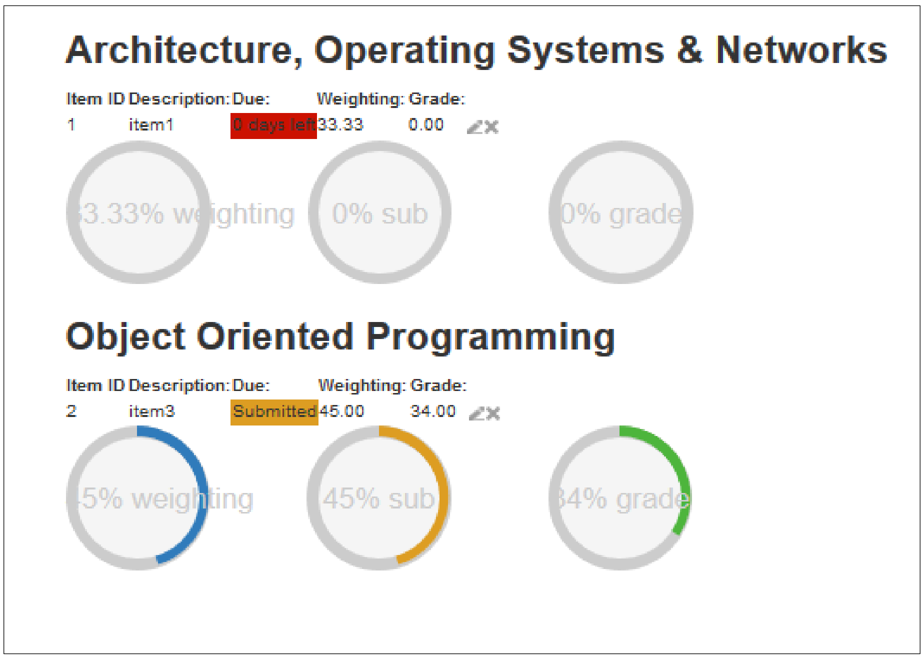
Course	Object Oriented Programming
Description*	item3
Weighting*	45.00
Grade	34
Date Due*	5 June 2015 15 20
Work Item Submitted	✓

There are required fields in this form marked *.

Figure 24 Edit form

Charmaine changes the grade to 34%. Clicks Save Changes. And should be redirected back to the progress.php where she can the updated changes reflected[3].

[3]Note the redirect has been commented out in the edit.php script. This was not fixed before the demonstration.



Architecture, Operating Systems & Networks

Item ID	Description:	Due:	Weighting:	Grade:
1	item1	0 days left	33.33	0.00

Object Oriented Programming

Item ID	Description:	Due:	Weighting:	Grade:
2	item3	Submitted	45.00	34.00

Figure 25 Update changes reflected

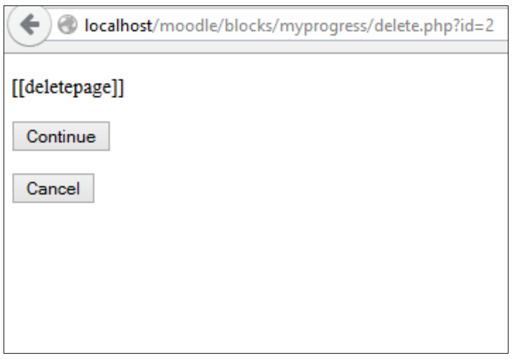
Now you can see from figure 14 that the changes have been reflected in the progress.php page[4].

User scenario 4: Charmaine deletes an item

Charmaine wants to delete the Object Oriented Programming item listed above. She clicks on the X icon beside the item.

She is brought to delete.php where she is prompted to choose between Continue or Cancel.

[4] Debug error: it should redirect to progress.php but instead redirects to an empty page. This is due to the fact that the redirect has been commented out on the edit.php


localhost/moodle/blocks/myprogress/delete.php?id=2

[[deletepage]]

Figure 26 delete.php Confirmation Page

Charmaine changes her mind and chooses to Cancel the deletion. She is redirected to the progress.php page as expected. She changes her mind again and chooses to delete the same item. This time she clicks ‘Continue’. She is redirected to the progress.php where the Object Oriented Programming item is no longer visible and has been deleted from the database.

User scenario: Another user logs in and enables the My Progress block.

This test was carried out to ensure that no other user can view another user’s progress. This is ensured by capturing the userid as a hidden form field and specifying the user id using Moodle’s global associative array $USER variable in the progress.php mentioned in chapter 4. Login as Shandon. Add the block by customising the page and adding the My Progress block described in the previous chapter. Click on the View My Progress link.

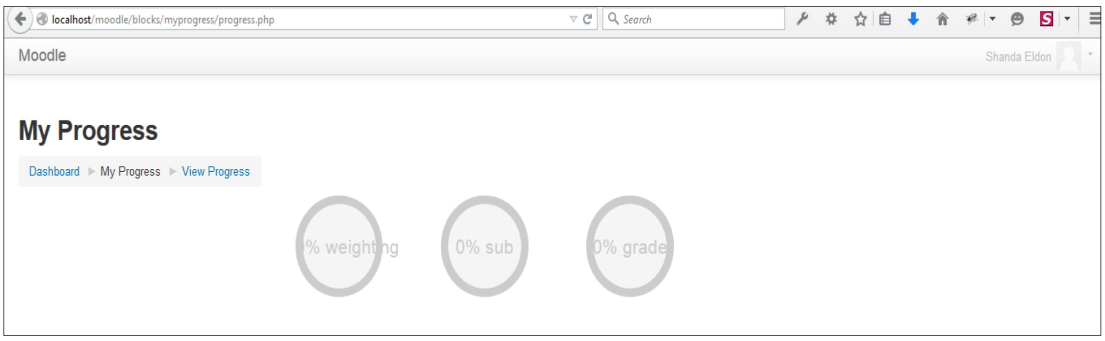

localhost/moodle/blocks/myprogress/progress.php

Figure 27 Verification of unique userid

As you can see from Figure 27 above, Shandon Elton has logged in and there are no items added which is what is expected.

6. 3 Evaluation

There were lots of issues that were raised in the testing stage that could not be fixed before demonstration day. I briefly discuss each of these specific issues below.

6.3.1 Display Issues

As I readjusted a CSS found online I was unable to figure out how to adjust the size of the font within the circles before demonstration day. This resulted in the text appearing outside the width of the circles which was not intended. Also the colours used to depict the different stages were not ideal. Again this was due to using a style sheet I had not created by myself. I had intended the weighting colour radial to be the same colour as the due date field when an item had been submitted. Also it was discovered that the cache was not purged after the language string for the delete page was added. This is the reason why it appears in double brackets in Figure 26 above. Also this confirmation box should include a Moodle page in the background but due to issues with headers discussed previously this was not possible before demonstration day.

6.3.2. Logic Not Implied

Upon testing I realised that the weighting radial does not update if the submitted button has not clicked also. This is due to the

```
if($myprogresspage->submitted == 1){
        $weightingsubmitted += $myprogresspage->weighting;
        }
```

conditional statement in progress.php. This was an oversight that should have been corrected before demonstration day. Also the grade radial does not update unless the submitted button has been clicked. However this is not made aware to the user and would cause confusion. At the very least there should have been tool tips beside each of the form fields to describe the action taken and description of each of the form fields. Also entering limits on the maximum percentage a user can enter should have been implied in the logic or there could have been a max and min value rule entered into the form elements that require a numerical value.

6.3.3. Navigation Issues

In progress.php $display = $OUTPUT->heading('heading'); had been commented out due to the fact that it was conflicting with CSS as it appeared at the bottom of the page. This I believe has something to do with the float element in the CSS page although this could not be fixed before demonstration day. The footer had also been removed which resulted in the page not resembling the look of a standard Moodle page.

6.3.4 Other issues

The text and table formatting were not intended to look like that. This is due to the fact that the text and tables were not targeted by CSS before demonstration day. As this was taken from the proof of concept described in previous chapters where the overall look of the tables was no considered, there was little time to consider this before demonstration day.

6.4 Comments

The testing stage led to reveal some obvious flaws of the system design. Unfortunately these issues would be easy to fix but could not be fixed before demonstration day. I discuss some of the issues raised here and some overall issues of the block in the final chapter now.

Chapter 7. Issues and Conclusions

Overview:

In this final chapter I discuss some of the issues that have been raised throughout the chapters in this document and try to assess how these could have been avoided. I then move on to discuss the conclusions and future of this block.

7.1. Issues: How They Could Have Been Avoided

7.1.1 Familiarity with the Moodle_System

I believe one of the main reasons why the block didn't reach the stage it could have before demonstration day was a lack of familiarity with the Moodle system. I initially believed that because Moodle had a vast amount of documentation available online that it would simple enough to implement a plugin for. However the documentation can be difficult to follow at times. As Moodle has a new release each week and with that comes new documentation it was therefore difficult to understand what version of Moodle some of the documentation referred to.

7.1.2 A Consistent Methodology

Although the methodology I chose (i.e. to build a prototype as fast as possible) proved to the best possible solution for this plugin, I believe there were stages in the build that I could have paid more attention to. For example as I used a block template for the four standard files which was easy enough to follow, however when it came to adding forms there were things I didn't understand completely. As time was limited there were issues that could have been fixed quicker if I knew where the problem had arisen from.

7.1.3 Integration with Moodle's existing progress tracking components

As mentioned in Chapter 2 when referencing the existing plugins. They use some of the core components of Moodle for tracking progress, grading… I believe this block can integrate these components in some form or another. For example there could be a form field that lists all the assignments that a student has to complete that have been set by each of their teachers. If a teacher has set a weighting for the assignment this could be automatically generated in the weighting field. Likewise if the teacher has set a due date for the assignment this could also be auto-populated. Also when it is graded it will auto-populate the grade field of an item. However I don't think the block should rely on teacher input alone - as this block allows student to control their input. As the input by student doesn't affect the grade and is used for a student's reference only perhaps integration with existing components mentioned may have drawbacks that outweigh the benefits.

7.1.4 Separation of concerns: Partially fulfilled

One of the issues mentioned in the previous chapter is that myprogress_edit_form was needlessly added. In fact the myprogress_form is the exact same form that could have been required in the edit.php. I believe this came from trying to replicate that separation of concerns that were followed in the view.php and myprogress_form.php scripts. Also a lib.php script was created and is found in the myprogress folder. This was initially planned to hold the layout of the progress.php elements. This was to be done with the use of a html writer. However as this technology was new to me I could not figure out how to use it within the timeframe. A better data model could have been followed that kept to a stricter Object Oriented Approach.

7.1.5 Back-ups, version compatibility for Moodle

This ties into the issue of familiarity with Moodle's system mentioned above but ideally a back-up should be created of the plugin. What this entails exactly is unclear at present and further research is required. Also the block I created may not be compatible with prior versions of 2.9 as there could be conflicts with Moodle's form API that was only developed as of version 2.9. Further research is required to understand version compatibility issues that might need to be addressed before this plugin can be offered to Moodle.

7.1.6 Understanding what CSS can do over JavaScript

As mentioned in Chapter 4, understanding how CSS can be used over JavaScript was an important learning curve. I misunderstood JavaScript to be the only technology that can be used to reflect a change in behaviour. I assumed that this behaviour meant a change that was occurring in the PHP script. However after realising that this change referred to client side behaviour and that CSS could be used instead meant that I didn't have to rely on JavaScript to carry out the radial progress indicators. Although in theory I thought I understood this, in practice I didn't fully understand how this worked. This was the first time that I had to use JavaScript in AMD and it was the first time I heard of technologies such as NodeJS and NPM. These appeared as technologies that would take considerable amount of time to understand that could not have possible within the timeframe. Fortunately CSS could be used in its place but a major feature of the block would have been compromised if this was not possible. A full understanding of the technologies that are required to implement my design should have been considered before building a design around specific features such as radial progress bars.

7.1.8 Lack of time left for testing

Unfortunately enough time for testing was not afforded. This resulted in the overall appearance of the block looking shoddy. The text not fitting within the radials for example, or the text in the tables not targeted by CSS gave a really disjointed look and spacing out of the work items. Also the links and page headers and footers not showing that appeared in the testing stage makes the progress.php page lose the consistency of a Moodle page appearance. If more time was afforded to testing, these issues could have easily been fixed.

7. 2 Concluding Remarks and Future Work

Summarising the issues that have been raised above and throughout the chapters reflects the negative aspects of the block where if careful planning had been carried out some of these issues could have been avoided. Allowing enough time for testing could have resulted in some of the issues of the block to be resolved quickly enough. Having a greater understanding of Moodle's system would have been invaluable when issues arose. However while these issues are important to reflect on it is also important to reflect on what I learned from this project. This was my first experience of developing something for an existing system and it was interesting to understand what this entailed. Although the block didn't turn out exactly as I had envisioned I have still learned a lot from different stages in the development. Although I was unable to use NodeJS or NPM in the project at least I know enough about them to perhaps use them in the future, if given more time. Also, although I would have gained experience in different ways by developing a stand-alone program from start to finish, I believe by choosing a plugin I was certain to gain different experience by developing for an existing system, such as understanding the key constraints that are involved.

I appreciate the Open Source community of Moodle and would like to return the favour of the extensive information that each person has contributed to its documentation and development. I wish to continue to fix the issues with this block and when satisfied that it might reach Moodle standards, send it in to them. If it is not up to the standards at least it might encourage the development of a block similar to mine that allows a student to have a holistic view of their progress, rather than a picture just based on one of their courses. This was the main objective of the project that I hope will be seen to fruition.

References

[1] D. Randy Garrison (2011) *E-Learning in the 21st Century: A Framework for Research and Practice* Taylor and Francis : New York

[2] David J. Nicol and Debra Macfarlane (2015) Dick *Formative assessment and self-regulated learning: a model and seven principles of good feedback practice* Studies in Higher Education, 31: 2, 199-218

[3] OECD/CERI International Conference *Learning in the 21st Century: Research, Innovation and Policy* (available on at http://www.oecd.org/site/educeri21st/40600533.pdf)

[4] Moodle Philosophy (available on at https://docs.moodle.org/24/en/Philosophy)

[5] Progress Bar Downloaded Zip File and all documentation available at https://moodle.org/plugins/view/block_progress

[6] FN-My Progress Downloaded Zip File and all documentation available at https://moodle.org/plugins/view/block_fn_myprogress

[7] Moodle installation documentation available at https://docs.moodle.org/22/en/Installing_Moodle

[8] Moodle architecture overview available at https://docs.moodle.org/dev/Moodle_architecture

[9] All relevant information regarding blocks can be found at https://docs.moodle.org/dev/Blocks

[10] All relevant information on Moodle's XMLDB editor can be found at https://docs.moodle.org/dev/XMLDB_editor