|  |  |
| --- | --- |
| Document Information | |
| Document name | Solution design document |
| Module Name | Lecturecast Connector |
| Project Name | Lecturecast Upgrade |
| Project Number |  |
| Project Account Code | P150558-00 |
| Author | Delvon Forrester |
| Issue Date |  |
| Status | Work in Progress |
| Version | 1.0 |

|  |  |  |
| --- | --- | --- |
| Document History | | |
| Version | **Date** | **Summary of change** |
| 0.1 |  | First Draft |
|  |  |  |
|  |  |  |

This document may include information that is **CONFIDENTIAL** and shall not be disclosed outside UCL and shall not be duplicated, used, or disclosed in whole or in part for any purpose other than to evaluate and implement procedures defined within this document.

CONTENTS

[1. Purpose 3](#_Toc502739527)

[2. Scope 3](#_Toc502739528)

[2.1 Interfaces 3](#_Toc502739529)

[2.2 References 3](#_Toc502739530)

[3. Requirements 4](#_Toc502739531)

[3.1 Business Requirements (Functional and Non-Functional) 4](#_Toc502739532)

[3.2 Technical Requirements 4](#_Toc502739533)

[3.3 Assumptions 6](#_Toc502739534)

[4. Application Architecture 7](#_Toc502739535)

[4.1 Application structure diagram 7](#_Toc502739536)

[4.2 Application structure 8](#_Toc502739537)

[4.2.1 Lecturecast connector 8](#_Toc502739538)

[4.2.2 Lecturecast Connector Batch 10](#_Toc502739539)

[4.2.3 MOODLE 11](#_Toc502739540)

[4.3 Sequence diagrams and or activity diagram. Show the communication pattern as a UML sequence diagram between applications. 11](#_Toc502739541)

[4.4 Application Function Matrix 12](#_Toc502739542)

[4.5 Application Collaboration 12](#_Toc502739543)

[4.5.1 INT001 12](#_Toc502739544)

[4.5.2 INT002 13](#_Toc502739545)

[4.5.3 INT003 13](#_Toc502739546)

[4.5.4 INT004 13](#_Toc502739547)

[5. Data Architecture 14](#_Toc502739548)

[5.1 Data Structure 14](#_Toc502739549)

[5.2 Data system Matrix – CRUD matrix 14](#_Toc502739550)

[5.3 Data/Entity flow between interfaces 14](#_Toc502739551)

[6. Technology Architecture 15](#_Toc502739552)

[7. Implementation Details 16](#_Toc502739553)

[8. Open Issues 22](#_Toc502739554)

# Purpose

Solution design document outlines the design and provides an overview of the business and technical requirements. It describes the high level and low level design approach, data flow for the solution, discreet services orchestrated in the flow and deployment scenario for the solution.

# Scope

This document covers the solution design and implementation of Lecturecast Connector. The connector is created as block in Moodle. The document also covers settings required in Moodle to create users in ALP. It also covers the Moodle batch job that keeps the data in Moodle and ALP in sync and Testing that is created in Moodle to test the Moodle block.

## Interfaces

The document covers the following interfaces

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Interface Number | Interface Provider System | Consumer System | Type of Interface (WS, DB View), push or pull | Frequency (batch job, real-time) | Data transfer (bulk, single record) |
| INT001 | ALP | Lecturecast Connector | Web service, push | Real-time | Single record |
| INT002 | Moodle | ALP | push | Real-time | single record |
| INT003 | ALP | Lecturecast connector batch | Web Service | Batch job | Bulk |
| INT004 | Lecturecast Scheduler | Lecturecast Connector | DB Table | Real-time | Bulk |

## References

This document must be read in conjunction with the HLD and the links listed below;

|  |  |
| --- | --- |
| Name | Link |
| ALP User Interface | https://echo360.org.uk/ |
| ALP API | <https://echo360.org.uk/api-documentation> |
| Wiki guide | <https://wiki.ucl.ac.uk/display/LecturecastResourceCentre/Linking+Moodle+to+Lecturecast> |
| Jira | https://ucldata.atlassian.net/ |
| Lecturecast HLD | [https://liveuclac.sharepoint.com/sites/ISD/DigitalEducationDomain/LectureCast/Shared%20Documents/Technical%20Documents/Design/HLD\_Lecturecast\_phase1.docx?d=w8f47f488ac1a419db02d45259117b2dd](https://liveuclac.sharepoint.com/sites/ISD/DigitalEducationDomain/LectureCast/Shared%20Documents/Technical%20Documents/Design/Lecturecast-HLD.docx?d=w8f47f488ac1a419db02d45259117b2dd) |

# Requirements

## Business Requirements (Functional and Non-Functional)

|  |  |
| --- | --- |
| Req. Number | Requirement Details |
| BR01 | The block in Moodle should have a title that clearly states what the block is |
| BR02 | The block should only be available to users who have editing capabilities |
| BR03 | The block should only be made available when editing is on |
| BR04 | The block should only be available for access when logged in to Moodle |
| BR05 | The block should only be available for access when logged in to a course |
| BR06 | The block should check whether the course is linked to ALP sections and if so display the section numbers but if not display there is no link |
| BR07 | The block should allow users to make any number of links to ALP sections |
| BR08 | The input form for ALP section linking request should use autocomplete (predictive search) after three characters are pressed |
| BR09 | The form should query a dataset (to be determined) of all section numbers and use this for the predictive search |
| BR10 | The user should be able to select a section number from a filtered list when creating mappings to link to this specific section in ALP |
| BR11 | After creating mapping, the user should be redirected to the original course and the block should display the mapped sections to the course |
| BR12 | The instructor /admin should be able to create/unlink mappings from ALP |
| BR13 | There should be the ability to store the client Id, client secret and LMS course Id from ALP for the institution so that if these change they can be added in the settings |
| BR14 | There should be the ability to store the database credentials from where the dataset (sections) will be pulled from for predictive search |
| BR15 | There should be a functionality to reposition the block in courses |
| BR16 | The block should still display the sections mapped to even if the course shortname is changed after initial mapping done to the section |
| BR17 | The block should only be made available on all course main page |
| BR18 | The block should be made available to all courses and any new course automatically without having to install it |
| BR19 | When a section in unlinked in ALP UI, it should be reflected in the block for that course that the section is not displayed as mapped |
| BR20 | If a section is manually linked in the ALP UI then that should be reflected in the block for that course that the mapping is created |
| BR21 | The plugin should store the current term so that the dataset can be managed in future academic years (terms) |
| BR22 | Accessing Lecturecast via Moodle should trigger user creation process in ALP if the user does not already exist in ALP. |

## Technical Requirements

|  |  |  |
| --- | --- | --- |
| Req. No | Details | Mapping to BR |
| TR01 | The block was given a title of ‘Lecturecast Connector Block’ which was added as a string in the language file | [BR01](#BR01) |
| TR02 | This must be implemented by calling the has capability function or the user\_is\_editing function in the blocks get\_content function | [BR02](#BR02) |
| TR03 | This must be implemented by calling the user\_is\_editing function in the blocks get\_content function | [BR03](#BR03) |
| TR04 | The block should be developed as a Moodle block plugin and require login to course. All supported PHP files should implement the require login | [BR04, BR05](#BR04) |
| TR05 | Add a table for the block in Moodle and save the course information and section information when a link is made and this table should be queried to see if a section is mapped to the course. | [BR06](#BR06) |
| TR06 | REST web service should be used to call ALP sections API and create a mapping using the section number, course shortname and section ID which is selected from the GEN Oracle Database and the shortname from the Moodle course | [BR07](#BR07) |
| TR07 | A new page with a form is created for users to be able to select the section they want to link to. The course information is passed to this form | [BR08](#BR08) |
| TR08 | The autocomplete will be implemented using JavaScript passing the keyword from the form to the PHP file that will query the database. The JavaScript keyup function will be used to capture the keypress for the keyword | [BR08](#BR08) |
| TR09 | A table in the GEN database will be used to hold the section details and this table will be queried to return the sections and their associated section IDs. | [BR09, BR16](#BR09) |
| TR10 | After pressing 3 characters the GEN table will be queried and a limit of 20 sections will be listed from which the user can select the section to be mapped. By clicking of create mapping the Moodle course will be mapped to the selected section via REST web service and ALP section API. | [BR10](#BR10) |
| TR11 | If the API returns an OK message that the section is mapped, the Moodle table will have inserted, the record of the mapping and then redirected to the course main page where the block will display the sections the course is linked to. | [BR11](#BR11) |
| TR12 | While the user is on the page with the form they will see any errors display (in red) in the section where the sections should be displayed. If error to access the GEN database, or the keyword does not return any sections it will display an error. | [BR10](#BR10) |
| TR13 | The block will have a test folder that holds unit testing code and acceptance testing code. The units tested are testing for the existence of the Moodle table to hold the mappings, connection to the GEN database and connection to the ALP API. The acceptance testing is done using behat testing to test the user journey to create a mapping (but not making the mapping but cancelling and returning to the course main page) |  |
| TR14 | Where possible the moodle recommended functions are used such as get\_records, require\_login, required\_param etc. |  |
| TR15 | Comments should be added to the Moodle standard for functions and complex functionality should have comments explaining what the code is expected to do. |  |
| TR16 | ALP allows a many to many relationship between the Moodle shortname and the section. When making a link store all shortnames linked to that section and append the current shortname to the array (lmsCourses) and also add the lmsProfileId | [BR07](#BR07) |
| TR17 | The Moodle table storing the mappings does not use a unique key for the course the sections are mapped to and neither for the section so that it can store multiple sections for the course. It uses a numeric auto-increment primary key and searches on the course id to display all sections mapped. | [BR07](#BR07) |
| TR18 | A settings page to be added that will only be accessible by site admins who will set the credentials for the API, LMS Profile and the GEN table so that if these are modified in GEN or ALP then they can be easily updated in the settings. The setting will also hold the position that the block must be positioned on all courses and a functionality to reposition the blocks if needed. | [BR13, BR14, BR15](#BR13) |
| TR19 | In the get\_contents function, wrap the functionality of the block that returns content with a page type condition set to 'course-view'. Also add the block to the site home and make it available across the site, then form any course edit the block configurations and set it to only display on course main pages. | [BR17](#BR17) |
| TR20 | The block should be hooked to a Moodle scheduled task which will compare the Moodle table with the mappings and ALP section links to remove any link that was unlinked in ALP and add any links that were added manually in ALP to the Moodle table so that it is reflected in what the block displays. This scheduled task must be set to run as often as needed but will only run the script when Moodle CRON runs and pending a run. | [BR19, BR20](#BR19) |
| TR21 | A term column is added to the Moodle table that holds the mapping and it is updated when the script for the scheduled task is run to ensure it is the correct term in ALP. This way the dataset can be manipulated using the TERM in other academic years. | [BR21](#BR21) |
| TR22 | Once the block is installed on the server it then needs to be added on the site home page and made available to all the site. You would then go into any course and change the configuration to display on only course main pages | [BR18](#BR18) |
| TR23 | Once the mapping/link is created between the course and ALP section, the LTI activity of that course would take a user to the mapped section. If you do not have an account then one will be created with the details passed by Moodle. | [BR22](#BR22) |

## Assumptions

1. There will be a Moodle instance installed

2. The Scheduler should provide the section details from ALP, which it saves in one of its local tables.

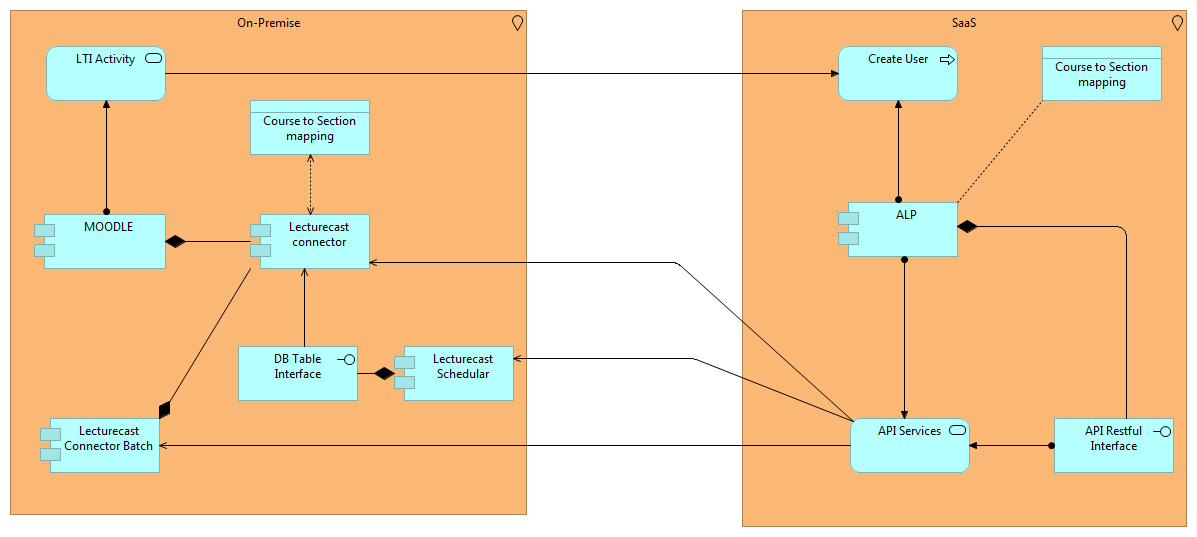
3. The API credentials will be added to ALP and provided to the block settings in Moodle.

4. The LTI activity in Moodle will be configured with the correct information from ALP.

# Application Architecture

## Application structure diagram

The following diagram gives an overview of the landscape of applications that are involved in providing the solution for Lecturecast connector requirements.



Application Catalogue:

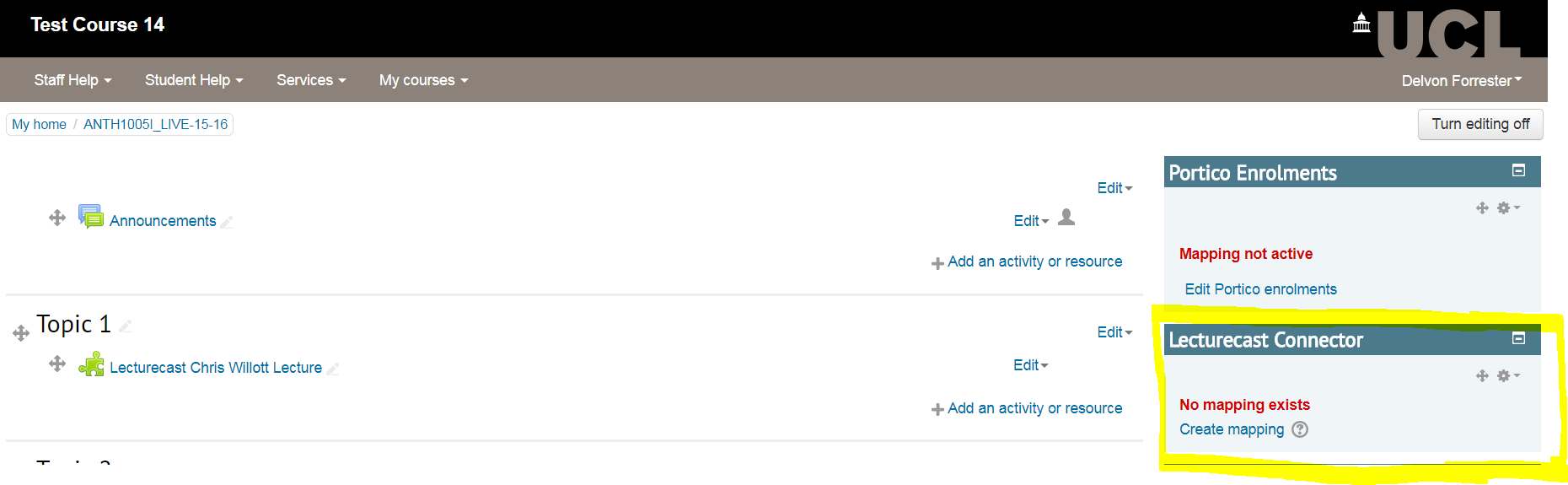
|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| LC01 | Lecturecast connector | Lecturecast Connector Block is a module/plugin that is added to the Moodle platform to provide additional functionality in Moodle. It is used by admin/instructor to create links between a Moodle course and an ALP section. It also displays all links along with the section names that are created for the current course. |
| MD01 | MOODLE | Modular Object-Oriented Dynamic Learning Environment (MOODLE) is a Learning Management System to provide an organized interface for e-learning, or learning over the Internet. This is where the courses are stored and the main interface that instructors and students use to access their learning. This interface is where the Lecturecast Connector Block is developed as a plugin to the main MOODLE. |
| AP01 | ALP | ALP - Active Learning Platform (ALP) is a cloud based application that controls the Lecture capture devices, records the Lectures and stores them in appropriate sections for students to be viewed via Moodle. It makes APIs available that we can call the different functions to create the link, to get the sections, links and terms to compare them with the database table in MOODLE that is added by the Lecturecast Connector Block plugin. |
| LCB2 | Lecturecast connector Batch | This is a scheduled task that synchronize data from ALP to Lecturecast connector. The functionality is developed in the Lecturecast connector but is triggered by MOODLE’s cron job. |
| LS01 | Lecturecast Scheduler | Scheduler is a Java application developed using Spring framework. It has a web interface by which the users can search for future and captured events. The scheduler provides the sections that are used in Lecturecast connector to create mappings in ALP. |

## Application structure

The Lecturecast connector application integrates the following components:

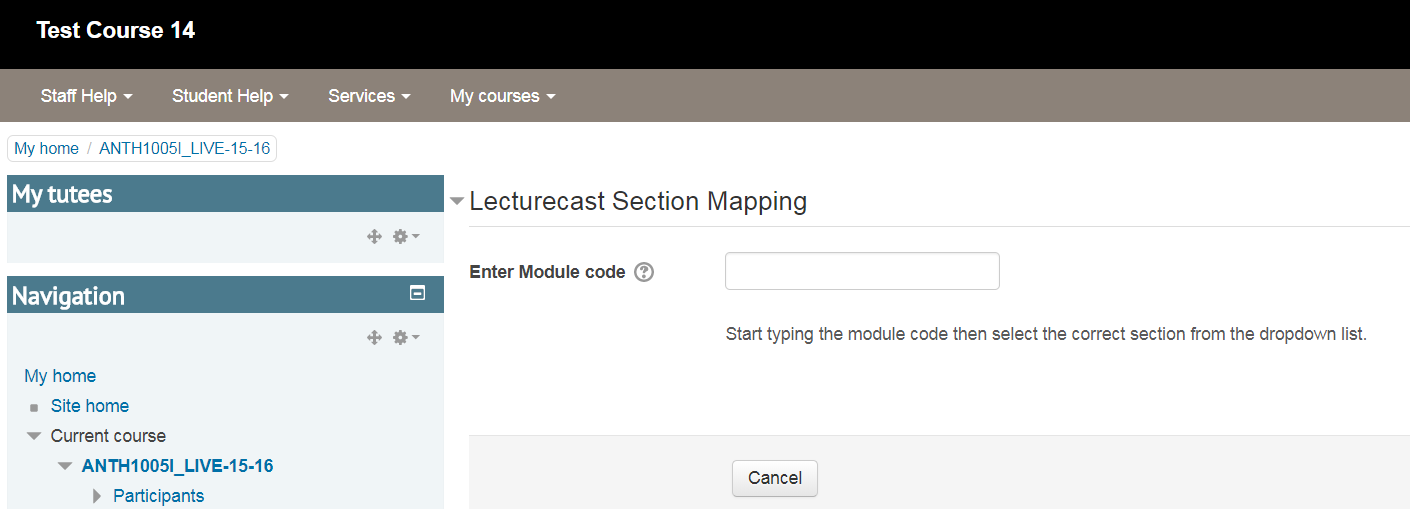
1. Lecturecast connector – (LC01 in Application catalogue).
2. The Active Learning Platform (ALP) - (AP01 in Application catalogue).
3. The Lecturecast connector Batch - (LCB2 in Application catalogue).
4. The Lecturecast scheduler - (LS01 in Application catalogue).

### Lecturecast connector

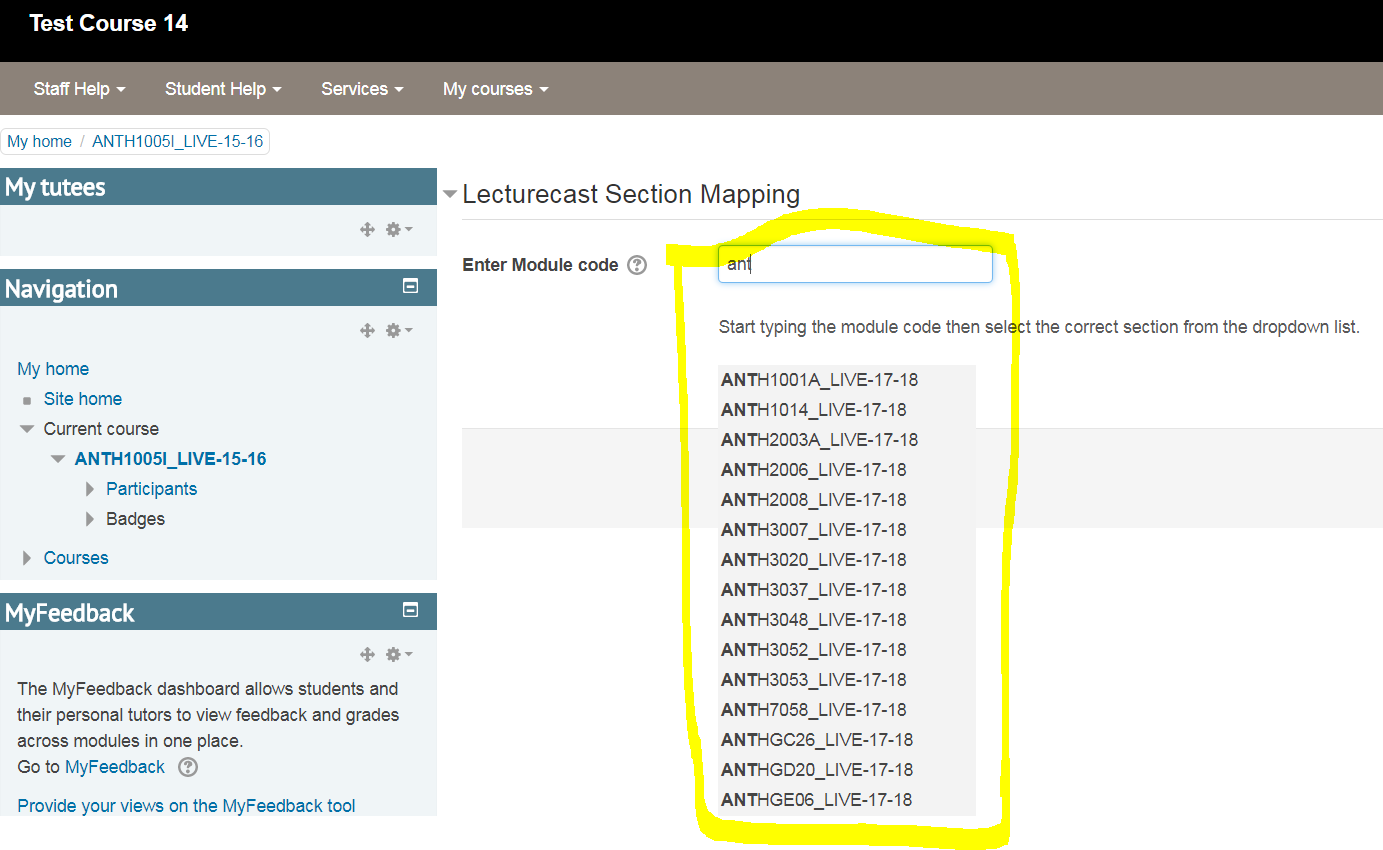


Here is a Lecturecast connector block within a MOODLE course highlighted in the screen capture above. It is displaying that no mapping exist after querying the Lecturecast Connector table if any record exists for this course (**4.5.1 INT001 get mappings**).

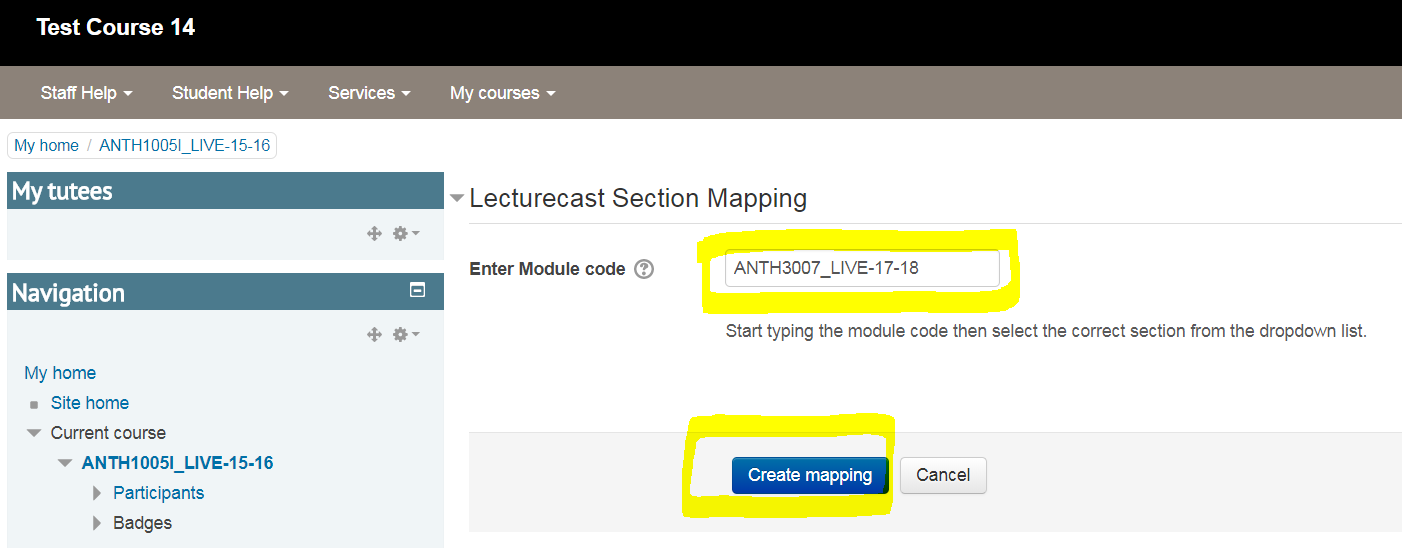
To create a mapping the instructor would click on the ‘Create mapping’. This will open up a new page to the instructor where they can enter the section to be mapped.



Above the user can start typing the module code that they want to map this MOODLE course to in ALP. Once the user types the third character the system will integrate with the Lecturecast scheduler which stores the section numbers and IDs that are in ALP (**4.5.4 INT004 return section details**). It will request sections based on what the instructor has entered and present a maximum list of twenty sections where the instructor can select which section they want to map to (see below)

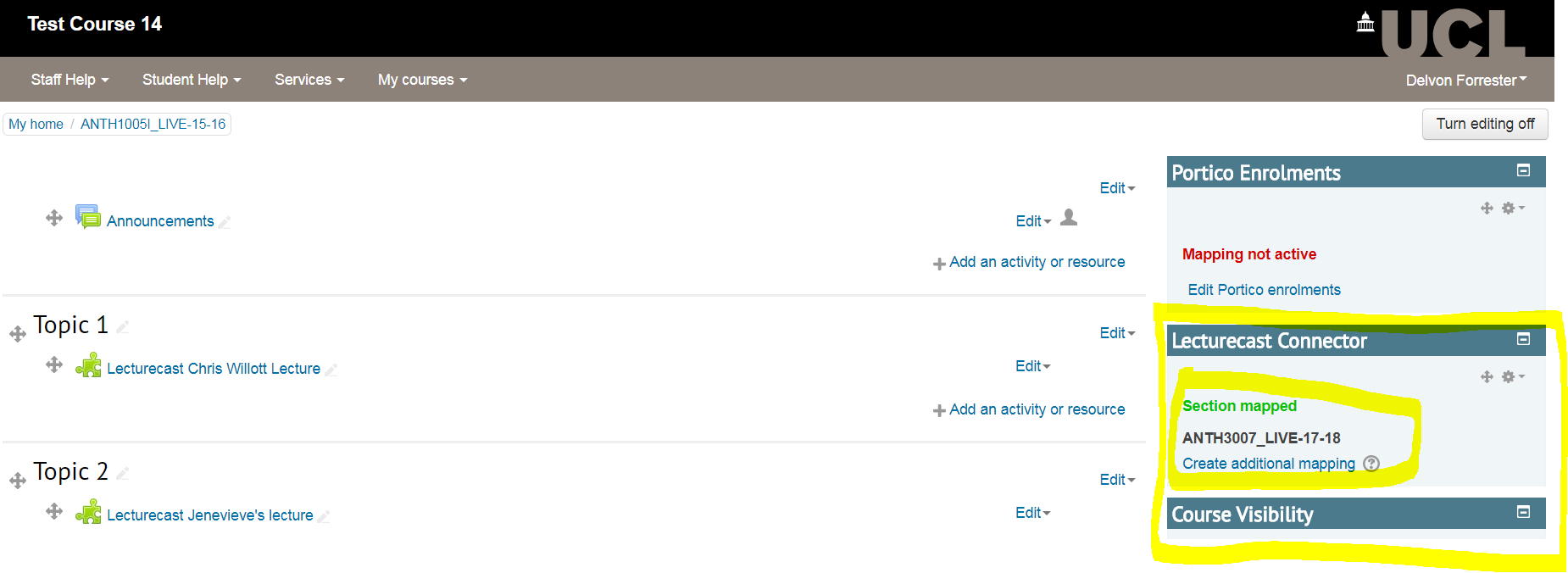


Once a section is selected it will be displayed in the input form and a create mapping button appears for the instructor to create the mapping (see below)



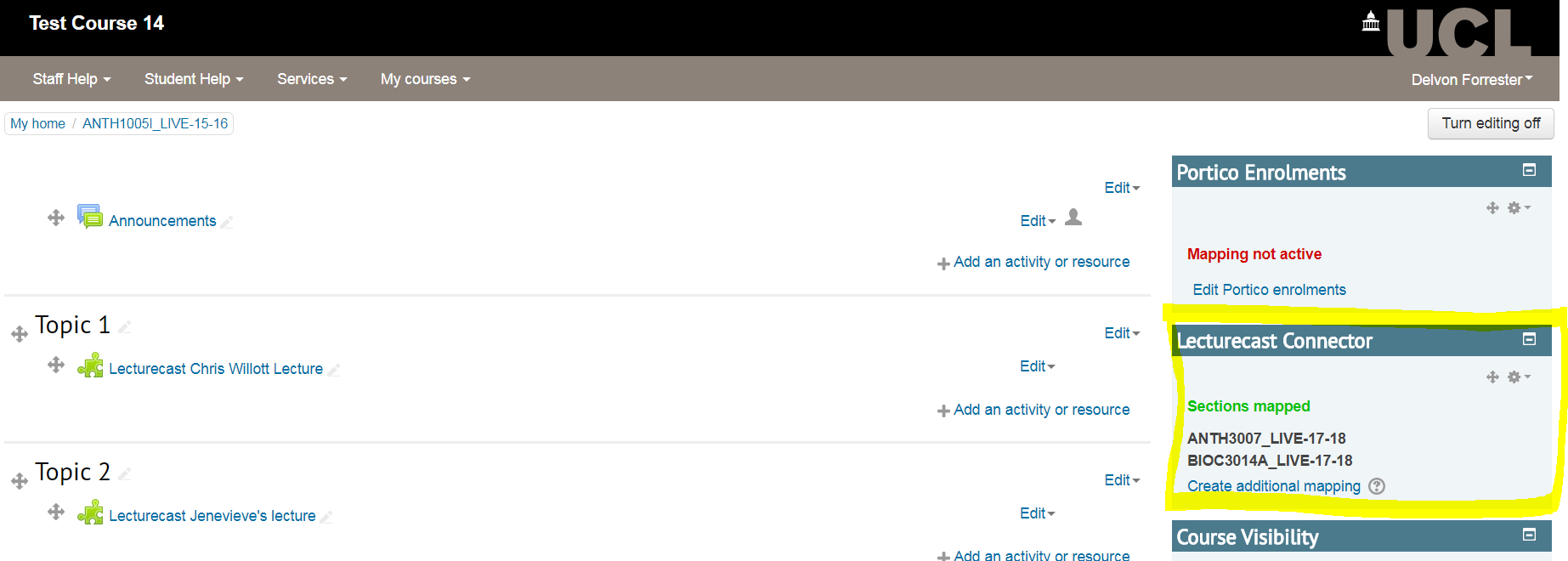
Once create mapping is clicked, a number of functions are done:

1. The MOODLE course shortname is checked to see if there is any white spaces before and after the name. If there is then it will be removed and the name saved back to MOODLE’s database as this name will be used in the mapping process.
2. The Lecturecast connector will call the ALP API services using the RESTful interface to:
   1. Get a token from ALP to be used for API calls (**4.5.1 INT001 authenticate access**).
   2. Call the sections endpoint to return the selected section by passing the ID returned from the Lecturecast Scheduler (**4.5.1 INT001 get\_sections**).
   3. Pass the MOODLE shortname and LMS profile ID to the Courses property of the section in order to create the link (**4.5.1 INT001 create link**).
   4. Insert the new link in the lecturecast connector table stored in the MOODLE database (**4.5.1 INT001 insert mapping**).
3. The instructor is then redirected to the course main page and a query is made to the lecturecast connector table to check mappings that are made for that course and display all mappings in the block (see below).



The section mapped to will be displayed and to make additional links the instructor needs to click on ‘create additional mapping’.

The instructor can create as many mappings as needed (see below).



### Lecturecast Connector Batch

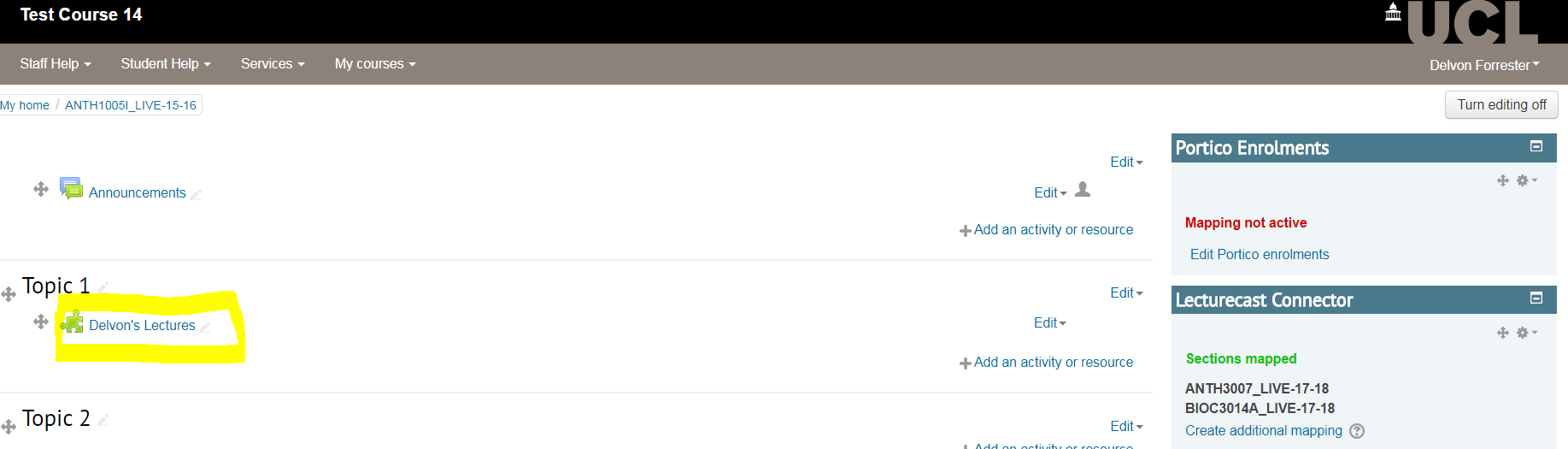
This batch job was created because instructors or admins have the ability to create mappings directly in ALP. If this is done then the Lecturecast connector does not have any knowledge of this being done. Therefore, when instructors create an LTI link from the course that they created the mapping for, then users will be able to click on that link and access lectures in that section in ALP, but within the Lecturecast connector block in the course it will not display that a mapping has been made with the course and the ALP section.

They can also delete mappings directly in ALP and that would not be reflected in the block within the course on Moodle.

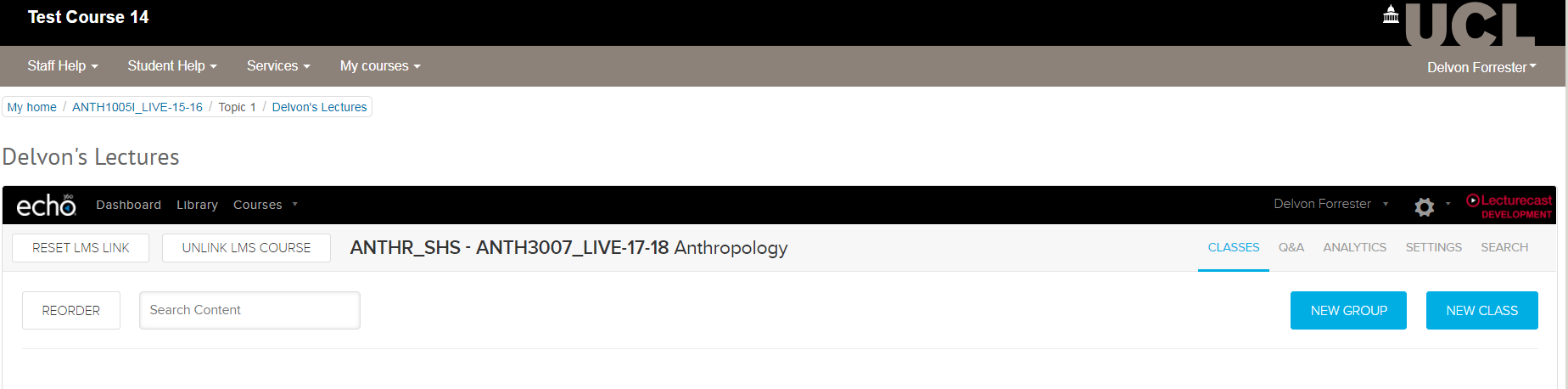
What the batch job is doing is syncing the Lecturecast Connector with ALP sections. What this does is use the ALP API functions listed at (**4.5.3 INT003**) to call the **get\_sections**, **get\_lti-links** and **get\_terms** functions and then compare these to the Lecturecast Connector Table in Moodle that is created and used by the plugin.

### MOODLE

The diagram above indicates that users are created in ALP when accessed through MOODLE and they do not have an account in ALP. This is through the LTI activity that is created in the MOODLE course and the user will then have a link to ALP which will give them automatic access to the section/sections mapped to that course by the instructor. (See **4.5.2 INT002**)



The lectures will be available in the section below if and when the instructor publishes them for the students.



## Sequence diagrams and or activity diagram. Show the communication pattern as a UML sequence diagram between applications.

The following sequence diagram shows the sequence mentioned above.

****

## Application Function Matrix

The following table lists the functional capabilities that the business wants and the application that provide these.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Functions | Connector | ALP | Batch | Moodle |
| Create Link between Moodle Course and ALP Section | Y | Y | N | N |
| Sync Lecturecast Connector and ALP | N | Y | Y | N |
| User access to ALP through LTI link | N | Y | N | Y |

## Application Collaboration

The following table provides an overview of the interfaces between ALP and Moodle Connector. These are high level interfaces which are made up from one to many requests. The details of the requests that make up an interface can be found in following section. The data entities that are transferred can be found in the data architecture section.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Interface Number | Source System | Consumer System | Type of Interface (WS, DB View), push or pull | Frequency (batch job, real-time) | Data is Replicated /Transient? | Data transfer (bulk, single record) |
| INT001 | ALP | Lecturecast Connector | WS, push | Real-time |  | Single record |
| INT002 | Moodle | ALP | push | Real-time |  | single record |
| INT003 | ALP | Lecturecast Connector Batch | WS, push | Batch job |  | Bulk |
| INT004 | Lecturecast Scheduler | Lecturecast Connector | DB Table, push | Real-time |  | Bulk |

### INT001

Individual Requests that made up INT001 are listed below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Request Name | Delivery Method | Input Parameter | Output Entities | Description |
| /oauth2/access\_token | Restful Web service | Client ID, Client secret, Grant type | ALP token | A string representing a token |
| /public/api/v1/sections/{sectionId} | Restful Web service | Section ID | ALP Object | An object with section properties |
| /public/api/v1/sections/{sectionId}/lms-course-ids | Restful Web service | Section ID, shortname, lmsProfile ID | ALP OK message | An OK message if successful |
| get\_records() | DB Table | Course ID | Array of sections | Section numbers the block displays |
| insert\_record() | DB Table | Course ID, shortname, section number, term |  | Insert the record of any mapping done |

#### Interaction

The interaction between the functions were mentioned in section 4.2.x above.

### INT002

Individual Requests that made up INT002 are listed below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Request Name | Delivery Method | Input Parameter | Output Entities | Description |
| Create user | LTI Activity | Email, firstname, lastname | User object | A user with firstname, lastname and email |
| Access lectures | LTI Activity | Email, firstname, lastname | User object | Giving user access to lectures in a section |

### INT003

Individual Requests that made up INT003 are listed below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Request Name | Delivery Method | Input Parameter | Output Entities | Description |
| /oauth2/access\_token | Restful Web service | Client ID, Client secret, Grant type | ALP token | A string representing a token |
| /public/api/v1/sections/{sectionId} | Restful Web service | Section ID | ALP Object | An object with section properties |
| /public/api/v1/terms | Restful Web service | Term ID | ALP Object | An object with term properties |
| /public/api/v1/sections/{sectionId}/lti-links | Restful Web service | Section ID | ALP Object | An object with Linked section properties |

### INT004

Individual Requests that made up INT004 are listed below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Request Name | Delivery Method | Input Parameter | Output Entities | Description |
| Return section details | DB table | 3 or more characters of the section name | Section ID, Section number | Array of multiple Section IDs, Section numbers |

# Data Architecture

## Data Structure

Lecturecast Connector only masters the link between the Moodle course and ALP section. To establish the link it uses the following entities:

* ALP Section – An ALP section is where lectures are stored once they are captured
* Moodle Course – Is an online series of lectures or lessons in a particular subject, leading to an examination or qualification.
* Term – for this purpose the term is the academic year that teaching is taking place for eg. 17-18 term.
* LTI Link – Learning Tools Interoperability (LTI) is a standard set of specifications for allowing third party learning applications (or "tool providers", in this case ALP) to integrate with Moodle.
* Course to section mapping - This is the mapping that is created between a Moodle course and ALP section. The Moodle course shortname is used from the Moodle side and the section Id from the ALP side to make the mapping.

## Data system Matrix – CRUD matrix

The following table shows which system is creating, updating and reading the appropriate data entities.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Lecturecast Connector | MOODLE | Scheduler | ALP | Batch |
| ALP Section | RU |  | R | CRUD | R |
| Moodle Course | RU | CRUD |  | CRUD | R |
| Term | R |  |  | CRUD | R |
| LTI-Link | R | R |  | CRUD | R |
| Course to Section Mapping | CRUD |  |  | CRUD | R |

## Data/Entity flow between interfaces

The following table gives details of the data that flows between systems via the requests/interfaces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Interface Number | Request Name | Input Parameter | Output Entities | Attributes (used in solution) |
| INT001, INT003 | Authenticate Access | Client ID, Client secret, Grant type | ALP token | access\_token |
| INT001, INT003 | Get Sections | Section ID | Section Object | id, sectionNumber, Lms\_courses (lmsProfileId, lmsCourseId, ltiLinkstoSection) |
| INT001 | Create Link | Section ID, shortname, lmsProfile ID | ALP OK message | String (‘OK’) |
| INT001 | Get mapping | Course ID | Array of sections | Section numbers |
| INT001 | Insert mapping | Course ID, shortname, section number, term | int | id |
| INT002 | Create user | Email, firstname, lastname | User object | Sections, classes |
| INT002 | Access lectures | Email, firstname, lastname | ALP object | Videos, schedules |
| INT003 | Get Terms | Term ID | Term Object | Id, name |
| INT003 | Get LTI-Links | Section ID | Array of LTI-links object | SectionId, lmsProfileId, lmsCourseIds, contextId |
| INT004 | Return section details | 3 or more characters of the section name | Section ID, Section number | Section\_id, section\_number |

# Technology Architecture

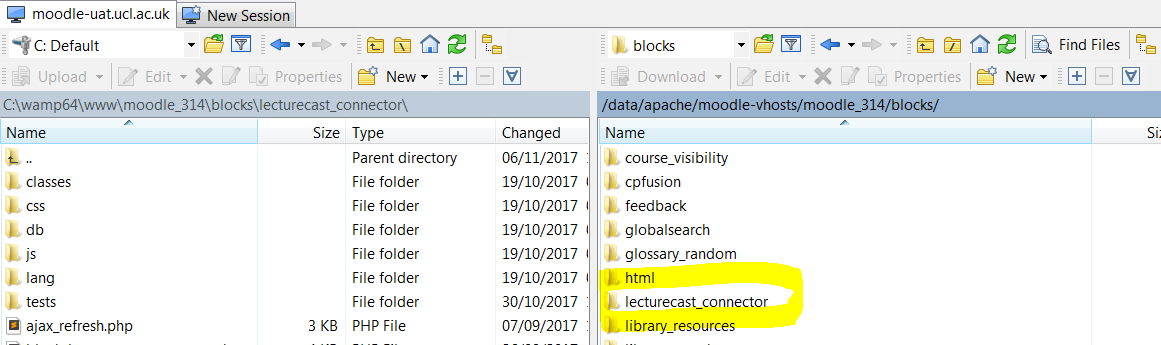
This section lists the technical components those need to be developed and the existing components those need to be modified to fulfil the technical requirements. The mapping to the identified low level technical requirement is also given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Component | Type | Deployed in | Description | TR |
| /blocks/lecturecast\_connector | Moodle Block | moodle.ucl.ac.uk | The directory where the code is stored | [TR22](#TR22) |
| lcast\_echo\_section\_map | Table | GEN DB | The table that the sections properties are returned from | [TR09](#TR09) |
| Add block on front page | Plug-in | Moodle | Setting done in Moodle to make block visible on the site | [TR22](#TR22) |
| Leccturecast\_connector block settings | Moodle Config | Moodle | Lecturecast LTI Tool Setup | [TR23](#TR23) |
| Leccturecast\_connector block settings | Moodle Config | Moodle | Lecturecast Configuration for API | [TR18](#TR18) |
| Leccturecast\_connector block settings | Moodle Config | Moodle | Gen DB setup | [TR18](#TR18) |
| block\_lecturecast\_connector | Table | Moodle DB | Lecturecast Connector Table to store mappings | [TR05](#TR05) |
| Leccturecast\_connector block settings | Moodle Config | Moodle | Lecturecast Connector block repositioning | [TR18](#TR18) |
| Sync Lecturecast connector table with ALP | Moodle Scheduled task | Moodle | Scheduling the time for the batch job for syncing table with ALP should be triggered by Moodle cron | [TR20](#TR20) |
| UNIT Testing | Plugin unit test | Lecturecast connector block | Code to test different units of the application | [TR13](#TR13) |
| Behat Testing | Plugin acceptance test | Lecturecast connector block | Code to imitate the functional test | [TR13](#TR13) |

# Implementation Details

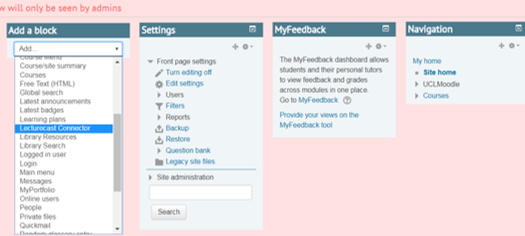
The implementation instructions with screenshots for different areas of the plugin are listed below:

1. Adding the code to the server

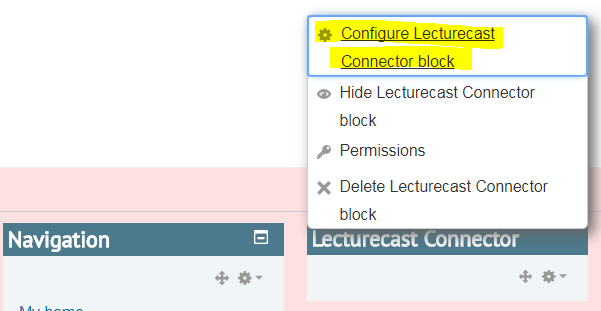


The code is added above to the blocks folder of the server your Moodle site is on. As highlighted you install the main folder ‘lecturecast\_connector’ to the blocks directory.

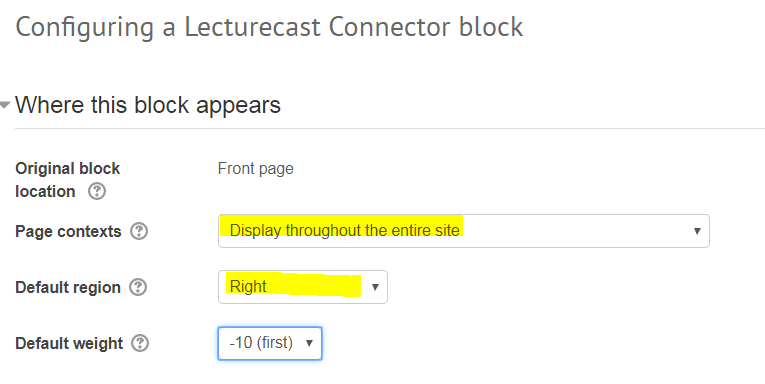
1. Adding Block to the Site



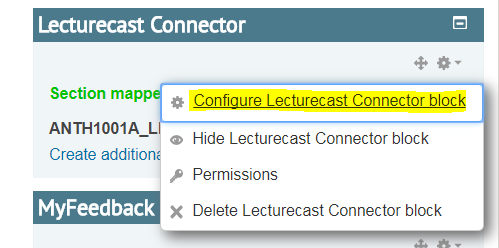
With editing on, add the block to the site home page.



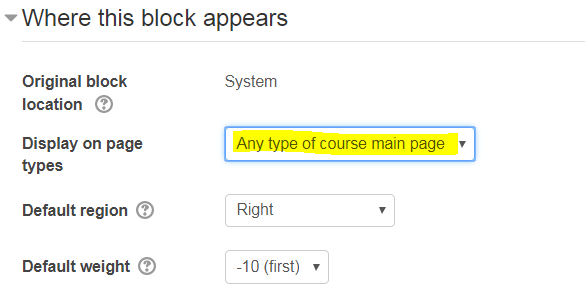
Configure the block



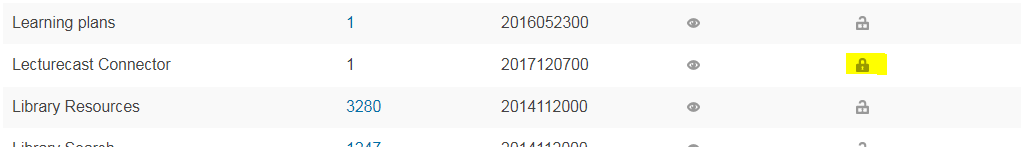
The important aspects are highlighted above. Set the block to display throughout the site and default region to the right. Please save the configurations.



In any course, with editing turned on, configure the block again.



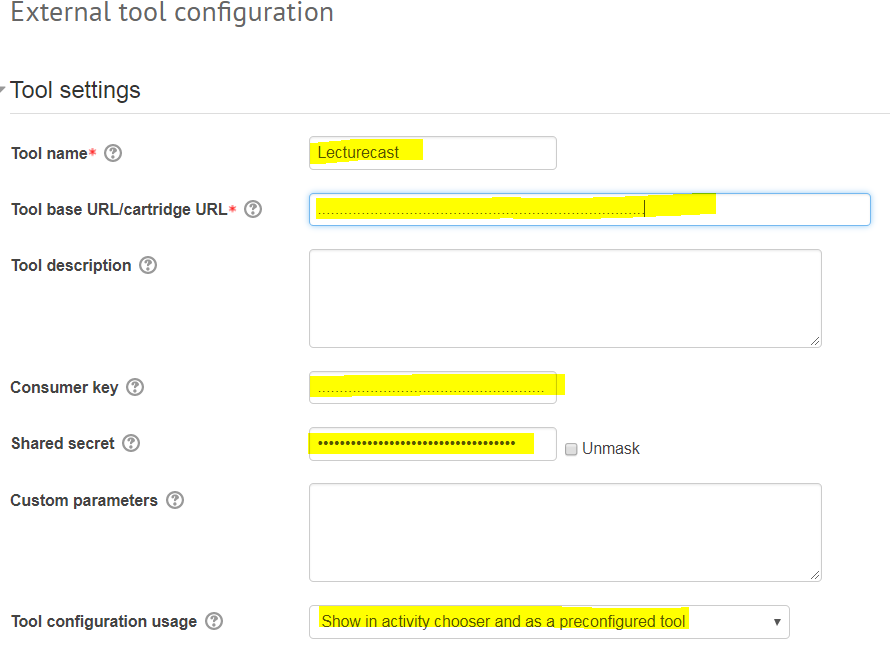
This time the only important aspect is to display in any type of course main page as we only want this block to be on course main pages.



If the block is deleted from any course, it will be deleted across the site so it is important to lock the block so that it cannot be deleted. So in Site administration >> Plugins >> Blocks >> Manage blocks, ensure that you click to lock the padlock for the Lecturecast Connector block as seen above.

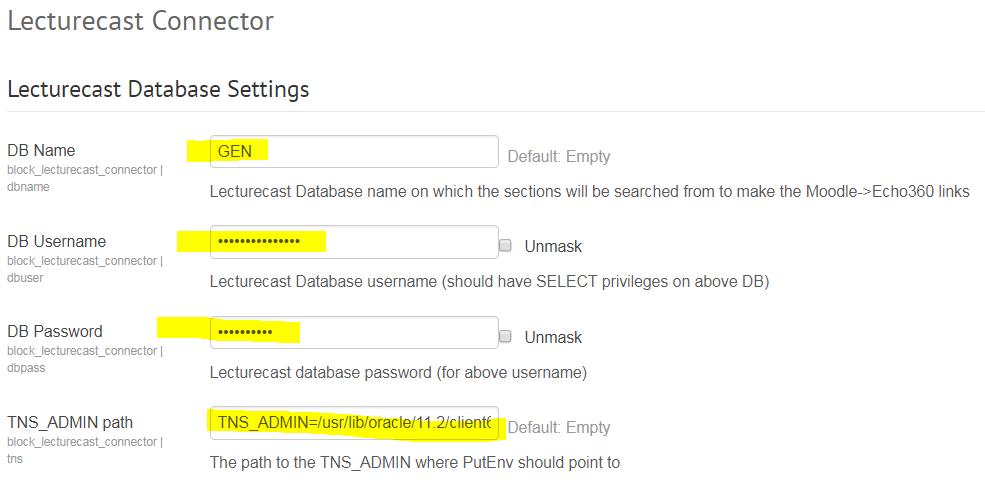
1. Setting up Lecturecast LTI Tool

For this part of the implementation you will need an admin for ALP to provide the following details: Consumer Key, Shared Secret, and Host URL



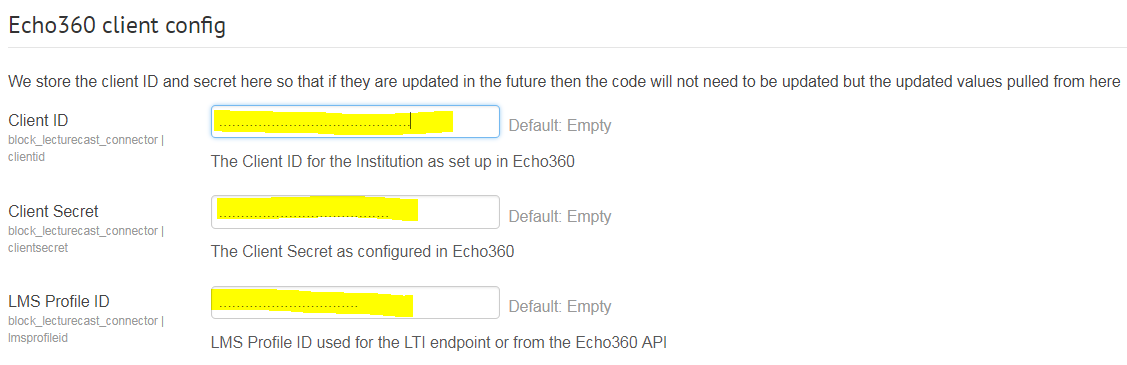
If your LTI tool is not yet configured you will need to create an external activity. From Site administration >> Plugins >> Activity modules >> External tool >> Manage tools >> configure a tool manually, give the tool a name like ‘Lecturecast’, enter the Host URL form ALP in the ‘Tool base URL/cartridge URL’, enter the consumer key and shared Secret. It is also a good idea to show as a preconfigured tool so that it will be listed in the activities in Moodle courses when you attempt to create the activity. You then need to save the settings.

1. Setup done in Moodle for connecting to GEN Database table



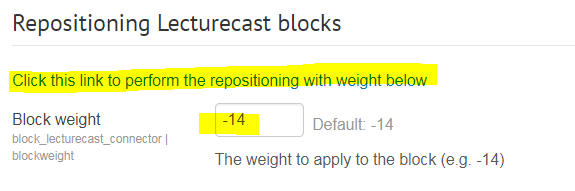
You will need your DBA to provide the details of the GEN oracle database and user that has access to the Scheduler tables. Access the Lecturecast connector settings page at Site administration >> Plugins >> Blocks >> Lecturecast Connector. Enter these as above and also the TNS\_ADMIN path. The TNS\_ADMIN environment variable is used to specify the directory location for the tnsnames.ora file in order for the OCI8 libraries to work (for PHP to talk to Oracle DB). Please save the settings

1. Setup done in Moodle for connecting to ALP API



You will again need these details from your ALP admin to be entered in the Lecturecast Connector settings page.

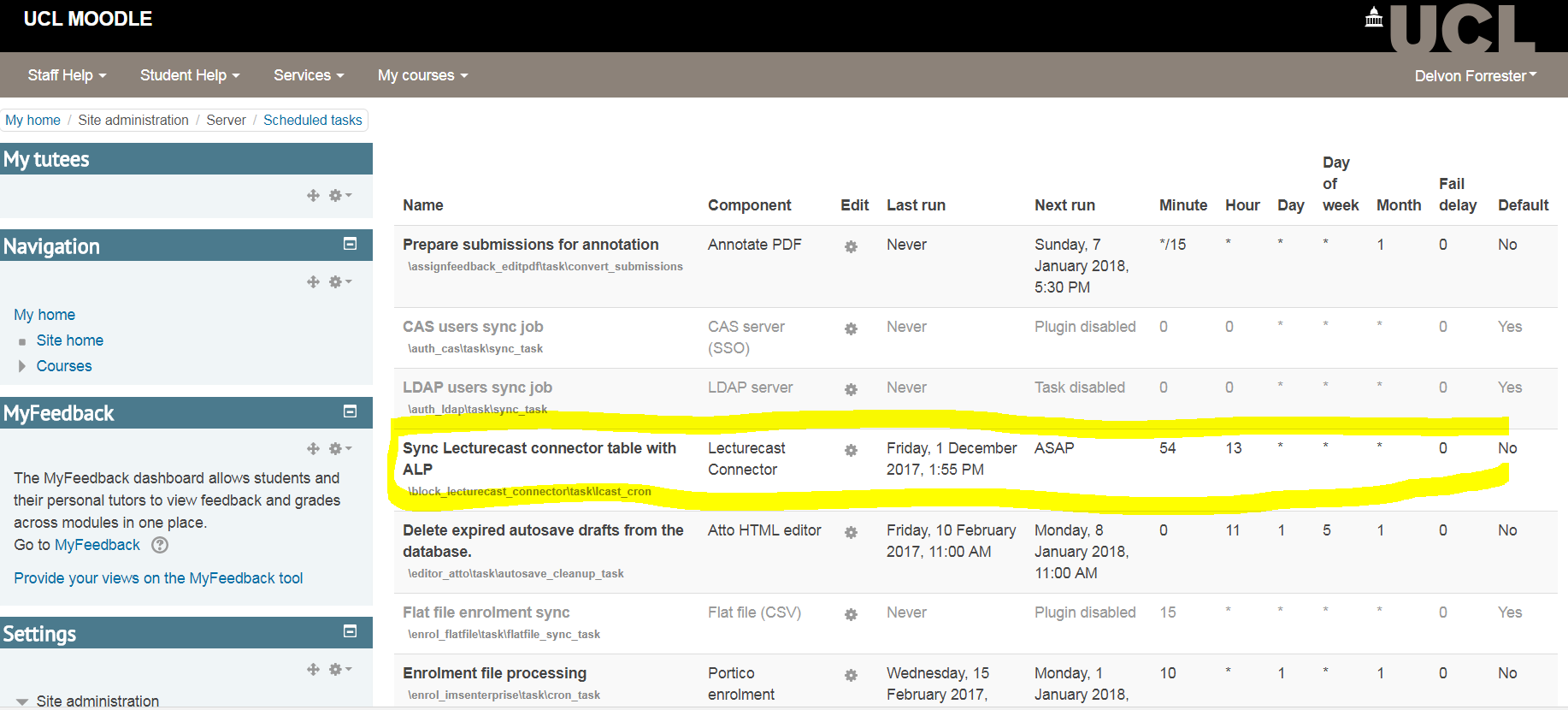
1. Setup done in Moodle to reposition the block



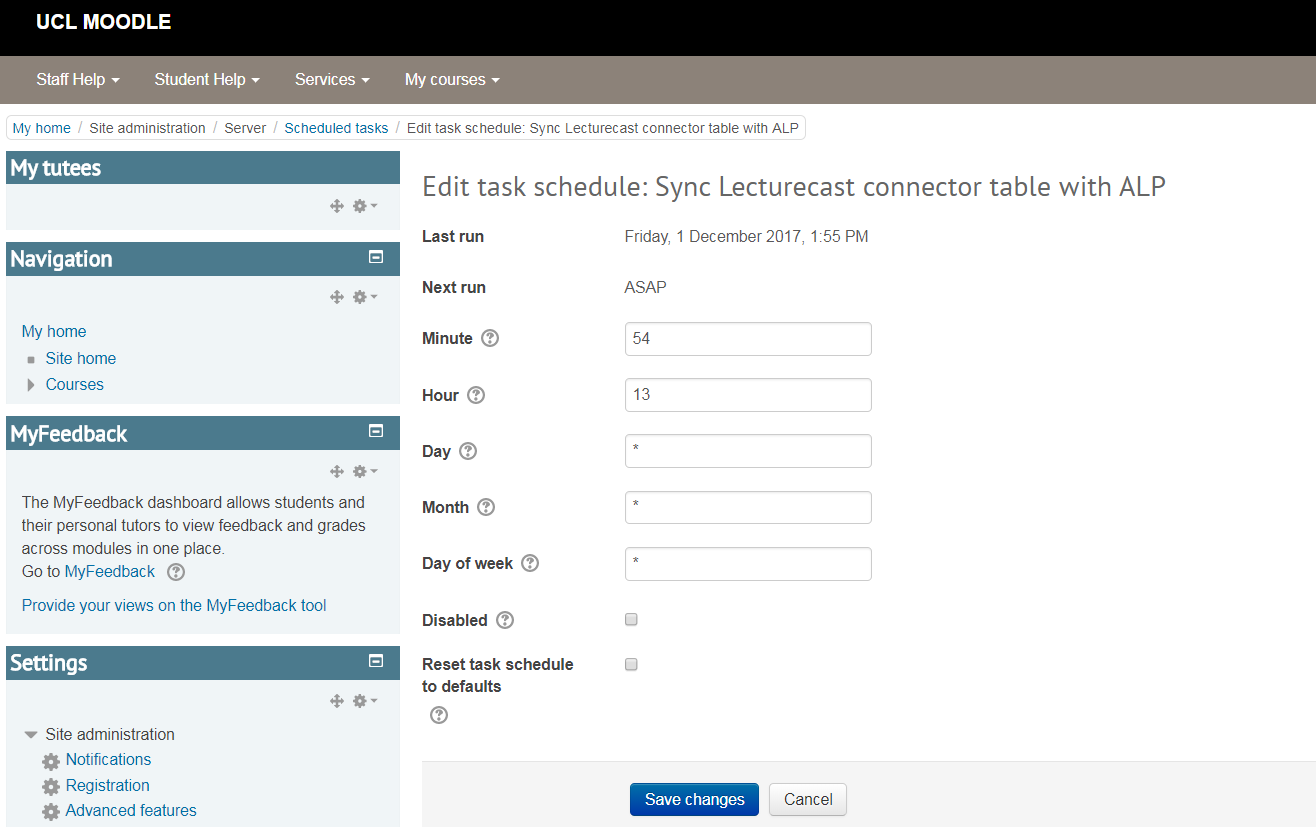
Within the section of the settings you enter where the block should be positioned. It is current set below the portico block which has a default weight of -14. The lower the integer, the higher up the block is shown. After setting the integer, click on the link as above and the block will be repositioned to the weight set on all courses.

1. Setup done in Moodle to scheduled batch job

The task is run when the schedule is set in MOODLE’s scheduled tasks (see below)



Depending on the time that is scheduled for the task to run (see below) then once MOODLE cron runs and the time has passed then the task will run and the Lecturecast connector table will be synced with ALP section mappings



1. Setup done in Moodle to perform unit testing

**Composer installation**

Composer is a dependency manager for PHP projects. It installs PHP libraries into /vendor/ subdirectory inside your moodle dirroot.

1. install Composer - <http://getcomposer.org/doc/00-intro.md>
2. install PHUnit and dependencies - go to your Moodle dirroot and execute php composer.phar install

**Configure your server**

You need to create a new dataroot directory and specify a separate database prefix for the test environment, see config-dist.php for more information.

* add $CFG->phpunit\_prefix = 'phpu\_'; to your config.php file
* and $CFG->phpunit\_dataroot = '/path/to/phpunitdataroot'; to your config.php file

**Initialise the test environment**

Before first execution and after every upgrade the PHPUnit test environment needs to be initialised, this command also builds the phpunit.xml configuration files.

* execute php admin/tool/phpunit/cli/init.php

**Execute tests**

* execute vendor/bin/phpunit from dirroot directory
* you can execute a single test case class using class name followed by path to test file vendor/bin/phpunit core\_phpunit\_basic\_testcase lib/tests/phpunit\_test.php
* it is also possible to create custom configuration files in xml format and use vendor/bin/phpunit -c mytestsuites.xml

**How to add more tests?**

1. create tests/ directory in your add-on
2. add test file, for example local/mytest/tests/my\_test.php file with local\_my\_testcase class that extends basic\_testcase or advanced\_testcase
3. add some test\_\*() methods
4. execute your new test case vendor/bin/phpunit local\_my\_testcase local/mytest/tests/my\_test.php
5. execute php admin/tool/phpunit/cli/init.php to get the plugin tests included in main phpunit.xml configuration file

**Windows support**

* use \ instead of / in paths in examples above

1. Setup done in Moodle to perform the behat/acceptance testing

The following steps are carried out to perform a behat testing on a local WAMP install of Moodle and the lecturecast connector block:

>> composer install - Only if it has not been installed before on that Moodle folder

>> Create a new 'dataroot' area for files especially for behat adjusting permissions accordingly.

>> If not there already, add Section 11 from config-dist.php to your config.php file and review the settings.

>> $CFG->behat\_wwwroot needs to point to your Moodle site yet be different from the 'normal' wwwroot (e.g. if you used localhost for wwwroot use 127.0.0.1 for the behat\_wwwroot). Whatever you choose, make sure it works.

>> $CFG->behat\_dataroot should point to the directory you created above

>> $CFG->behat\_prefix should be fine.>> Download the Selenium Standalone Server. It is a single JAR file, put it anywhere handy.

>> cd c:\wamp64\www\moodle\_314\

>> java -jar "C:\wamp64\selenium-server-standalone-2.42.2.jar"

>> http://127.0.0.1:8080/moodle\_314/

>> php admin\tool\behat\cli\init.php

>> vendor\bin\behat --config C:\wamp64\behatdata\_314\behat\behat.yml C:\wamp64\www\moodle\_314\blocks\lecturecast\_connector\tests\behat\lecturecast\_connector.feature

# Open Issues

There is currently no open issues. The project has adapted to all changes that were requested and all requirements have been met.