Bridging eLearning and Social Networks

by

Kris Noesgaard
BSc, University of Victoria, 2001

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science
in the Department of Computer Science

Supervisory Committee

Bridging eLearning and Social Networks

by

Kris Noesgaard
BSc, University of Victoria, 2001

Supervisory Committee

Prof. William Wadge, Supervisor (Department of Computer Science)

Prof. Alex Thomo, Departmental Member (Department of Computer Science)

Prof. Ulrike Stege, Departmental Member (Department of Computer Science)

Prof. Catherine Caws, Outside Member (Department of French)

Supervisory Committee

Prof. William Wadge, Supervisor (Department of Computer Science)

Prof. Alex Thomo, Departmental Member (Department of Computer Science)

Prof. Ulrike Stege, Departmental Member (Department of Computer Science)

Prof. Catherine Caws, Outside Member (Department of French)

Abstract

In this thesis, I present a framework for bridging the concepts of Social Networks and eLearning Learning Management Systems (LMSs) by finding elements and entities common to both systems, and eliminating these common elements to form an LMS Gadget which can hypothetically be used to extend any Social Network with an API that exposes User and User-Group information so that it also has the functionality of an LMS. I will also include a history of these systems, and a concrete example of such a unification. Finally, I will explore the application of the LMS Gadget to any Web Portal System that exposes User and Group management through an Application Programming Interface (API). This is an application that I conceived of, designed, and did the large majority of the programming for (with assistance on some data access routines to help meet an April 2008 conference deadline). Graphic design was provided by Amos Rowsell of Udutu Learning Systems (a private company for which I have held the role of Lead Developer since November 2005).

Contents

List of Figures

Acknowledgments

Thanks first of all to my grandfather, Borge Noesgaard, who supported me morally and financially throughout my academic career. Thanks to Mom & Dad for your support and encouragement – I love you both. And thanks to Tamara Yewchuk (my lovely wife) for putting up with me stressing about this thing. Thanks to Prof. Bill Wadge for his supervision and guidance of my post-graduate education, and Prof. Alex Thomo, Prof. Ulrike Stege, and Dr. C. Caws for participating in my Oral Defense committee. And, finally, thanks to Roger Mundell and the staff of Udutu for providing the inspiration and support.

Thesis – Bridging eLearning and Social Networks

1. Introduction

Electronic Learning (or eLearning) software has been around in one form or another since well before the popularization of the Internet. It saw early home use in Vic 20 computers, with programs that taught children math, and even earlier with computers that would run the famous Lemonade Stand program[1], which could be used to teach basic concepts of supply and demand. Universities through at least the later part of the 1980s used networked computers to assess and teach students mathematics, though machine-based instruction can actually be traced to as early as the 1950s, from work done by B.F Skinner[2]. When Windows 3.0 was released in 1990, it shipped with Toolbook. Toolbook was a piece of software aimed at bringing the authoring of multimedia documents to the masses. It was an authoring system for producing multimedia applications, simulations for software, and electronic courseware, with its most popular use being to create Computer Based Training (which is what they called eLearning at that time). eLearning also includes something called Computerized Adaptive Testing, which has been around

1 A faithful version of which can be found here: http://codenautics.com/lemonade/
2 http://www.coe.uh.edu/courses/cuin6373/idhistory/skinner.html

since at least the early 80s[3]. Computerized Adaptive Testing is an attempt to apply artificial intelligence algorithms to adaptively present to learners educational material by trying to establish what the gaps in their knowledge are, and adjust the presentation of the material accordingly with the goal of minimizing the time spent by the learner in achieving expertise of the subject matter by reducing review of material the learner may already know, and focussing and expanding on material the learner has less knowledge of.

More recently, with the birth of the web, Online Learning has become extremely popular. It takes the form of simply posting articles, to blogging, to using wikis such as Wikipedia to catalog knowledge by subject matter experts that then become indexed by search engines for learners to seek out. There is also software available that is geared to provide more structured learning, as well as a means for instructors or other interested parties to track learners through the learning material. These systems aim at learners being able to 'prove' their knowledge, either to themselves (knowing that they know the material), or to others (through certification)[4]. This software is called a Learning Management System (LMS), and it takes a dizzying array of forms – from WebCT, to Moodle, to custom and proprietary corporate systems. There is also a wide variety of software available that will produce courseware that will run on these LMSs, including Articulate, Raptivity, myUdutu, and Captivate. There is even a protocol for communicating information between electronic courses and LMSs, called SCORM (Shareable Content Object Repository Model), that has become the industry standard[5].

3 http://www.ericdigests.org/pre-9213/tests.htm
4 The difference between the two approaches to learning being akin to learning on your own through a library and the learning provided by a school or university
5 http://sorubank.ege.edu.tr/~e190411147/scorm/scorm4.pdf pp3

1.1 My Background

At the time of this writing, I have been a professional software developer for about 12 years with a focus on Microsoft-centric technologies and (more recently) ASP.NET. I began writing eLearning software over 5 years ago while working at Royal Roads University, where I was the Lead Developer in their Technology Transfer & Research Division, under director Roger Mundell. Our biggest project while I was employed there was a SCORM compliant online course authoring system called KoolTool. I later moved to the Centre for Teaching and Educational Technologies at Royal Roads, and further developed KoolTool as well as a smaller-scale quiz building web application. In November 2005 I left Royal Roads to form a company with my former director Roger Mundell called Udutu Learning Systems (of which I am also a partner). I am responsible for formulating the technical direction of Udutu, and have designed and am the Lead Developer for all of its technology. Our flagship product is called myUdutu. It is an online course authoring system that grew out of research that was done out of Royal Roads (KoolTool, in particular). It is currently in use by several thousand public and private organizations, our highest profile clients being the United Nations, Delek US Holdings, as well as a number of BC Government corporations. I was also responsible for the development of training kiosks currently in use by over 5,000 Delek employees. During my time as an eLearning software developer, I have become deeply familiar with a half dozen Learning Management Systems, and an expert in SCORM.

1.2 Problem and Motivation

One of the biggest obstacles to providing eLearning is cost. There is not just the cost of producing learning material, which can be considerable (10s of thousands of dollars for a single 30 minute course), but also in the purchase of an LMS (which can run to the hundreds of thousands of dollars), as well as the cost of hosting and maintaining the LMS. Because of this cost, it is often only large corporations or educational institutions that can even contemplate serving structured eLearning material in this way.

Costs have come down somewhat with the arrival of free template-driven Course Authoring systems, such as myUdutu[6], and the availability of the free open source LMS, Moodle[7]. However, barriers to entry still remain. A significant barrier to those who wish to host a Moodle LMS is the technical knowledge required to set it up, host it, and maintain it. For a high school social studies teacher, for example, who with a little perseverance could create a course on, say, Medieval history using one of the free course authoring tools, would still be faced with a technical hurdle in making this content available to his or her students that would likely lie well outside their area of expertise. Additionally, if they wanted to make this course available to other teachers and their students, how could they do it? Even if they were to set up a Moodle server for their own students, the other teachers would have to go through the same obstacles in setting up such a server for their students.

6 Http://www.myudutu.com
7 Http://www.moodle.org

In essence, there are currently two obstacles for common subject matter experts in making their knowledge available to learners. Delivery, in the forms of an LMS that can track learner progress (as well as the other services an LMS provides), and dissemination, in the form of making learning material reusable and accessible to the widest audience possible.

1.2 Thesis Outline

The topic of this thesis essay is on integrating social network software systems, such as Facebook, with eLearning protocols to create a social network based Learning Management System, as well as a framework for building Learning Management Systems into Facebook and social networks that make use of the Google OpenSocial API.

By doing so, we'll eliminate the technical obstacles of setting up and hosting an LMS for non-technical subject matter experts, since the solution will be a hosted solution that is immediately available to all users of these social networks. We'll also reduce the issue of dissemination, since by making course material available on these social networks, subject matter experts will instantly have the potential audience of all the users of those networks.

This thesis will be organized into chapters that:

- will give the history of Learning Management Systems, the industry standard protocol of communicating between courseware and LMSs (SCORM), as well as details of what those systems typically comprise of

- details of what Social Networks such as Facebook are, their history, and how their APIs work
- a proposal for a software system that will provide LMS functionality within these Social Networks
- details of how this software system works
- the challenges faced in the creation of this software system
- what lies ahead for systems of this nature

2 History – Learning Management Systems & SCORM

2.1 Learning Management Systems

Learning Management Systems, as they are known today, typically have the following features:

- The ability to manage users, instructors, and administrators, and assign roles and permissions to various aspects of the software to each
- The ability to create groups of learners (and possibly instructors) and assign roles and permissions to those groups
- The ability to deliver learning material
- The ability to deliver assessments
- The ability to score learners, and store this information
- The ability to generate reports on learner progress and mastery, with varying levels of granularity
- The ability for learners to communicate to the instructors about any given course and with each other

Most current LMSs are web-based, with the intention of being able to provide learning materials “any time, any place”. This, however, wasn't always the case (i.e. pre-internet LMSs). Before web-based LMSs, learners had to go to specially equipped training

centers to access the learning material. While this reduced the overhead of having instructors disseminate the learning material and manually test and grade the learners' mastery of them, there was still costs associated with housing the eLearning kiosks, maintaining each kiosk, and possibly arranging the travel of the learners to the learning centers. The biggest cost savings didn't come until it became possible to deliver this material over the Internet. Once this was possible, it entirely eliminated travel costs, and potentially eliminated kiosk purchase and maintenance costs[8], since the material could be accessed from learners' home computers.

Early generation LMSs typically shipped with some means of authoring learning material. These LMSs took a monolithic approach to authoring/serving content – content authored on one system couldn't be transferred to another. If systems were upgraded, care would have needed to be taken to ensure that the content was compatible and could still be delivered. If systems were switched, content would often need to be re-authored entirely so it would be compatible with the new system – an often costly and time-consuming affair that would discourage extensive creation of learning material. The types of content that could be delivered was also necessarily restricted by what the LMS offered. One system may offer the ability to create learning materials using multimedia; another may be restricted to just text. One system may offer a variety of assessment possibilities (i.e. ordering of steps, labeling diagrams, fill-in-the-blank, scenario simulations); another may have more limited possibilities (i.e. only multiple choice questions). While this may have served a somewhat tactical interest of these LMS

8 Some organizations still insist on learning materials only being accessible on specially approved machines - for example, if the learning material is classified, or if testing requirs an invigilator to be present.

providers by locking organizations into their solution, it certainly left adopters of these solutions with several still unsolved problems.

What was needed to solve many of these problems was a way for content to be transferred from one system to another - or, for content to be authored using one system, and presented in another. In order for such a thing to happen, a commonly agreed upon set of protocols needed to be created to first enable delivery of said content on a system. Second, a set of common protocols was needed for communication between the content as it was delivered to the learner and the system on which it was delivered (in order to track learner progress). At this time, some standards were born. Like most standards, at first there were many of them, it took a number of generations to get things right, and there were a few false steps along the way.

2.2 LMS/ Courseware Standards

The first commonly adopted set of standards was put forth by the Aviation Industry Computer-Based Training Committee (AICC)[9]. From their website, their initial mandate was to:

- Assist airplane operators in development of guidelines which promote the economic and effective implementation of computer-based training (CBT) media.
- Develop guidelines to enable interoperability.

These things they began to do, back in 1988. They did such a good job of defining their standard, that non-aviation industry eLearning solution providers adopted it, and it

9 Other "standards" put forward over the last 20 years include: IMS Global, OKI, IEEE/LTSC, LETSI, and ISO/SC36 – and there are even more.

became the *de facto* standard for nearly a decade – and is still in use today. It even has a critical advantage over the current most-widely accepted standard in that the answers to the assessments need not be stored on the client computer in any way. However, it lacks a fully developed specification for describing and loading content into an LMS, and was mostly developed before the Internet became commonly used to delivering content – though one of the aspects of its specification that dealt with web-deliverable content became adopted in SCORM.

The most widely accepted standard today, and one that was designed from the outset for web-deliverable courseware, is SCORM. It stands for Shareable Content Object Repository Model. It is the product of the Advanced Distributed Learning Initiative (ADL). ADL was created by the United Stated Federal Government in 1997[10] through the Department of Defence. In January, 1999 the Department of Defence was mandated to lead a collaborative standards development (through ADL) to standardize and modernize training and education management and delivery.

Their goal in creating this standard was to achieve the following "ilities"[11]:

Accessibility: The ability to locate and access instructional components from one remote location and deliver them to many other locations.

Adaptability: The ability to tailor instruction to individual and organizational needs.

Affordability: The ability to increase efficiency and productivity by reducing the time and costs involved in delivering instruction.

10 http://www.adlnet.gov/downloads/AuthNotReqd.aspx?FileName=IP2D0116T0900.ppt&ID=92, see page 6

11 SCORM 2004 Overview, pp SCORM-1-6

Durability: The ability to withstand technology evolution and changes without costly redesign, reconfiguration or recoding.

Interoperability: The ability to take instructional components developed in one location with one set of tools or platform and use them in another location with a different set of tools or platform.

Reusability: The flexibility to incorporate instructional components in multiple applications and contexts.

To this end, they created a standard that began to address these issues with SCORM 1.0 a year later. It would be a couple of quick generations later in October 2001, with SCORM 1.2, that ADL would have the robust specification and industry acceptance that would make SCORM a true "standard"[12]. The primary thing that led to the success of this standard is it included an XML specification for describing, cataloging, and loading courseware. This XML describing the courseware was defined in part with work IMS Global had done. For intercommunication between courseware and LMSs, they adopted some of AICC's Javascript protocols. The result was a specification that satisfied each of the "ilities" of their goal, and was so successful that within a few years most commercial LMSs and authoring tools supported SCORM 1.2.[13]

Then, a strange thing happened. Over the next couple of years, the designers of SCORM 1.2 realized they had some unresolved ambiguities and had left out some important features – namely the ability to describe sequencing of learning objects that would allow

12 http://xml.coverpages.org/SCORM-Version12-Ann.html

13 Blackboard/WebCT, Moodle, Saba, Learn.com, and most every other LMS supports SCORM 1.2. For Authoring tools, myUdutu, Articulate, RapidIntake, and most others do as well.

for conditional branching. To address these concerns, in January of 2004 they released SCORM 2004. However, in doing so they slightly changed the names of most variables and functions of the API protocols laid forth in SCORM 1.2[14], and renamed many elements in the XML specification. It resulted in making SCORM 1.2 courses incompatible with SCORM 2004 LMSs, and SCORM 2004 courses would likewise not work in SCORM 1.2 LMSs. By doing so, it can be argued that many of their prior "ilities" no longer held true. SCORM 2004 is a more complete and robust specification, but the lack of backwards compatibility resulted in it taking several more years before vendors were willing to adopt it[15]. ADL had effectively created two "standards", and expected everyone to either abandon the first one (and, one can only suppose, reconfigure, reauthor, or throw out all courses authored in 1.2), or support both. Several major LMSs to this day still do not support SCORM 2004[16], though all major course authoring systems do now support SCORM 2004 in addition to SCORM 1.2. With acceptance of SCORM 2004 having finally taken root, it seems that this will be the "standard" going forward – though to keep backwards compatibility with courses authored for the earlier standard LMSs now have the costly task of maintaining compatibility with two systems going forward. ADL seems to have learned from this mistake, fortunately, and all iterations of their standard since then (it is now at the 3rd edition) have retained backwards compatibility with the first SCORM 2004 iteration.

2.3 The SCORM 2004 Specification

14 For example, the function LMSInitialize() became named Initialize(). LMSSetValue became just SetValue, etc.
15 Once bitten, twice shy!
16 The Learning Manager and Saba, for example

The SCORM specification has two main parts. The XML document(s) describing the learning content (the description of a single piece of content may span over multiple XML files), and the protocols for communicating between the content and the LMS. The file structure of a SCORM course is a single file folder containing all resources necessary for loading and running a course. At the root of the folder must reside a file named 'imsmanifest.xml'. A simple example of this file is shown below[17]:

```
<?xml version="1.0" ?>
- <manifest identifier="UDUTU_Course11107" version="1.3" xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
  xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1.xsd
  http://www.adlnet.org/xsd/adlcp_v1p3 adlcp_v1p3.xsd">
  - <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>CAM 1.3</schemaversion>
    </metadata>
  - <organizations default="Course11107">
    - <organization identifier="Course11107">
        <title>Simple SCORM course</title>
      - <item identifier="Module49164" identifierref="ModuleAssets49164">
          <title>Launch Simple SCORM course</title>
        </item>
      </organization>
    </organizations>
  - <resources>
    - <resource identifier="ModuleAssets49164" type="webcontent" adlcp:scormType="sco"
        href="course/course11107.html">
        <file href="course/course11107.html" />
        <dependency identifierref="CourseAssets11107" />
      </resource>
    - <resource identifier="CourseAssets11107" type="webcontent" adlcp:scormType="asset" xml:base="course/">
        <file href="4multiplechoiceimage.swf" />
        <file href="animatedlist.swf" />
        <file href="blank.gif" />
        <file href="initializeTop.js" />
        <file href="interactiveUtils.js" />
        <file href="themes/default/logo.jpeg" />
        <file href="themes/default/interactives/whatswrong.xml" />
      </resource>
    </resources>
  </manifest>
```

Figure 2.1 - imsmanifest.xml example for SCORM 2004

Under "metadata" the "schema" identifies the document as a SCORM document. The "schemaversion" identifies it as a SCORM 2004 document.[18] The LMS is expected to

17 The SCORM 2004 XML specification with addendums is over 300 pages long, and is quite exhaustive. Few LMSs actually implement the entire range of the document, and focus on the most commonly used aspects of it. What is seen in this example is the most basic subset of items that an LMS is expected to implement to achieve basic compatibility. Not covered is the sequencing features that can describe more complex courses, nor the extensive metadata tags that enable full description of the objects for cataloging and indexing purposes.

18 CAM 1.2 indicates a SCORM 1.2 document

parse this document, and depending on what it finds in the "schemaversion" section, to determine what version of its SCORM loader to use, and what API to use to communicate with the learning material. The "organizations" element contains the structure of the course. If there are multiple chapters in the course, for example, you would see multiple "organization" elements. In this example, there is only one chapter. The "title" element is the title of the course as it will by default display in the course directory of the LMS. The "item"s of the chapter are the resource files that comprise the chapter, and they refer to elements in the "resources" section of the document. The "resources" section of the document can refer to either "SCO"s (which stands for Shareable Content Objects), or assets. SCOs are items that are expected to launch and track independently in the LMS, and therefore communicate via the SCORM communication protocols for scoring, progress, etc.. The html "launch" file (entry point) for the SCO is defined by the "href" attribute on the resource element. "Asset"s are simply documents or files that are either referred to by SCOs, or launch independently – but have no requirement for LMS tracking or communication. In this document, we have a set of assets in a resource ("CourseAssets11107") that are referred to by a SCO resource ("ModuleAssets49164") through a dependency. In this manner, you can have a single asset group referred to by multiple other resources – which can simplify and reduce the size of the document for large, complex courses. The document is supposed to fully define all elements in a SCORM course; all files in the course should appear in the document. Pathing to the files is relative, so the appropriate file and folder hierarchy can be faithfully recreated when the course is imported; to this end, all internal references to these files must also be relative. By structuring the document in this way, it becomes

possible for individual SCOs to be reused and reassembled into new courses through the LMS. Most LMSs don't actually take advantage of that feature of the specification, but it is made possible through it.

SCOs are expected to utilize client-side technology only – there are no allowances for server-side scripts (such as PHP, ASP, etc), since SCORM courses should be operating system/ web server setup agnostic (to facilitate re-usability of the content)[19]. To this end, they are almost always simple html files with Javascript and Flash embedded in the document to provide communication with the API, and to provide the dynamic, interactive aspects to the course. A drawback to this approach is that all answers to assessments are theoretically accessible to the learner, since they will be cached by their browser when they load the content. There are ways to limit this access through encryption or obfuscation, but a technically savvy learner will usually be able to reverse-engineer the answers; even if encryption is used, since the keys and algorithms used for encryption will all necessarily lie in the cache of the learner's browser. A way around this is to restrict delivery to "kiosk"s (computers for which the learners' access to system files and functionality is restricted) – though this limits the inherent appeal of web-deliverable technology. Another way around this restriction is to have files utilize Ajax to call out to external server software to provide answers or encryption keys. This also limits use to World Wide Web accessible computers (some learning systems may be corporate private network accessible only), and re-use and maintainability issues become

19 Also, Some LMSs may prevent server-side scripts from running in the SCORM course folder for security reasons.

a problem (the "answer" server on the other end of the Ajax calls must remain operational throughout the lifetime of the course, and not change addresses).

The SCORM API is the final, perhaps most important, component of the specification. It is through the API that learner progress and scores are tracked. As previously mentioned, it is only applicable to learning objects flagged as SCOs in the imsmanifest file. The API is a set of Javascript functions that the LMS must implement, and that the SCORM objects expect to be able to access. As such, implementation of any given LMS requires that the LMS reside on the same web domain as the learning objects; otherwise, the objects will not be able to communicate because of the cross-domain scripting restriction of Javascript.

An illustration of the communication architecture is below[20]:

20 From SCORM_RunTimeEnv.pdf, pp 18, SCORM 2004 documentation

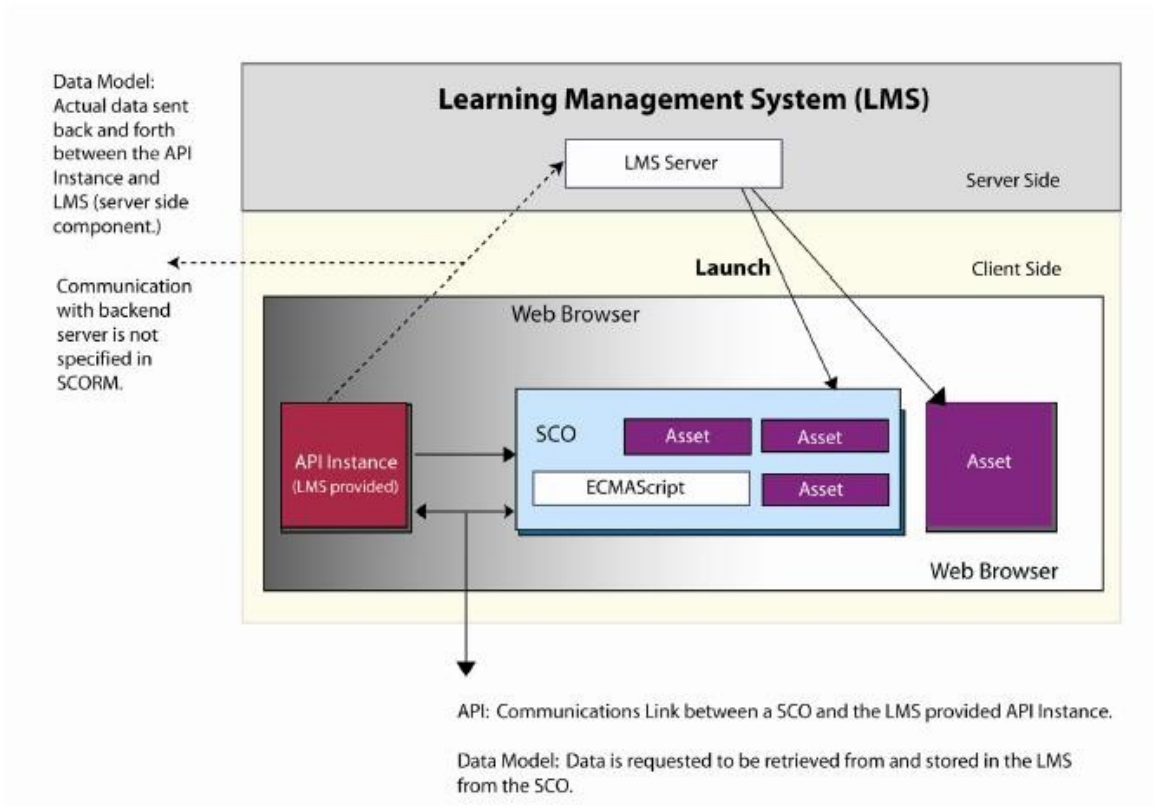


Figure 2.2 - SCORM conceptual run-time environment diagram

It is intended to work as follows:

1. The LMS launches the SCO either in a new browser window, or in a frame within the current browser
2. The SCO searches the browser window/opener hierarchy until it finds an object named: API_1484_11[21]
3. The SCO makes a call to the function “Initialize(")” on this object
4. The SCO communicates through the API making various calls until the learner is finished with the material (they close the window, click an 'exit' or 'save' button, etc)

21 This is a variable name given to ADL by IEEE for SCORM 2004 to prevent name conflicts. In SCORM 1.2, the variable was simply named “API” (since “API” could be a common name in Javascript libraries, which don't allow for namespaces)

5. The SCO completes with a call to the function “Terminate('')”

The complete list of Javascript calls the LMS is expected to provide for an implementation are as follows:

```
//the variable SCOHa in this example would be a hashtable that
//retrieves or sets the value of the 'parameter' key
//all methods that return 'true', should only return 'true' if
//the call was successful.  Unsuccessful calls may be retried, or an
//alert sent to the learner.

        function Initialize(parameter)
        {
          //the parameter is unused - reserved by SCORM for future use
          //called once when a SCO first loads in the learner's browser
          return 'true';
        }
        function Terminate(parameter)
        {
          //the parameter is unused - reserved by SCORM for future use
          //called once when the learner is finished with the SCO
          return 'true';
        }
        function GetValue(parameter)
        {
          //retrieve the value of the supplied parameter from the LMS
          return SCOHa.get(parameter);
        }
        function GetLastError()
```

```
{
  //return the code of the last error resulting from a failed
  //communication between the SCO and the LMS
  return LastErrorCode;
}
function GetErrorString(parameter)
{//return the text description of the supplied error code
 switch(parameter)
 {
  case 0:
   return 'No Error';
  default:
   return 'Unknown';
  }
}
function GetDiagnostic(parameter)
{//return the reason why the supplied error code was raised
  return '';
}
function SetValue(parameter, value)
{
    //set the value of the supplied parameter
    SCOHa.update(parameter,value);
    return 'true';
}
function Commit()
{
  //persist all accumulated data from SetValue calls to the
        //database
```

```
    return 'true';
  }
```

Some parameters will have meaning only to the SCO, and are used to save and restore learner state within the SCO. Others will be interpreted by the LMS. A few of the most typical parameters are:

cmi.location: typically used to store the last known location of the user in the SCO (for 'bookmarking'). The format of this value is determined and interpreted by the SCO.
cmi.suspend_data: typically used to store the complete state of the SCO, so it may be restored when the learner returns to it. The format of this value is also determined and interpreted by the SCO
cmi.score.raw: the learner's current score (there are also cmi.score.max and cmi.score.min parameters, to put the score in context)
cmi.progress_measure: the learner's current progress through the SCO, expressed as a number between 0 and 1.

There are a number of other parameters, some mandatory for basic SCORM compliance, others optional. Most authoring systems make use of only the mandatory parameters, and LMSs typically will be able to generate reports on the values (like score, progress measure, etc) that aren't simply state-storehouses for the SCO.

The LMS typically uses Ajax, HTTP POST through a hidden iframe, or Java applets to communicate data sent through the API back to the server for processing and storage.

The LMS will either host the API in an 'opening' window (the LMS will open a new browser window for the content, and the communication will be across windows), or the API will be in a 'parent' frame of the same window. There are advantages and disadvantages of each of these approaches. If the API is in a different browser window than the course and the course window closes unexpectedly (i.e. if it crashes or the user clicks on the top right 'x'), the API can still propagate the results onto the LMS back-end. This is generally not possible if the content is in the same window as the API, since if the window housing both closes then the API frame often won't be able to get its 'save' commands off to the server before it ceases to exist. A disadvantage with having the content and LMS API in different windows is if the user navigates the LMS API browser window away from that page, and the course content window is still open, then they may not realize that work they continue to do in the course content window will not be saved (since the LMS API is no longer there).

3 History – Social Networks and Facebook, and the Facebook API

3.1 Social Networks

Early Computerized Social Networks were popularized through the form of Bulletin Board Systems (BBSs). A BBS is a piece of software running on a computer that acts as a hub for other computers. These other computers access the BBS over a network. Initially, they accessed the BBS through phone lines with modems. There was typically a means for a user to create a personal account that would store information pertaining to the user, and a way for users to login to these accounts. BBSs served as a means for enthusiasts to communicate to one another, and to ultimately share files or play games with one another. The first public BBS was created on February 16, 1978 in Chicago, Illinois by Ward Christensen.[22] Some incorporated sets of ad-hoc standards that allowed modules (called "Doors")[23] to run (offering various multi-user games and other features) on any BBSs that incorporated the standard.

When the Internet became popular, BBSs became accessible over the Internet, but soon lost popularity for those primarily interested in social networking when chat rooms, dating sites, and "meet up with old classmates" (i.e. www.classmates.com) type sites began to proliferate.

22 http://www.bbsdocumentary.com/software/AAA/AAA/CBBS/memories.txt
23 FOSSIL was a popular standard: http://www.ftsc.org/docs/fsc-0015.001

On February 4th, 2004[24], Mark Zuckerberg, a student of Harvard University, released "The Facebook" (he would soon drop the "The") to other students of his school. At first, it was only available to Harvard students, but soon became available to other Ivy League schools, and eventually to the world at large. At the time of this writing, it has more than 110 million users, and growing[25]. Facebook wasn't the first social network service, but it is certainly the largest. In fact, you'd be hard pressed to find anyone between the ages of 15 and 25 in the United States and Canada that does not have a Facebook account. While, however, Facebook is largely a young person phenomenon[26], older users are joining in increasing numbers (often to the consternation of the younger user base)[27] – often to keep long-distance relationships with children and grandchildren.

Social network services are characterized by offering a few common components. First, they have a way of creating an account that uniquely identifies users. They also all offer a way for users to communicate to one another, be it through chat or forums. They also offer a way to build a list of other users that they can easily communicate with directly. Modern social network services offer a way to invite people that aren't currently members of that service into that service. Finally, they usually allow a way to create groups that other users may search for and join.

In August of 2006[28], Facebook created an Application Programming Interface (API) that allowed third-party developers to create applications that would integrate with their

24 http://www.facebook.com/facebook
25 http://www.facebook.com/press/info.php?statistics
26 http://www.facebook.com/press/info.php?statistics
27 http://www.crunchgear.com/2007/10/14/will-facebooks-older-users-drive-away-all-the-young-kids/
28 http://developers.facebook.com/news.php?blog=1&year=2006&month=8

system. This allowed other developers to effectively piggy-back on the vast user base that Facebook had, as well as use their authentication system for uniquely identifying users as well as the groups they belonged to. Other social network services quickly followed suit. Bebo began to use the API protocols they developed so that applications that worked in Facebook could easily be made to work with Bebo. Google has also developed something called the OpenSocial API (released November 2007)[29] that has similar functionality for any social network services (such as LinkedIn, MySpace, and Orkut) that implement it. Basically, any applications that implement these interfaces will work on the social networks.[30]

3.2 The Facebook API

Facebook applications are accessed through Facebook once the user logs in, and reside in the following space (highlighted in red):

29 http://battellemedia.com/archives/004058.php
30 For example, it is currently in use by the Canadian Home Builders Association

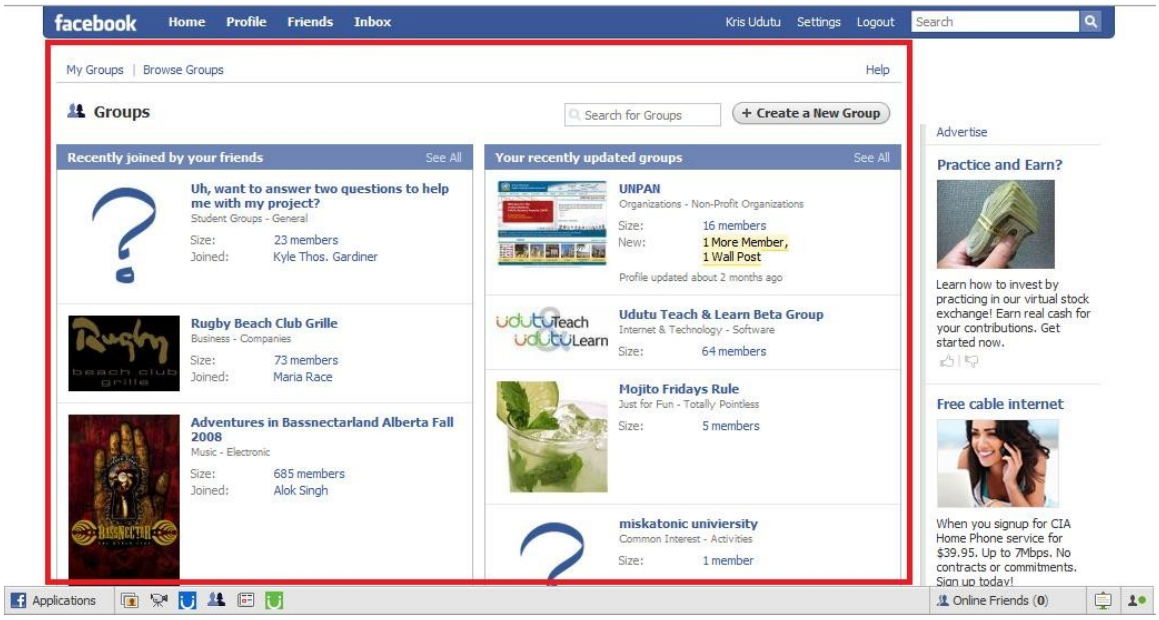


Figure 3.1 - Facebook Application Canvas

The menu of applications available to any user is through the "Applications" button on the bottom left of the screenshot. New applications can be added from an option from this button. Developers can create new applications for Facebook, and list them in the Applications directory by adding the "Developer" application and using it to define their own custom application.

Facebook applications can be hosted on Facebook's servers for free (which requires them to be written in Facebook's custom markup language known as FBML and/or in PHP). However, by doing so, application authors do not have access to the file system or the operating system of the hosting machine. The other alternative is to host the application on a machine controlled by the application author, and to simply configure a link to the off-site application. The second option is preferable if direct access to the file system or other OS commands are needed – the drawback is the corresponding hosting and bandwidth costs. The application will reside in an HTML iframe in this case, but will

still be able to communicate to Facebook through their REST interface (a kind of simplified web-service that involves HTTP POSTS and GETS to and from Facebook servers to request and receive data), and will otherwise operate as any other Facebook application would.

The Facebook API consists of two main parts[31], both of which are accessible through REST. The first part[32] is a series of commands that establish a session state with Facebook, and provides a very common set of data requests concerning information related to the user. An example of these are as follows:

users.getInfo: Returns a wide array of user-specific information for each user identifier passed, limited by the view of the current user.

profile.setFBML: Sets the FBML for a user's profile for the application, including the content for both the profile box and the profile actions.

notifications.send: Sends a notification to a set of users.

groups.get: Returns all visible groups according to the filters specified

groups.getMembers: Returns membership list data associated with a group

friends.get: Returns the identifiers for the current user's Facebook friends.

fql.query: Evaluates an FQL (Facebook Query Language) query.

The other main part of the Facebook API is FQL (Facebook Query Language). FQL is similar to SQL in syntax (though has only a subset of available commands), and allows

31 Facebook also provides an API that enables authors to provide a look and feel consistent with Facebook as well as some common interface features, but is not accessible to solutions not hosted on Facebook's servers

32 http://wiki.developers.facebook.com/index.php/API

application authors to query for data directly. The advantage of using FQL over the fairly rich array of API commands is that the developer can request the exact data they want, and only the data they want, reducing the need for repeated calls, resulting in a potentially significant savings in response time and bandwidth.

4 Hypothesis – Unification

The core thrust of this thesis is that it is possible to unify these two technologies, eLearning with Social Networks (and even web portals in general), with the aim of achieving the goals laid out in section 1.1. Facebook, and other Social Networks, provide not only the audience for eLearning, but a mechanism for simplifying its access and delivery. There is a commonality between traditional LMSs, and what Social Networks already offer.

This commonality is:

- account creation
- user authentication
- group creation and management
- facilities for users to identify and communicate with one another
- facilities for users to form communities

A simplified entity-relationship diagram to bridge the two systems is therefore suggested to be:

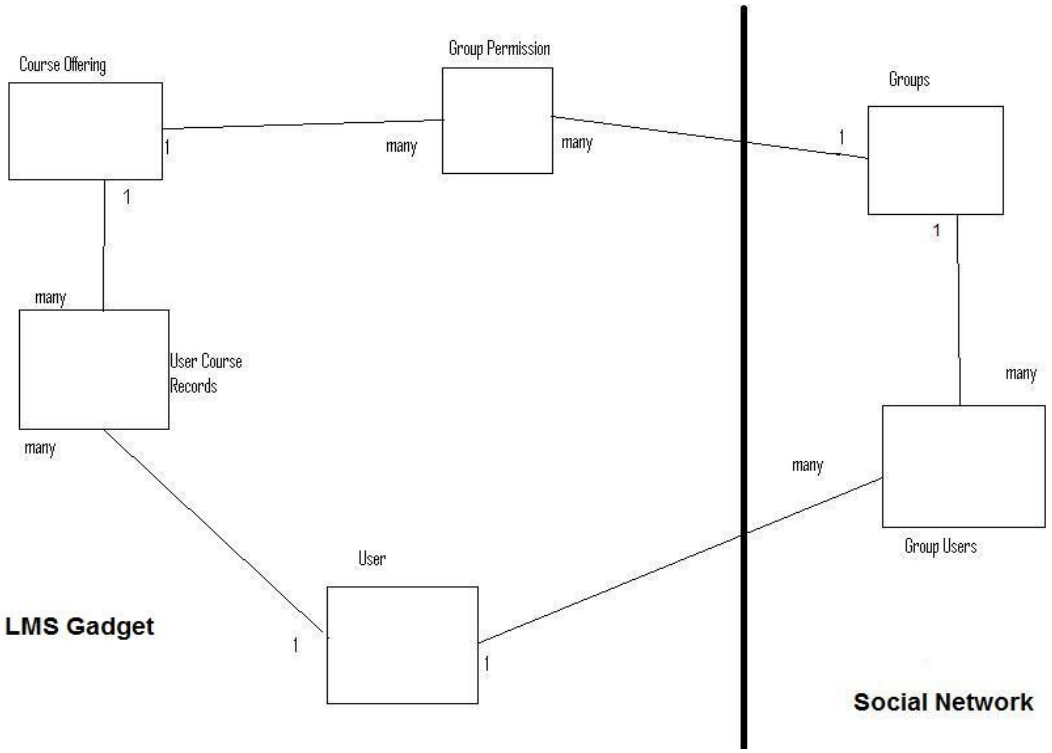


Figure 4.1 - LMS Gadget/ Social Network Entity-Relationship Diagram

The User table would be a list of UserIDs as they existed on the Social Network, along with basic information about each user that would be mirrored in the LMS gadget database to reduce the need for repeated calls to the source database. (i.e. First Name, Last Name)

Note that this LMS gadget could therefore be applied to any system that provides an interface that supplies user and group information through a web-accessible interface (i.e. SOAP Web Services or REST).

5 How to Unify/ Considerations for Unification

There are “real-estate” considerations not only when designing any web-deliverable application, but also when designing an application that is meant to reside on a Social Network. Social Networks (or other 'portal' applications) have areas on the left or right of the application's 'canvas' that are set aside for navigation and/or advertising. Facebook allows a maximum width of 646 pixels for any application that is to reside within it. Any application should be less than that, or horizontal scroll bars will appear, which is generally considered a nuisance for users. The vertical size of the application is more flexible, since you can either tap into the vertical autosize mechanism that Facebook provides, or accept vertical scrollbars (which is more acceptable for users – web sites often have a vertical scrolling component).

Another consideration is to reduce calls to the Social Network for data, since there is an additional round-trip overhead beyond delivering interface information to the user's web browser to get the information necessary to deliver the interface and accomponying data. Primarily, if it is necessary to deliver, for example, detailed information for a group of users, it is better to combine the call such that all information that will be bound to a

datagrid is made in a single call, rather than a call to get user ids, and subsequent calls to get the information for each user based on those ids. This is less of a concern if the data for an application is all in the same database, since response times are so much quicker over a LAN or simply within the server than over the internet.

Direct access to the filesystem is necessary for this application, since the SCORM courses need to be unzipped (and which for most web development frameworks require a library which needs to be specifically installed on the hosting server), and their directory structure must remain intact (for pathing reasons – all components need to know where the files they rely on are) and on the same domain as the LMS that they will be communicating with (to avoid cross-domain scripting issues). As a result of this requirement, using Facebook's servers to host the application is simply not an option since they do not offer all of these abilities[33].

Generally, learners and course authors/instructors are not the same people. Most learners will have no interest in creating and offering their own courses. As such, it is useful to break the functionality of each aspect into two separate components. This reduces the interface options of the application (too many options can confuse users if they have no intention of using the options anyway). Furthermore, it reduces computation, storage, and bandwidth costs of learners who are fumbling through the author/instructor functions with no real intention of offering courses themselves.

33 No direct filesystem access being the deal-breaker, here.

Once courses are loaded into the system, a mechanism must be provided to assign learners access to courses. The most popular Social Networks (i.e. Facebook, Bebo, LinkedIn, and most proprietary portal systems), provide a means to create groups and assign individuals to those groups. Utilizing this already provided grouping mechanism not only reduces the need to rewrite what is already there, but from a usability perspective utilizes what the user already knows how to do to create and manage cohorts. The system need only be able to poll group membership and create course offering/group associations to facilitate management of course access.

Obviously, from both the "Teach" and "Learn" aspects of the application, the course offerings that teachers and learners have access to should be listed, with commands pertinent to the course and permission level made available. (i.e. take course, view results, delete course, etc)

Since groups in a Social Network environment may number in the hundreds of thousands (or more) of users (i.e. the group of Canadians), it may be necessary to restrict who may offer courses to any given group. If this restriction is not in place, then learners may find their course list flooded with 'spam' courses if they belong to large groups. One way to do this is to limit the power of making courses available to groups to officers or administrators of that group.

Finally, the solution should abstract calls to the identity/membership provider (i.e. the Social Network or portal) into a separate layer, to ease the integration of additional providers with the system.

6 Description of Implementation

This system was implemented with an array of technologies. The back-end was programmed in C#, using the ASP.NET 2.0 framework by Microsoft for interface and web-deliverability functionality, the ADO.NET framework for database access, and numerous other .NET libraries for XML parsing, Regular Expressions, string manipulation, file system access, and other functions. The database is Microsoft SQL Server 2005, using T-SQL for all stored procedures. The front-end is all Javascript, HTML, and CSS. The SCORM communication layer (as well as some UI components) utilizes AJAX. 3rd party libraries to ease some interface creation and AJAX calls are provided by Telerik[34]. SharpZipLib[35] (a public domain library) provides the unzipping of SCORM packages. The Facebook Developer Toolkit for ASP.NET by CodePlex was used to ease communication with Facebook's back-end. The SCORM version that was implemented was SCORM 2004. This implementation also integrates directly with myUdutu (a separate course authoring application designed and primarily developed by me over the past couple of years). The system requires Windows Server 2003 or newer.

34 http://www.telerik.com/products/aspnet-ajax/overview.aspx
35 http://www.icsharpcode.net/OpenSource/SharpZipLib/

The system is currently implemented only for Facebook and the Canadian Home Builders' Association's user portal (in beta). The library pertaining to user information, authentication, and group membership is abstracted so as to be easily adapted to other Social Networks or portals.

The implementation has two parts: UdutuTeach and UdutuLearn. It was developed for Udutu Learning Systems, Inc beginning in January, 2008.

UdutuTeach is the administrator/instructor aspect of the system. Through UdutuTeach, users can load courses into the system by either providing SCORM 2004 compatible packages, or directly through myUdutu. Once a course is loaded, users can assign permissions to the course for Facebook Groups (with further granularity provided to members of those Groups) or Friends (a kind of Group that all Facebook members have for themselves and are Administrators of). Users may also generate reports on learner progress through their courses. There is also a mechanism by which they can communicate to their learners about any given course offering (a kind of simplified forum). There are three levels of permission that can be granted on any given course: Administrator, Officer, and Learner. Administrators may edit a course offering, preview it, assign permissions to it, delete it, engage in its forum, or take the course as a learner. Officers may do anything an Administrator can except edit or delete the offering. Learners can only take the course or engage in its forum. The 'Instructor' class is anyone who is either an Officer or Administrator of a course offering. Users may also use this

interface to add UdutuLearn to their Social Network account (if they haven't already). Finally, users may invite 'Friends' to add UdutuTeach to their accounts (if they know anyone that would have use of this application).

UdutuLearn is the learner aspect of the system. Through UdutuLearn, users can access courses for which they have been given permission through UdutuTeach. They can also search and request access for courses that have a public profile, and engage in public forums for courses for which they have access.

To add either of these applications to their Facebook account, users may:

1. Log into Facebook
2. Click on Applications (at the bottom left corner of their browser)
3. Choose 'Find More'
4. Enter 'udutu' in the search box
5. Click on the desired application to add, and follow the prompts
6. The application will appear in the users' list of Applications from this point forward. Clicking on UdutuTeach or UdutuLearn will launch the application.

The UdutuTeach aspect has the following primary interfaces:

- The main (default) screen
  - 'Add Course'
  - 'Add UdutuLearn'

- 'Invite Friends'
- For individual courses, access to 'Preview' the course
- 'Edit' course attributes
- Assign 'Permissions' to courses
- Engage 'Discussion'
- generate 'Reports' on courses
- 'Delete' courses

Figure 6.1 – UdutuTeach main page

Add Course:

Here users can either provide a SCORM 2004 package (.zip file), or go to their myUdutu account and add a course directly through there. If the user clicks on 'launch myUdutu', they will be taken to the login screen of myUdutu. If they choose to login, their Facebook identity will be associated to their myUdutu login, so if they go to

'distribute' a course through myUdutu, it will be published to their Facebook UdutuTeach account, and automatically appear there. If they wish to upload a SCORM 2004 package from their computer, they can click 'Browse', and navigate to it.

Figure 6.2 - add course

Add UdutuLearn:

Clicking on this link will begin the process of adding UdutuLearn (if the user doesn't already have it – otherwise it will just take them to the UdutuLearn interface)

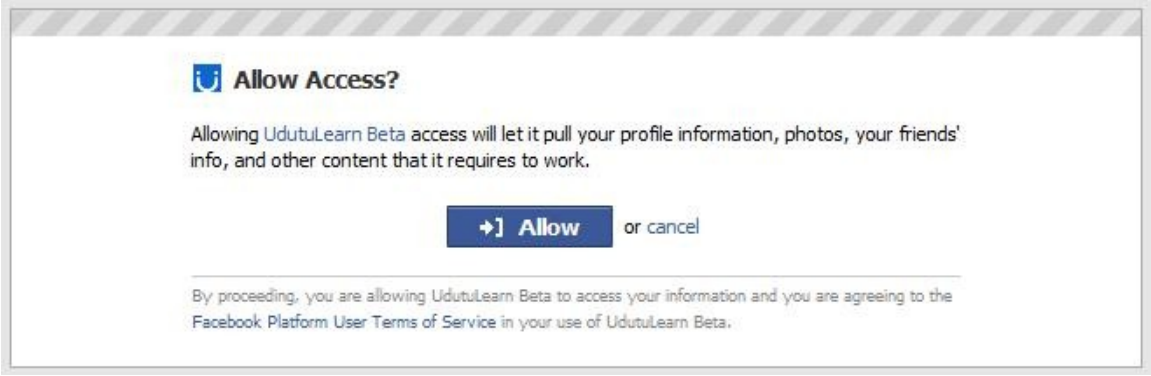


Figure 6.3 - add UdutuLearn

Invite Friends:

Through this interface, the user may send notifications to their 'Friends' that contain links that will add the UdutuLearn and/or the UdutuTeach applications.

Figure 6.4 - invite friends

Preview:

This launches a course that has been loaded into the system in a new window. Progress is not tracked in 'Preview' mode.

Figure 6.5 - preview course

Edit:

When the user Edits a course, they may change the name and description (the defaults of which are automatically added from the SCORM package when first loaded). They may also change:

Maximum Attempts: learners may reset the state of the course this many times

Course Active: whether or not learners may access this course

Access Code/Public Availability: if the course is public, it will appear if learners search for it. From the search interface, a learner may request access to the course (which must be granted by an administrator), or enter an access code to be automatically granted access. The access code they must enter is defined here.

Passing %: the percentage of success the learner must achieve on scored assessments to be regarded as having 'passed' the course.

Width/Height: the dimensions to adjust the window of the course once it loads. If left blank, no adjustments to the window will be made[36].

New content may overwrite existing courses through this interface. This can create a problem if assessment questions have changed (if there is prior learner activity for this offering, data will not synch).

36 Many SCORM courses have Javascript that will adjust the window size to the appropriate dimensions

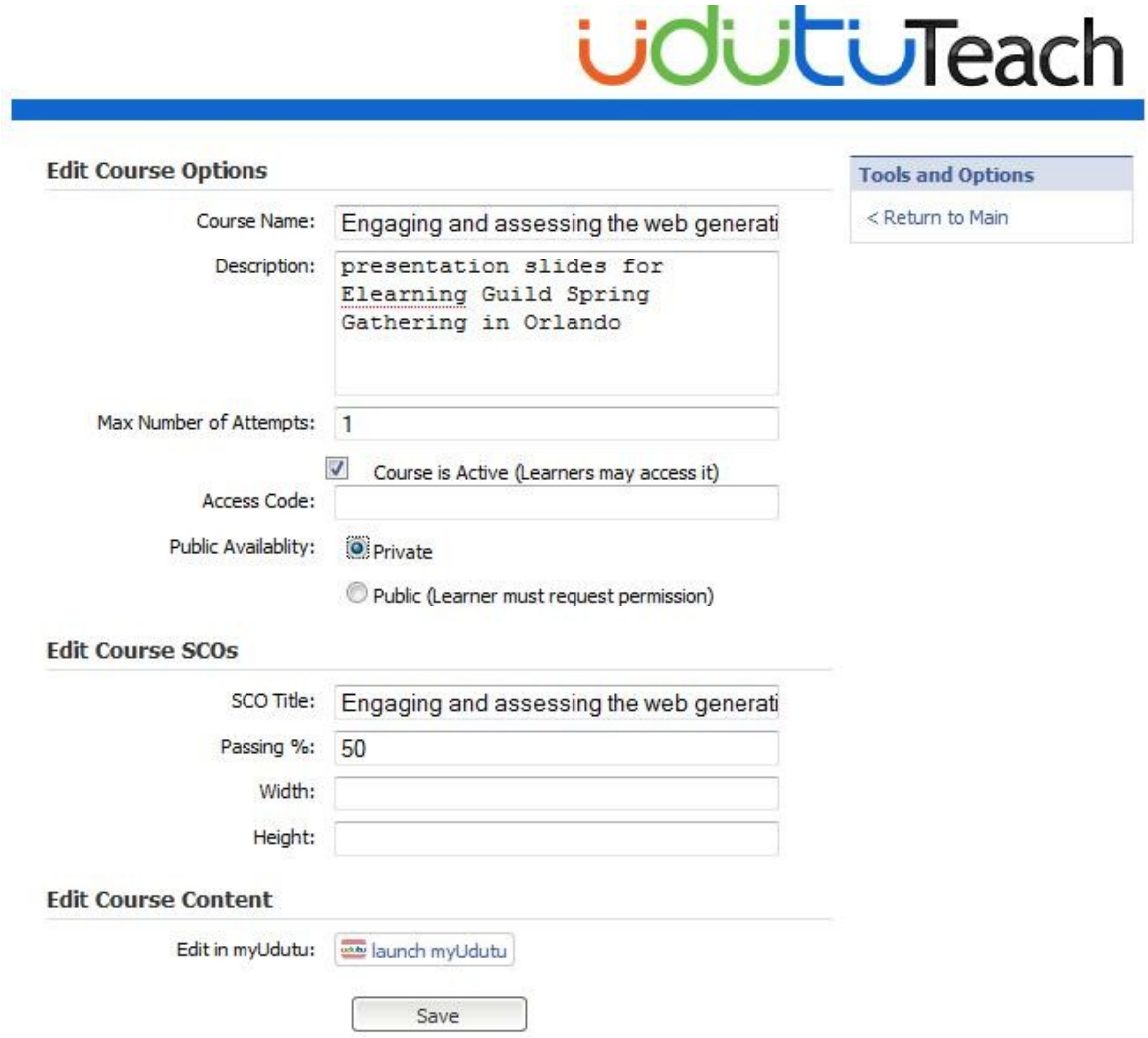


Figure 6.6 - edit course

Permissions:

Here, permissions to groups, group members, and Friends are granted. Clicking on the 'View Access Requests' link will take the user to an interface where they can grant access to users who found the course and requested access to it through the 'search' interface. 'Permission Exceptions' show an editable list of all individual users that have been granted specific permissions to the course (apart from inheriting access through Group membership).

6.7 course offering permissions

Discussion:

The Discussion area of a course provides a mechanism for Learners and Instructors to communicate with one another. The interface shows all comments sorted in descending order by date. Comments that are directed to Instructors (again, those with Administrator or Officer access to the course) are shown in blue, and are only visible to Instructors. Only Instructors can delete comments.

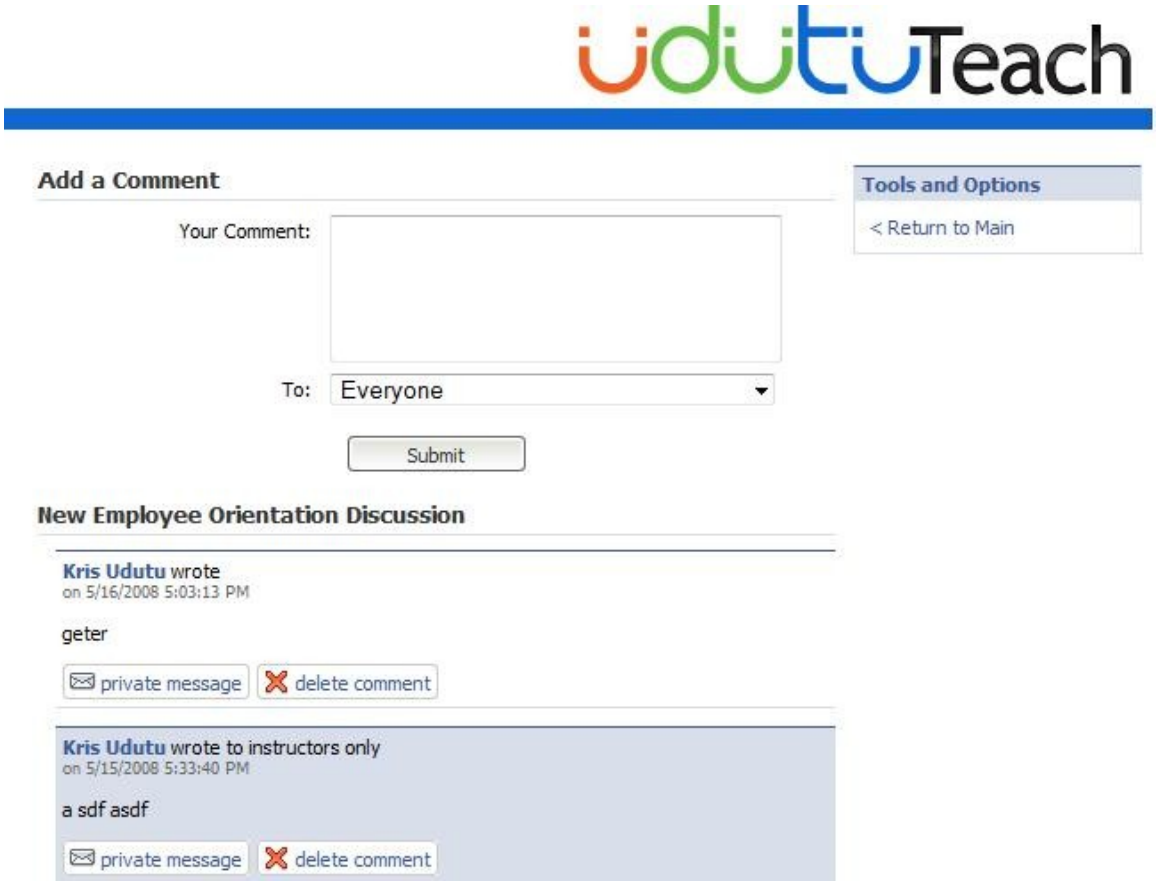


Figure 6.8 - discussion

Reports:

Navigating to this screen will show the course activity for all learners who have accessed the course in a new browser window. By deselecting 'Include Inactive Users', only those learners who currently have access to the course will be shown. Clicking 'Check Active Users' will refresh what UdutuTeach knows about who does and doesn't have access to the course[37]. There is the ability to generate reports in a variety of formats that the user can save to their computer from this interface.

37 Discovering this data is a time-consuming process because of repeated calls to the Facebook database, so live data is not shown.

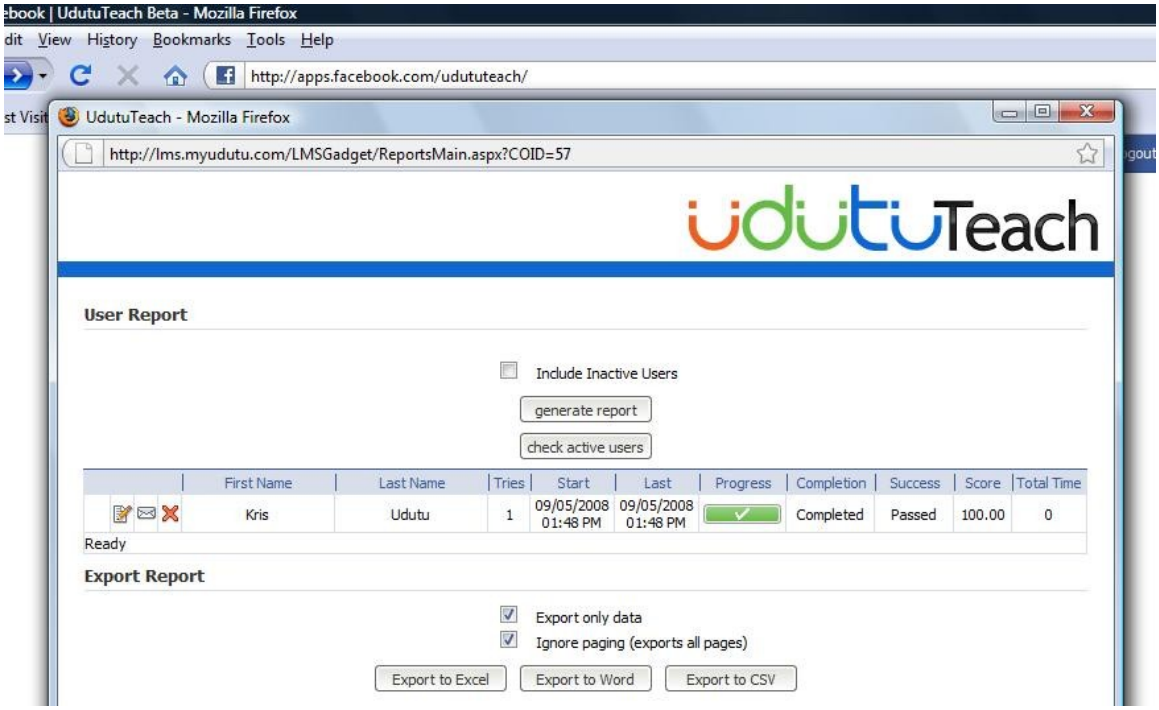


Figure 6.9 - reports

Delete:

Simply deletes the offering, and all associated data.

The UdutuLearn aspect has the following primary interfaces:

- the main (default) screen
  - 'course search'
  - 'course page'
  - 'discussion'
  - 'message instructor'

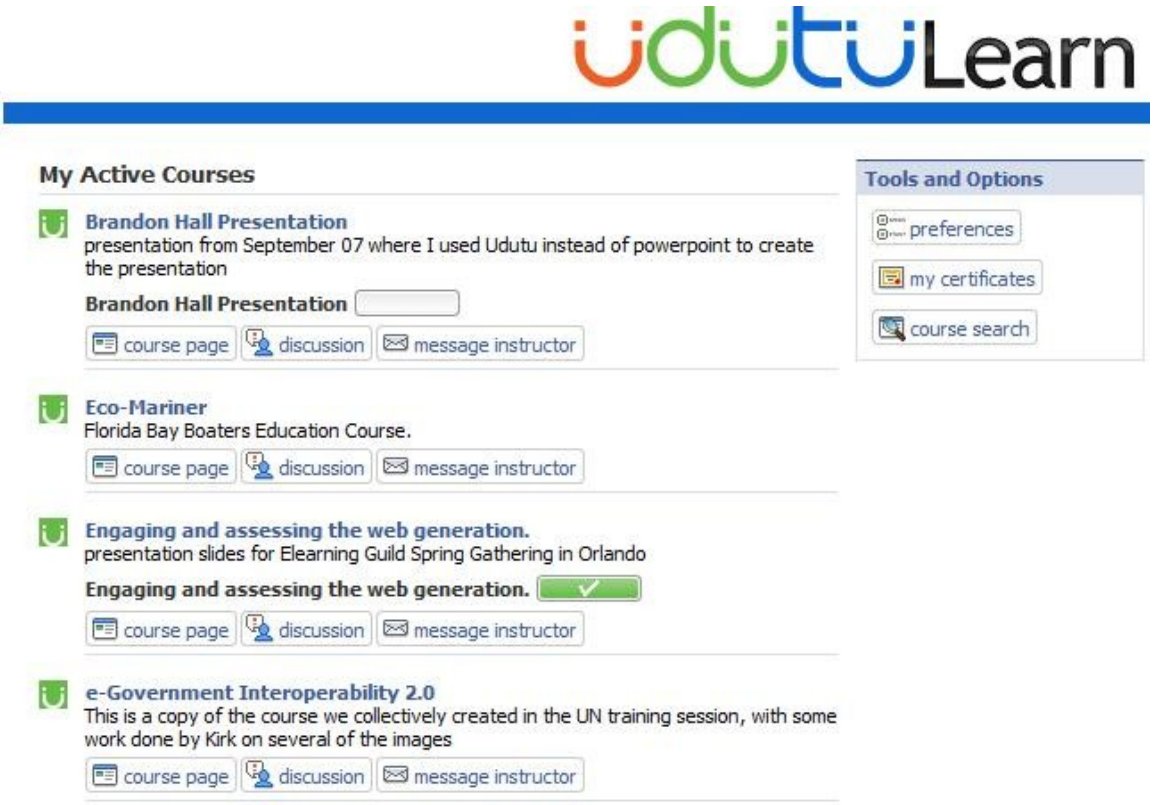


Figure 6.10 - main page

Course Search:

Through this screen, learners can search for courses that have been flagged as 'Public' by their Administrators. The search box returns results that match on course offering title or description. Clicking on 'request access' will either send a notification to the Instructors of the course requesting access, or prompt the learner to enter an access code which an Instructor has communicated to them through other means (the correct entry of which will automatically grant access).

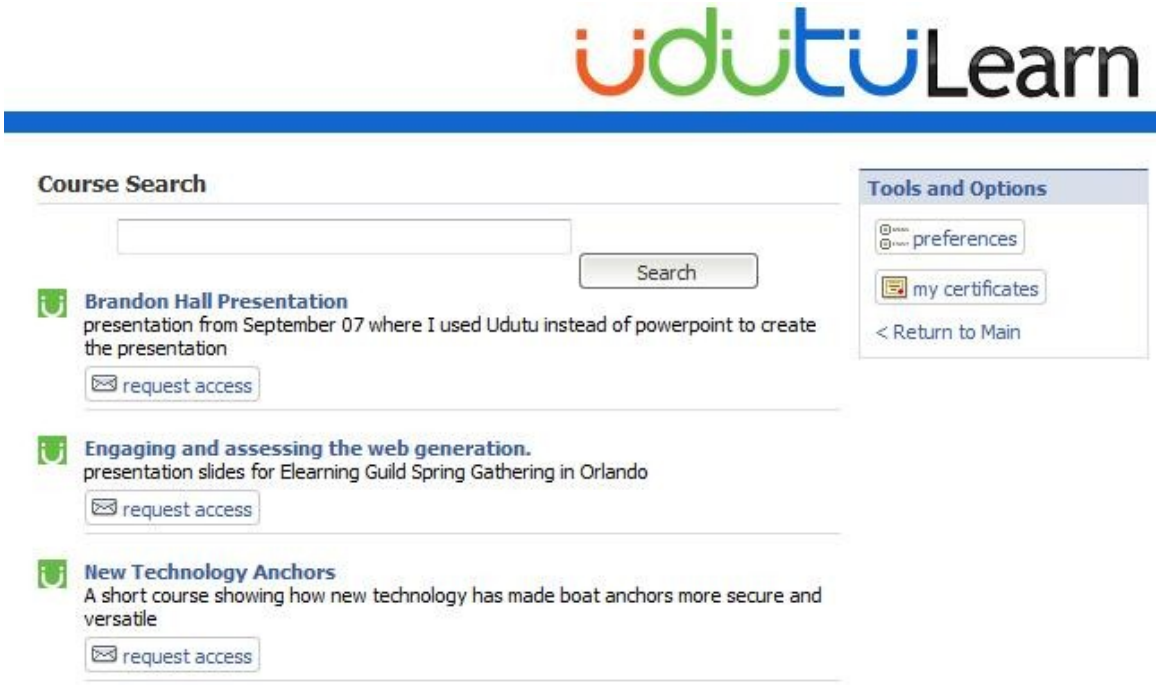


Figure 6.11 - course search

Course Page:

The course page is the learner's main interface to a given course. It is through here that they can take the course and view details about how they are progressing through the course. From a technical standpoint, it is this page that actually contains the SCORM API for the course. If the learner navigates the browser away from this page, Javascript will attempt to close the course window (to reduce the likelihood of the learner attempting to navigate through the course and not have their progress recorded)

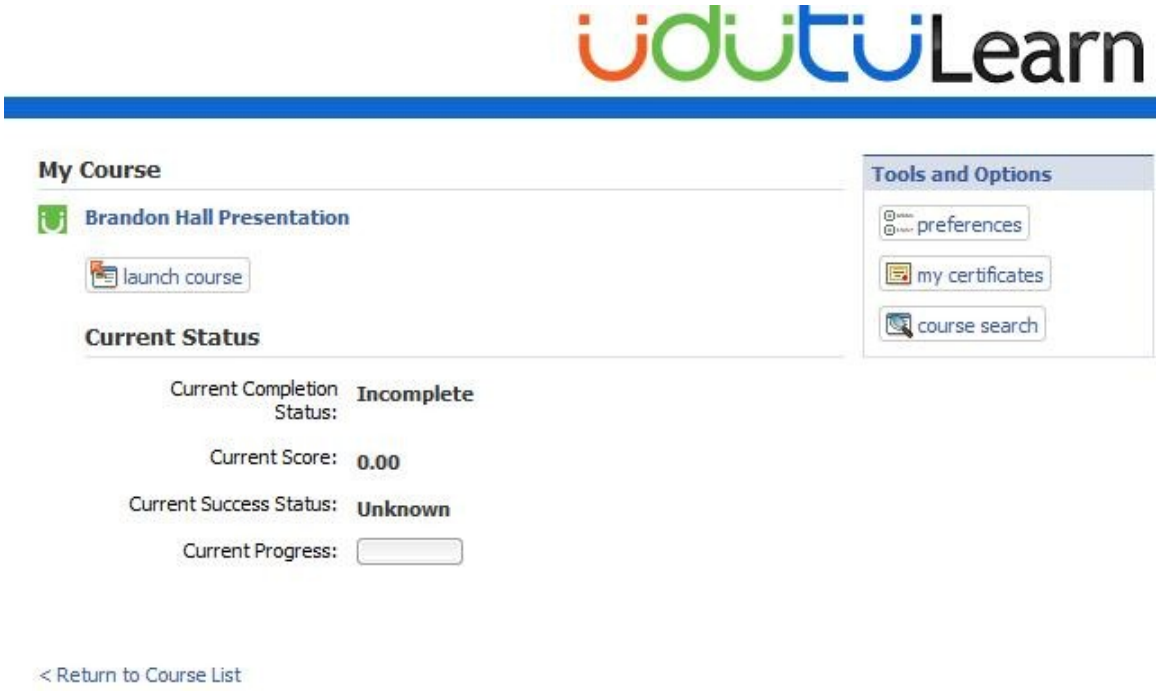


Figure 6.12 - take course

Discussion:

Similar to the Instructor 'Discussion' page, but all comments are public (all users with access to the course will be able to see comments posted through this page).

Message Instructor:

Will bring the learner to a page where they can compose a message that only Instructors of the course will see.

7 Challenges

Challenges in creating the product primarily involved being beholden to the functionality and accessibility provided by Facebook. If Facebook is offline, the LMS is offline. If specific aspects of the Facebook API breaks[38], the LMS does not work. If it wasn't possible to generate a specific set of data through the provided API without making a potentially large sequence of synchronous calls, you either had to live without the functionality that dataset would provide, or offer it only on demand and at a slow quality of service. These challenges will apply to any Social Network or user/group management/authentication portal the gadget plugs into.

Another challenge is related to how what would be ordinary data relationships that could be easily and expediently addressed through SQL can no longer be handled in that way.

38 There was a period of several weeks in June, 2008 where it wasn't possible to discover groups a user belonged to if the group was recently created, making it impossible to assign course permissions to a newly created group.

Data must be accessed through web services, and the relationships joined through a more cumbersome programmatic mechanism to provide necessary UI data or to produce meaningful reports on learner activity.

Further challenges relate to the difficulty in instructors getting learners to add the application (it is never simpler than a multi-step process). Additionally, adding learners to a course that aren't a friend or in a group that you are also in is impossible through Facebook's API (you can't discover anyone that you don't in some way already have a Facebook relationship with). It's also impossible to bulk add learners if they don't already have a Facebook account (or even if they do, if the userIDs are not accessible). It is also impossible to automate the adding and deleting of users from a Facebook Group, since the API does not allow for Group management, and this functionality is not available to the Groups application natively.

8 Future

Future developments include:

- a more thorough implementation of the SCORM 2004 specification
- a course marketplace, allowing authors to share and/or sell courses to one another
- a more robust course search and indexing facility, that makes full use of the SCORM 2004 metatags
- a means to create and offer certificates based on course completion
- the ability to create hierarchical learning paths, to structure several courses together with the goal of achieving a particular learning objective
- full implementation of SCORM 1.2
- seamless integration with myUdutu, that doesn't require an account to be created on myUdutu in order to collaboratively author and deliver courses
- full implementation of the OpenSocial API
- enable other portal creators to integrate with the application without needing custom code development from myself

- the ability for instructors to advertise and sell access to their courseware and/or inviligation

Generating reports on learner activity is a requirement for any LMS. Commonly, LMSs provide dozens or hundreds of report possibilities and permutations, with the aim of providing "out-of-the-box" most reports an organization may need to interpret their training data. In traditional LMS applications, the database of this learner data resides on a web-server managed by the organization that purchased the LMS. If a report is not available "out-of-the-box" by the LMS, the organization will have the ability to create their own reports, provided they have the expertise in the organization to do so. Since the proposed unification of Social Networks and LMS learner tracking functionality does not necessarily offer a database that can be directly accessed by an organization making use of it, a mechanism by which an organization can pull this data should be made available – particularly where the solution is implemented for a portal system controlled by the organization. This functionality would be necessarily more difficult to offer to organizations accessing the system through Social Networks, for security reasons.[39] Web Services (i.e. SOAP), are an ideal technology to make this data accessible. Should the LMS gadget availability be broadened to private organization portals, at minimum the following functions are necessary[40]:

- GetCourseOfferingPermissionsByLMSID

39 There is no mechanism in Facebook, for example, to remotely authenticate a user, and thereby derive what data that user can and cannot see. For custom iterations of the LMS gadget, this is less of a problem, since global access to data for that iteration is not a security risk (or at least no more of one than if the database for learner activity resided on one of their servers).

40 Security considerations concerning access to data (password protection, IP access restrictions, etc.) are assumed

    - where given the supplied LMSID (private iteration of the LMS Gadget), all CourseOffering Permissions for Groups and Users are returned.
- GetCourseOfferingsByLMSID
    - return all courseofferings for a given LMSID
- GetCourseOfferingsByLMSIDUserID
    - return all courseoffering records for a given user
- GetSCOAttemptsByCourseOfferingID
    - return all records for a given course offering id
- GetSCOAttemptsByLMSIDUserIDs
    - return all records for a given list of users

For these methods, all group and user ids that are supplied as parameters and returned in results would be the same ids as those on the portal.

9 Conclusion

In this thesis essay I've described some of the history and function of eLearning as it applies to Learning Management Systems, and Social Networks (and, as a corollary, web portals in general), and suggested a way that these two concepts have natural common ground. Furthermore, I've specified a technological approach to merging the two technologies. Finally, I've demonstrated how these concepts and technologies can be merged by creating a real-world example of this integration. Social Networks such as Facebook appear to indicate what the future of the Internet will look like, in terms of how it is commonly used by the majority. Through APIs made available by these networks to extend their functionality, eLearning may finally be something that is inexpensive and easy to use – without needing technology experts to help set up and run – and may finally enable learners of all persuasions to access the proof-based learning they desire with the convenience and affordability of the Internet.

References

Sharable Content Object Reference Model (R) 2004 3rd Edition Content Aggregation Model Version 1.0, Advanced Distributed Learning (ADL), SCORM(R) 2004 3rd Edition Content Aggregation Model (CAM) Version 1.0, November 14, 2006

Sharable Content Object Reference Model (R) 2004 3rd Edition Overview Verison 1.0, Advanced Distributed Learning (ADL), SCORM(R) 2004 3rd Edition Overview Version 1.0, November 14, 2006

Sharable Content Object Reference Model (R) 2004 3rd Edition Run-Time Environment Version 1.0, Advanced Distributed Learning (ADL), SCORM(R) 2004 3rd Edition Run-Time Environment (RTE) Version 1.0, November 14, 2006

Sharable Content Object Reference Model (R) 2004 3rd Edition Sequencing and Navigation Version 1.0, Advanced Distributed Learning (ADL), SCORM(R) 2004 3rd Edition Sequencing and Navigation Version 1.0, November 14, 2006

Functionality and SCORM-compliancy Evaluation of eLearning Tools
Ganchev, I.; O'Droma, M.; Andreev, R.;
Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on18-20 July 2007
Page(s):467 - 469
Digital Object Identifier 10.1109/ICALT.2007.149

Lemonade Stand. (2008). Retrieved November 12, 2008, from http://codenautics.com/lemonade/

B.F. Skinner. (2008). Retrieved November 12, 2008, from http://www.coe.uh.edu/courses/cuin6373/idhistory/skinner.html

Computerized Adaptive Tests. ERIC Digest. (2008). Retrieved November 12, 2008, from http://www.ericdigests.org/pre-9213/tests.htm

SCORM: The eLearning Standard; Why it matters, what's in it for you, Best practices in getting started. (2009). Retrieved March 09, 2009, from http://sorubank.ege.edu.tr/~e190411147/scorm/scorm4.pdf

myUdutu - Sign In. (2008). Retrieved November 12, 2008, from http://www.myudutu.com

Moodle.org: open-source community-based tools for learning. (2008). Retrieved November 12, 2008, from http://www.moodle.org

Advanced Distributed Learning - Start Download. (2008). Retrieved November 12, 2008, from http://www.adlnet.gov/downloads/AuthNotReqd.aspx?FileName=IP2D0116T0900.ppt&ID=92

The Cover Pages: SCORM Version 1.2 Released. (2008). Retrieved November 12, 2008, from http://xml.coverpages.org/SCORM-Version12-Ann.html

No Title (Correspondence Transcript). (Version 5, February 11, 1988). Retrieved November 12, 2008, from http://www.bbsdocumentary.com/software/AAA/AAA/CBBS/memories.txt

Fundamentals of FOSSIL implementation and use. (Version 5, February 11, 1988). Retrieved November 12, 2008, from http://www.ftsc.org/docs/fsc-0015.001

Facebook | Facebook. (2008). Retrieved November 12, 2008, from http://www.facebook.com/facebook

Facebook | Statistics. (2008). Retrieved November 12, 2008, from http://www.facebook.com/press/info.php?statistics

Will Facebook's older users drive away all the young kids? (October 14, 2007). Nicholas Deleon. Retrieved November 12, 2008, from http://www.crunchgear.com/2007/10/14/will-facebooks-older-users-drive-away-all-the-young-kids/

Facebook Developers | Facebook Developers News. (2008). Retrieved November 12, 2008, from http://developers.facebook.com/news.php?blog=1&year=2006&month=8

Google Launches OpenSocial - John Battelle's Searchblog. (October 30, 2007). John Battelle. Retrieved November 12, 2008, from http://battellemedia.com/archives/004058.php

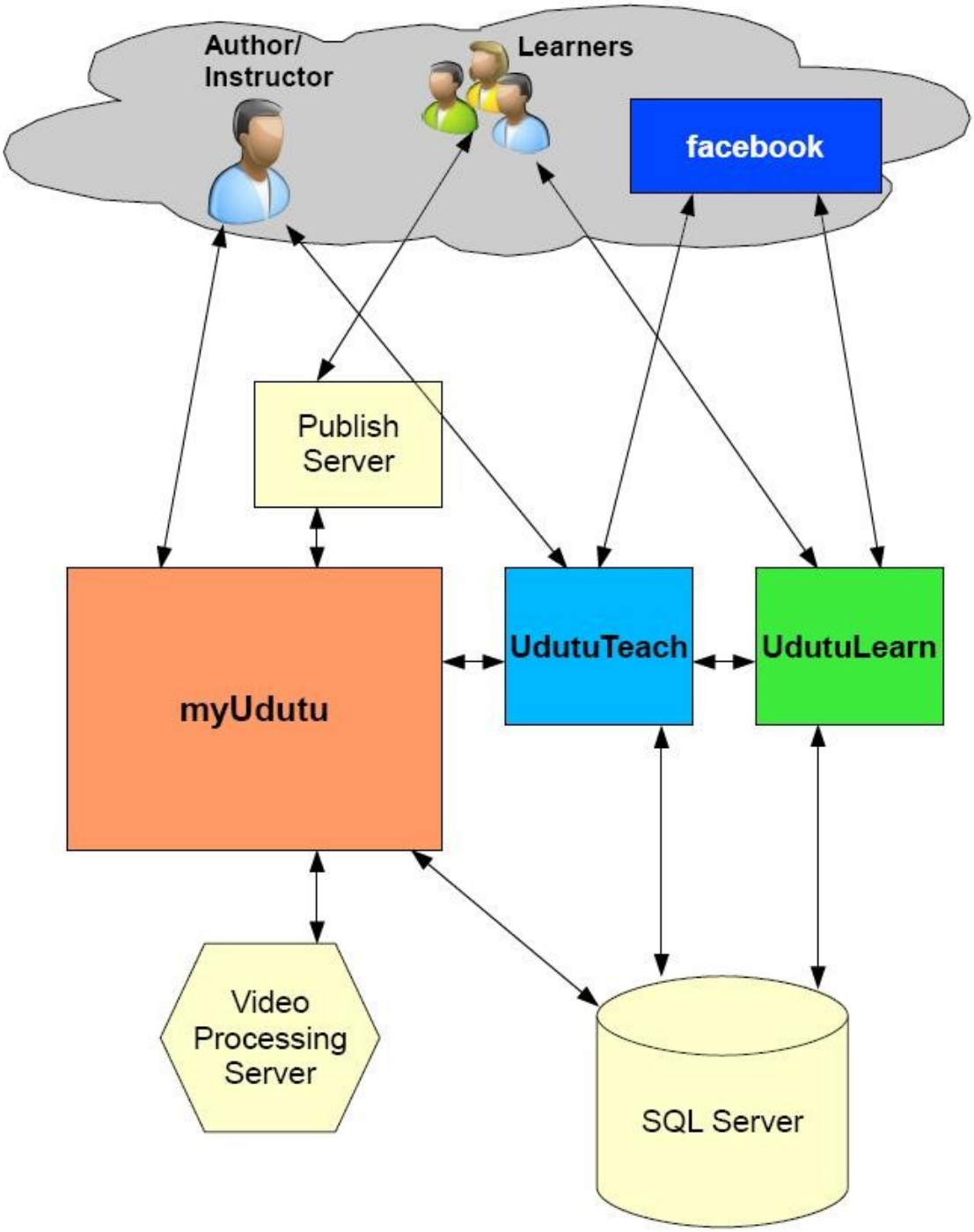
API - Facebook Developers Wiki. (2008). Retrieved November 12, 2008, from http://wiki.developers.facebook.com/index.php/API

ASP.NET AJAX Controls, Web UI Components | Telerik. (2008). Retrieved November 12, 2008, from http://www.telerik.com/products/aspnet-ajax/overview.aspx

.NET Zip Library #ziplib (SharpZipLib). (2008). Christopher Wille. Retrieved November 12, 2008, from http://www.icsharpcode.net/OpenSource/SharpZipLib/

Appendix A

Complete Udutu eLearning System Architecture:



- Requires

  - SQL Server 2005 or newer
  - Windows Server 2003 or newer
  - ASP.NET 2.0 Framework
- Each component may reside on their own server (to aid scalability), but can all reside on the same server for smaller deployments

myUdutu

myUdutu is the course authoring web application, and the flagship product of Udutu. It is designed to be an easy to use system for authoring online training in a framework that enables multiple authors to collaborate on the same materials.

It has two sub-components:

Basic Publish Server

This is where courses are published for viewing by learners or stakeholders in the authoring process. There is no learner tracking or access control when courses are delivered through the Publish Server (this additional functionality is delivered through Udutu Teach & Learn on a different pricing model)

Video Processing Server

This application optimizes video for streaming utilizing the same technology that YouTube and other well-known video servers are based upon

Udutu Teach & Learn (not all items fully implemented)

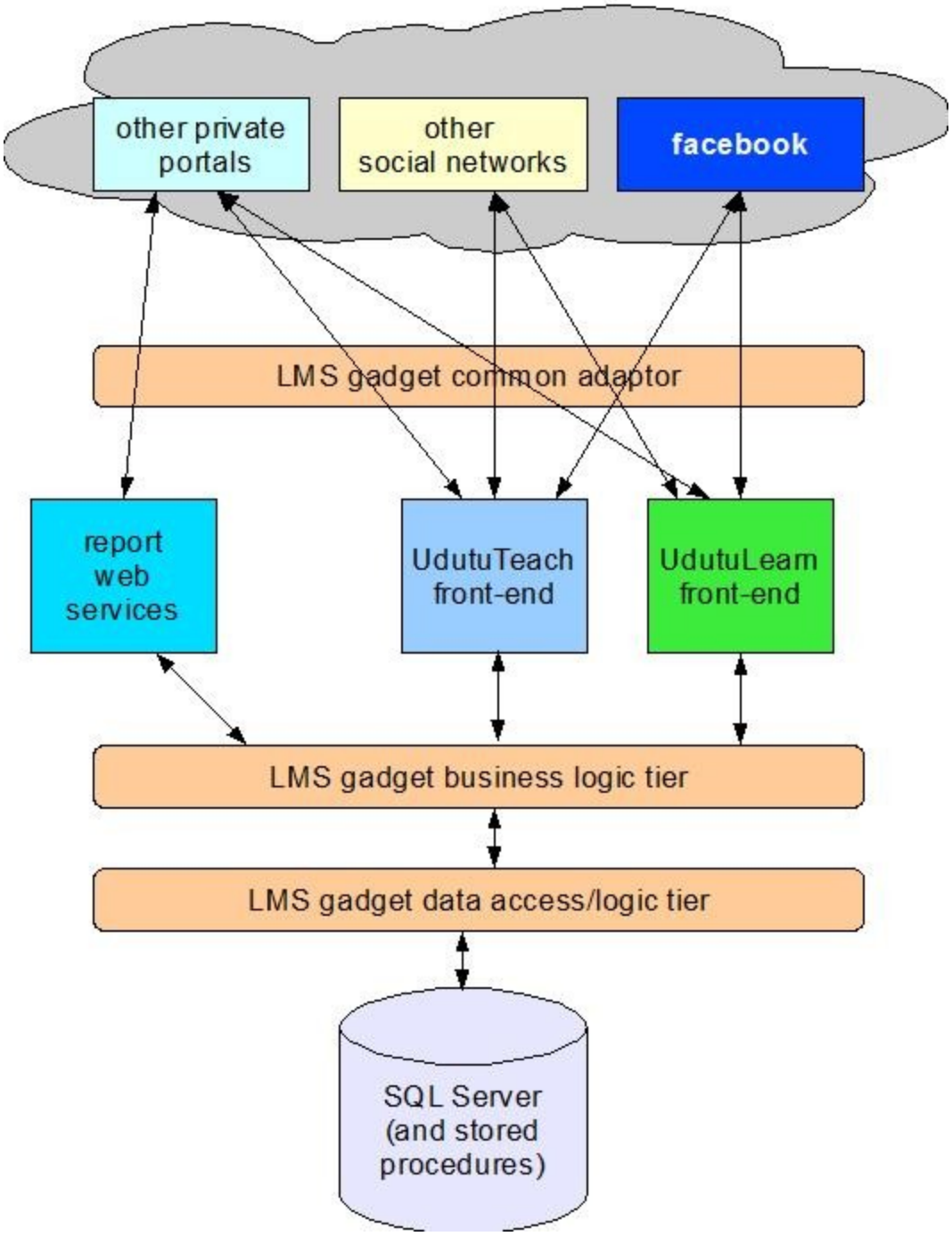
UdutuTeach

UdutuTeach provides teachers, corporations, and advertisers to place and promote their learning materials, as well as track learner progress, issue certification, and optionally collect revenues from tuition. It is available through Facebook, and the core technology can be readily adapted to SharePoint or any other proprietary user management system, portal, or social network. It integrates seamlessly and accepts content from myUdutu, as well as other SCORM 2004 compliant course authoring systems.

UdutuLearn

UdutuLearn provides learners with a way to access courses to which they've been granted permission, as well as obtain certification for coursework. It is tightly integrated with UdutuTeach.

Udutu Teach & Learn System Architecture:



LMS gadget common adaptor: the base class containing interface methods common to all Social Network/portal APIs. Specific implementations are delegated to inherited classes.

Report web services: SOAP methods that will return learner activity for a specific implementation.

Udutu Teach & Learn front-ends: the UdutuTeach interfaces, with UI branding abstracted for particular implementations

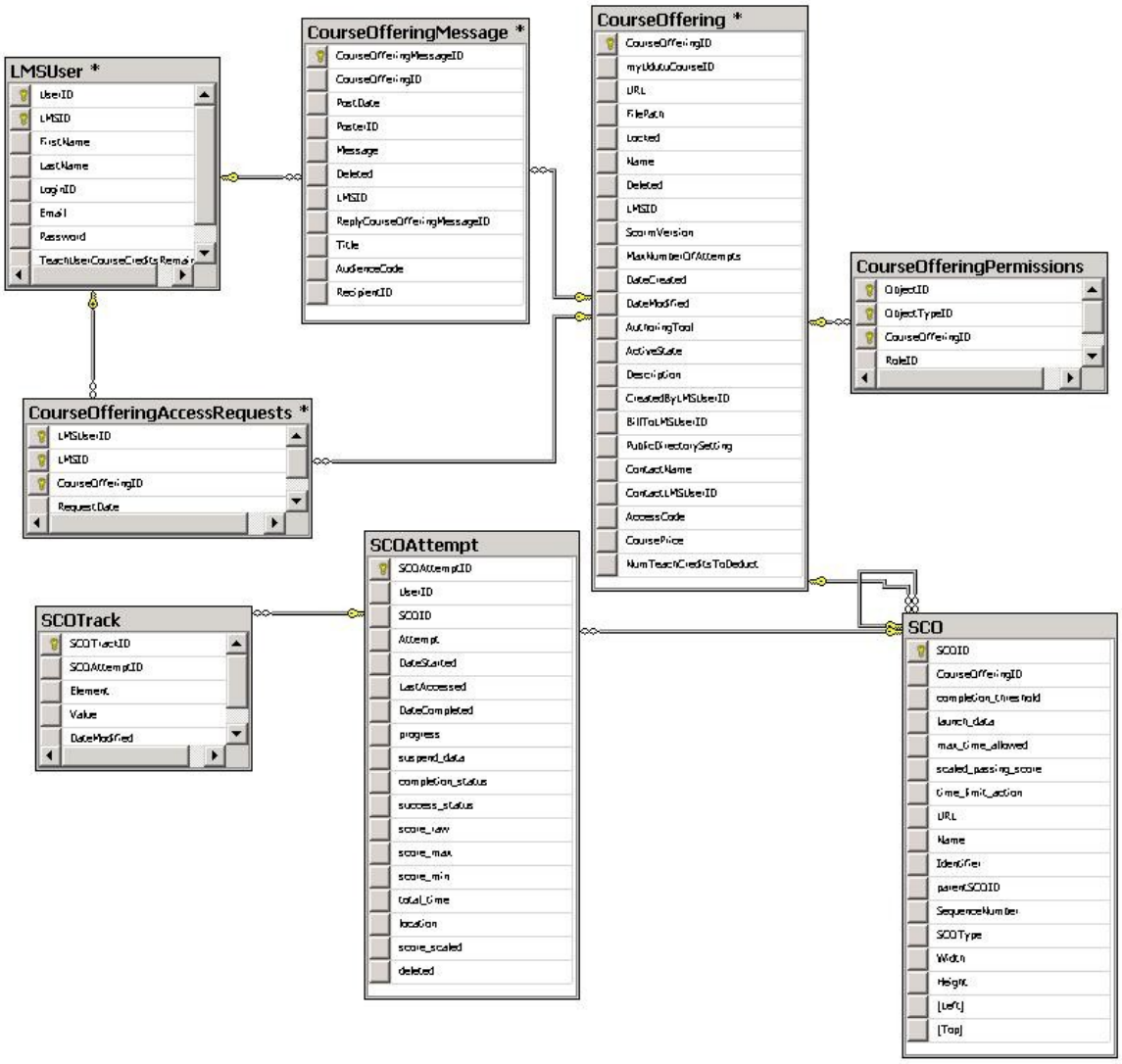
LMS gadget business logic tier: logic and class factories common to all UI and web-services

LMS gadget data access/logic tier: data for the system passes through this tier, abstracting data access and storage from a specific database implementation, allowing for the possibility of other database back-ends

SQL Server: SQL Server 2005 or newer is the data warehouse for this implementation.

Appendix B

Simplified database schema for LMS Gadget:



* Tables not included in diagram are any tables related to temporary caching of data or eCommerce. Also, not all LMSUser relationships have been filled in, since some LMSIDs are derived from parent tables.

Appendix C

Principal Javascript and C# source code for SCORM 2004 API processes and imsmanifest.xml interpretation:

//Javascript code to store and extract course information passed through //the SCORM API

```
//prototype definition for the hashtable to be used for SCORM data
function SCORMHashtable(){
    this.clear = hashtable_clear;
    this.containsKey = hashtable_containsKey;
    this.containsValue = hashtable_containsValue;
    this.get = hashtable_get;
    this.isEmpty = hashtable_isEmpty;
    this.keys = hashtable_keys;
    this.put = hashtable_put;
    this.remove = hashtable_remove;
    this.size = hashtable_size;
    this.toString = hashtable_toString;
    this.values = hashtable_values;
    this.update = hashtable_update;
    this.cleanDirty = hashtable_cleanDirty;
    this.dumpDirty = hashtable_dumpDirty;
    this.getCount = hashtable_getCount;
    this.getKeyString = hashtable_getKeyString;
    this.hashtable = new Array();

}


//get the value for stored learner activity
//some values are hardcoded for this iteration
//(i.e. they will be the same for all users)
//so there is no need to persist them to the database
function hashtable_get(key){
    var ptnCount = /count/g;
    if (ptnCount.test(key))
    {
    //the user wants the count of an object
        return this.getCount(key);
    }
    else
    {
      switch (key)
      {
        case 'cmi._version':
          return '1.0';
          break;
        case 'cmi.comments_from_learner._children':
          return '';
          break;
        case 'cmi.comments_from_lms._children':
          return '';
          break;
```

```
        case 'cmi.credit':
          return 'credit';
          break;
        case 'cmi.interactions._children':
          return
'id,type,timestamp,correct_responses,learner_response,result,descriptio
n,weighting,latency';
          break;
        case 'cmi.learner_preference._children':
          return 'audio_level,language,delivery_speed,audio_captioning';
          break;
        case 'cmi.mode':
          return 'normal';
          break;
        case 'cmi.objectives._children':
          return
'id,score,success_status,completion_status,description';
          break;
        case 'cmi.score._children':
          return 'scaled,raw,max,min';
          break;
        default:
          return this.hashtable[key];
          break;
      }
    }
}

function hashtable_getCount(key)
{
  var strAllKeys = this.getKeyString();
  if (key == 'cmi.interactions._count')
  {
    var ptnInteractions = /cmi\.interactions\.\d+\.id/g;
    var numMatches = strAllKeys.match(ptnInteractions);
    if (numMatches == null) return 0;
    else return numMatches.length;
  }
  else if(key == 'cmi.objectives._count')
  {
    var ptnObjectives = /cmi\.objectives\.\d+\.id/g;
    var numMatches = strAllKeys.match(ptnObjectives);
    if (numMatches == null) return 0;
    else return numMatches.length;
  }
  else
  {
    var ptnCntCorrectResponses =
/cmi\.interactions\.\d+\.correct_responses\._count/g;
    if (ptnCntCorrectResponses.test(key))
    {
      var ptnDigits = /\d+/g;
      var num = key.match(ptnDigits);
      var ptnCorrectResponses = new RegExp("cmi\.interactions\." +
ptnDigits + "\.correct_responses\.pattern","g");
      var numMatches = strAllKeys.match(ptnCorrectResponses);
      if (numMatches == null) return 0;
```

```
      else return numMatches.length;
    }
  }
}

//return all values stored in the hashtable as a single
//delimited string
function hashtable_toString(){
    var result = "";
    for (var i in this.hashtable)
    {
        if (this.hashtable[i] != null)
            result += "{~{" + i + "}~},{~{" + this.hashtable[i] + "}~}\
n";
    }
    return result;
}


//define the SCORM 2004 API methods and variables
var API_1484_11 = window;
var LastErrorCode = '0';
var SCOHa = new SCORMHashtable();


function Initialize(parameter)
{
  //the parameter is unused - reserved by SCORM for future use
  return 'true';
}

function Terminate(tmp)
{
  //the parameter is unused - reserved by SCORM for future use
  return 'true';
}

function GetValue(parameter)
{
  //get the parameter value from the hashtable
  return SCOHa.get(parameter);

}

function GetLastError()
{
  return LastErrorCode;
}

function GetErrorString(parameter)
{
 switch(parameter)
 {
  case 0:
   return 'No Error';
  default:
   return 'Unknown';
```

```
  }
}

function SetValue(parameter, value)
{
    //store the value in the hashtable
    SCOHa.update(parameter,value);
    return 'true';
}

function Commit()
{//sebd all values in the SCORM hashtable to the server via AJAX

  SetValue('cmi.entry','resume');
  var saveVal = SCOHa.dumpDirty();
  var hdnSCORMState = document.getElementById('<%=
hdnSCORMState.ClientID %>');
  hdnSCORMState.value = saveVal;
  //store the learner state for the SCO in a hidden field, and post
  //back to the server (via AJAX to keep page alive)
  var btnSubmit = document.getElementById('<%= btnSubmitSCORM.ClientID
%>');
  btnSubmit.click();
  return 'true';
}



public string GenerateSCORMJS()
{
    StringBuilder strJS = new StringBuilder("");

    strJS.Append("SCOHa.put('" + SCORM2004Constants.Progress + "','" +
Progress.ToString() + "');");
    strJS.Append("SCOHa.put('" + SCORM2004Constants.SuspendData + "','"
+ SuspendData.Replace("'", "\\'") + "');");
    strJS.Append("SCOHa.put('" + SCORM2004Constants.Location + "','" +
Location.Replace("'", "\\'") + "');");
    strJS.Append("SCOHa.put('" + SCORM2004Constants.CompletionStatus +
"','" + CompletionStatus.ToString().ToLower() + "');");
    strJS.Append("SCOHa.put('" + SCORM2004Constants.SuccessStatus +
"','" + SuccessStatus.ToString().ToLower() + "');");

    //only show scores if there is a score
    if (ScoreMax >= 0)
        strJS.Append("SCOHa.put('" + SCORM2004Constants.MaxScore + "','"
+ ScoreMax.ToString() + "');");
    if (ScoreMin >= 0)
        strJS.Append("SCOHa.put('" + SCORM2004Constants.MinScore + "','"
+ ScoreMin.ToString() + "');");
    if (ScoreRaw >= 0)
        strJS.Append("SCOHa.put('" + SCORM2004Constants.RawScore + "','"
+ ScoreRaw.ToString() + "');");
    if (ScoreScaled >= 0)
        strJS.Append("SCOHa.put('" + SCORM2004Constants.ScaledScore +
"','" + ScoreScaled.ToString() + "');");
```

```
    if (!this.SCOTrackElements.ContainsKey(SCORM2004Constants.Entry))
        strJS.Append("SCOHa.put('" + SCORM2004Constants.Entry + "','ab-
initio');");

    //add the rest
    foreach (KeyValuePair<string, SCOTrackElement> scoTra in
this.SCOTrackElements)
    {
        strJS.Append("SCOHa.put('" + scoTra.Key + "','" +
scoTra.Value.Value.Replace("'", "\\'") + "');");
    }


    return strJS.ToString();
}

public Dictionary<string, SCOTrackElement> SCOTrackElements
{
    //lazy load this object
    get
    {
        if (_scoTrackElements == null)
        {
            //get scotracks from datatier
            //for each row, add to the dictionary

            _scoTrackElements = new Dictionary<string,
      SCOTrackElement>();

            SCOTrackData trackData = new SCOTrackData();
            DataView dv =
      trackData.GetSCOTrackBySCOAttemptID(_intSCOAttemptID);
            foreach (DataRow dr in dv.Table.Rows)
            {
                if (dr != null)
                {
                    SCOTrackElement scoTrack = new SCOTrackElement(dr);
                    _scoTrackElements.Add(scoTrack.Element, scoTrack);
                }
            }

        }
        return _scoTrackElements;
    }
    set { _scoTrackElements = value; }
}


//Sample dynamic JavaScript output from the above method
//When this javascript is interpreted, the state of the learner's
//previous access of the course is loaded into the javascript hashtable


SCOHa.put('cmi.progress_measure','0.0769231');
SCOHa.put('cmi.suspend_data','66024,66025,66027,66159');
```

```
SCOHa.put('cmi.location','66159');
SCOHa.put('cmi.completion_status','incomplete');
SCOHa.put('cmi.success_status','unknown');
SCOHa.put('cmi.score.max','20.0000000');
SCOHa.put('cmi.score.min','0.0000000');
SCOHa.put('cmi.score.raw','0.0000000');
SCOHa.put('cmi.score.scaled','0.0000000');
SCOHa.put('cmi.entry','resume');
SCOHa.put('cmi.core.session_time','00:03:13');
SCOHa.put('cmi.exit','suspend');
SCOHa.put('cmi.session_time','00:00:07');
```

//Server-side C# code to interpret learner data accumulated in the
//client-side JavaScript hashtable that is sent to the server
//upon a 'Commit' command

```
public void ProcessSCORMVars(string strSCORMVars)
{
      string splitKeyVal = "}~},{~{";
      string[] splitKeyValArr = new string[1];
      splitKeyValArr[0] = splitKeyVal;
      string splitRow = "}~!}";
      string[] splitRowArr = new string[1];
      splitRowArr[0] = splitRow;
      string[] strKeyVals = strSCORMVars.Split(splitRowArr,
      StringSplitOptions.RemoveEmptyEntries);

      eSuccessStatus tmpSuccessStatus = eSuccessStatus.Unknown;

      string[] strKeyValArr = new string[2];
      foreach (string strKeyVal in strKeyVals)
      {
            strKeyValArr = strKeyVal.Split(splitKeyValArr,
            StringSplitOptions.RemoveEmptyEntries);
            if (strKeyValArr.Length > 1)
            {
               switch (strKeyValArr[0])
               {
                 case SCORM2004Constants.Progress:
                   Progress = decimal.Parse(strKeyValArr[1]);
                   break;
                 case SCORM2004Constants.SuccessStatus:
                   if (SuccessStatus != eSuccessStatus.Passed)
                   {
                      switch (strKeyValArr[1])
                      {
                         case "passed":
                           tmpSuccessStatus = eSuccessStatus.Passed;
                           break;
                         case "failed":
                           tmpSuccessStatus = eSuccessStatus.Failed;
                           break;
                         default:
                           tmpSuccessStatus = eSuccessStatus.Unknown;
                           break;
```

```
        }
      }
      else
        tmpSuccessStatus = eSuccessStatus.Passed;
      break;
    case SCORM2004Constants.CompletionStatus:
      switch (strKeyValArr[1])
      {
        case "completed":
           if (CompletionStatus !=
           eCompletionStatus.Completed)
          {
             DateCompleted = DateTime.Now;
          }
          CompletionStatus = eCompletionStatus.Completed;
          break;
        case "incomplete":
          CompletionStatus = eCompletionStatus.Incomplete;
          break;
        case "not attempted":
          CompletionStatus=eCompletionStatus.NotAttempted;
          break;
        default:
          CompletionStatus = eCompletionStatus.Unknown;
          break;
      }
      break;
    case SCORM2004Constants.Location:
      Location = strKeyValArr[1];
      break;
    case SCORM2004Constants.MaxScore:
      ScoreMax = decimal.Parse(strKeyValArr[1]);
      break;
    case SCORM2004Constants.MinScore:
      ScoreMin = decimal.Parse(strKeyValArr[1]);
      break;
    case SCORM2004Constants.RawScore:
      ScoreRaw = decimal.Parse(strKeyValArr[1]);
      break;
    case SCORM2004Constants.ScaledScore:
      ScoreScaled = decimal.Parse(strKeyValArr[1]);
      break;
    case SCORM2004Constants.SuspendData:
      SuspendData = strKeyValArr[1];
      break;
    case SCORM2004Constants.TotalTime:
      TotalTime = int.Parse(strKeyValArr[1]);
      break;
    case SCORM2004Constants.LearnerID:
      break;
    case SCORM2004Constants.LaunchData:
      break;
    case SCORM2004Constants.CompletionThreshold:
      break;
    case SCORM2004Constants.Credit:
      break;
    default:
```

```
                    SCOTrackElement scotEl = new SCOTrackElement();
                    scotEl.Element = strKeyValArr[0];
                    scotEl.Value = strKeyValArr[1];
                    scotEl.SCOAttemptID = SCOAttemptID;
                    scotEl.DateModified = DateTime.Now;
                    scotEl.Save();
                    break;
                }
            }
        }

        //if the learner has passed, then they will have passed no matter if
they return to the SCO and do something to fail
        if (SCO.ScaledPassingScore > 0 && CompletionStatus ==
eCompletionStatus.Completed && SuccessStatus != eSuccessStatus.Passed)
        {
            if (SCO.ScaledPassingScore > ScoreScaled && ScoreMax > 0)
            {
              SuccessStatus = eSuccessStatus.Failed;
            }
            else
              SuccessStatus = eSuccessStatus.Passed;
        }
        else
          SuccessStatus = tmpSuccessStatus;
    }
```

//C# code to process a basic SCORM 2004 IMSManifest file:

```
public eProcessSCORMResult ProcessIMSManifest(string strIMSManifestDir,
string strBaseURL, string strFilePath, bool blnIsEdit)
{
    _gblSeqNum = 0;
    Dictionary<int, SCO> _oldSCOs = new Dictionary<int,SCO>();
    if (!System.IO.File.Exists(strIMSManifestDir + "imsmanifest.xml"))
    {
        return eProcessSCORMResult.IMSManifestNotFound;
    }

    XmlDocument doc = new XmlDocument();
    doc.Load(strIMSManifestDir + "imsmanifest.xml");

    string strVersion = doc.GetElementsByTagName("schemaversion")
[0].InnerText;

    if (strVersion == "CAM 1.3")
    {
        if (blnIsEdit)
        {

            foreach (SCO sco in SCOs.Values)
            {
                _oldSCOs.Add(sco.SCOID, sco);
            }

        }
```

```
        XmlNamespaceManager namespaceManager = new
XmlNamespaceManager(doc.NameTable);
        namespaceManager.AddNamespace("ab",
"http://www.imsglobal.org/xsd/imscp_v1p1");
        namespaceManager.AddNamespace("adlcp",
"http://www.adlnet.org/xsd/adlcp_v1p3");
        namespaceManager.AddNamespace("xsi",
"http://www.w3.org/2001/XMLSchema-instance");
        namespaceManager.AddNamespace("imsss",
"http://www.imsglobal.org/xsd/imsss");

        XPathNavigator nav = doc.CreateNavigator();
        XPathNodeIterator iterResourcesBase =
nav.Select("descendant::ab:resources", namespaceManager);
        iterResourcesBase.MoveNext();
        string strResourcesBase = string.Empty;// =
iterBase.Current.GetAttribute(tmp1,tmp2 );
        if (iterResourcesBase.Current.HasAttributes)
        {
            iterResourcesBase.Current.MoveToFirstAttribute();
            do
            {
                if (iterResourcesBase.Current.Name == "xml:base")
                    strResourcesBase = iterResourcesBase.Current.Value;
            } while (iterResourcesBase.Current.MoveToNextAttribute());
        }
        XPathNodeIterator iterTitle =
nav.Select("descendant::ab:organization/ab:title", namespaceManager);
        iterTitle.MoveNext();
        if (iterTitle.Count == 0)
            return eProcessSCORMResult.NoEntryPointDetected;
        else
            Name = iterTitle.Current.Value;


        XPathNodeIterator iter =
nav.Select("descendant::ab:organization/ab:item", namespaceManager);
        if (iter.Count == 0)
        {
            return eProcessSCORMResult.NoEntryPointDetected;
        }
        else// (iter.Count == 1)
        {
            while (iter.MoveNext())
            {
                if (iter.Current.GetAttribute("isvisible", "") !=
"false")
                {
                    ProcessIMSItem(strBaseURL, strFilePath,
namespaceManager, nav, strResourcesBase,
iter.Current,null,blnIsEdit,_oldSCOs);
                }
            }
        }
        foreach (SCO sco in _oldSCOs.Values)
        {
```

```
                SCOs.Remove(sco.SCOID);
                //TODO: DELETE the sco!! (how? logically?)

            }
            ScormVersion = eSCORM_Version.SCORM_2004;

            return eProcessSCORMResult.Success;
        }
        else
        {
            return eProcessSCORMResult.UnsupportedSCORM;
        }
    }

    private void ProcessIMSItem(string strBaseURL, string strFilePath,
    XmlNamespaceManager namespaceManager, XPathNavigator navRoot, string
    strResourcesBase, XPathNavigator navItem, SCO parentSCO, bool
    blnIsEdit, Dictionary<int,SCO> _oldSCOs)
    {
        _gblSeqNum++;
        string strIdentifierRef = navItem.GetAttribute("identifierref", "");
        string strIdentifer = navItem.GetAttribute("identifier", "");
        Regex reg = new Regex("Module\\d\\d+");
        Regex reg2 = new Regex("ModuleAssets\\d\\d+");
        if (reg.IsMatch(strIdentifer) && reg2.IsMatch(strIdentifierRef))
            AuthoringTool = eAuthoringTool.myUdutu_SCORM_Import;
        else
            AuthoringTool = eAuthoringTool.Unknown;


        if (strIdentifierRef != null && strIdentifierRef != string.Empty)
        {

            SCO tmpSCO = GetSCOBySCORMIdentifier(strIdentifer);
     //need to find out if this sco already exists??
            if (tmpSCO == null)
                tmpSCO = new SCO();
            tmpSCO.Identifier = strIdentifer;
            tmpSCO.CourseOfferingID = CourseOfferingID;
            tmpSCO.Sequence = _gblSeqNum;
            if (parentSCO != null)
                tmpSCO.ParentID = parentSCO.SCOID;

            string strParameters = navItem.GetAttribute("parameters", "");

            XPathNodeIterator iterTitle = navItem.Select("/ab:title",
    namespaceManager);
            iterTitle.MoveNext();
            tmpSCO.Name = iterTitle.Current.Value;
            XPathNodeIterator iterTimeLimitAction =
    navItem.Select("/adlcp:timeLimitAction", namespaceManager);
            string strTimeLimitAction = string.Empty;
            if (iterTimeLimitAction.MoveNext())
            {
                strTimeLimitAction = iterTimeLimitAction.Current.Value;
            }
            tmpSCO.TimeLimitAction = strTimeLimitAction;
```

```
        XPathNodeIterator iterDataFromLMS =
navItem.Select("/adlcp:dataFromLMS", namespaceManager);
        string strDataFromLMS = string.Empty;
        if (iterDataFromLMS.MoveNext())
        {
            strDataFromLMS = iterDataFromLMS.Current.Value;
        }
        tmpSCO.LaunchData = strDataFromLMS;

        XPathNodeIterator iterCompletionThreshold =
navItem.Select("/adlcp:completionThreshold", namespaceManager);
        decimal decCompletionThreshold = -1;
        if (iterCompletionThreshold.MoveNext())
        {
            decCompletionThreshold =
decimal.Parse(iterCompletionThreshold.Current.Value);
        }
        tmpSCO.CompletionThreshold = decCompletionThreshold;

        XPathNodeIterator iterRes =
navRoot.Select("descendant::ab:resource[@identifier='" +
strIdentifierRef + "']", namespaceManager);
        iterRes.MoveNext();
        string strHref = iterRes.Current.GetAttribute("href", "");
        iterRes.Current.MoveToFirstAttribute();
        string strResBase = string.Empty;
        do
        {
            if (iterRes.Current.Name == "xml:base")
            {
                strResBase = iterRes.Current.Value;
            }
            else if (iterRes.Current.Name == "adlcp:scormType")
            {
                if (iterRes.Current.Value == "sco")
                    tmpSCO.SCOType = SCO.eSCOType.SCO;
                else
                    tmpSCO.SCOType = SCO.eSCOType.Asset;
            }
        } while (iterRes.Current.MoveToNextAttribute());

        tmpSCO.URL = strBaseURL + strResBase + strResourcesBase +
strHref + (strParameters == string.Empty ? string.Empty : "?" +
strParameters);
        tmpSCO.Save();
        if (!_SCOs.ContainsKey(tmpSCO.SCOID))
            _SCOs.Add(tmpSCO.SCOID, tmpSCO);
        if (blnIsEdit)
        {
            if (_oldSCOs.ContainsKey(tmpSCO.SCOID))
            {
                _oldSCOs.Remove(tmpSCO.SCOID);
            }
        }
    }
    else
```

```
    {
        //this is probably a parent node

        XPathNodeIterator iterNextItem = navItem.Select("/ab:item",
namespaceManager);
        if (iterNextItem.MoveNext())
        {
            SCO tmpParentSCO = GetSCOBySCORMIdentifier(strIdentifer);
            if (tmpParentSCO == null)
                tmpParentSCO = new SCO();
            tmpParentSCO.Identifier = strIdentifer;
            XPathNodeIterator iterTitle =
iterNextItem.Current.Select("/ab:title", namespaceManager);
            iterTitle.MoveNext();
            tmpParentSCO.Name = iterTitle.Current.Value;
            tmpParentSCO.SCOType = SCO.eSCOType.Container;
            tmpParentSCO.Sequence = _gblSeqNum;
            tmpParentSCO.CourseOfferingID = CourseOfferingID;
            tmpParentSCO.Save();
            if (!_SCOs.ContainsKey(tmpParentSCO.SCOID))
                _SCOs.Add(tmpParentSCO.SCOID, tmpParentSCO);
            if (blnIsEdit)
            {
                if (_oldSCOs.ContainsKey(tmpParentSCO.SCOID))
                {
                    _oldSCOs.Remove(tmpParentSCO.SCOID);
                }
            }
            ProcessIMSItem(strBaseURL, strFilePath, namespaceManager,
navRoot, strResourcesBase, iterNextItem.Current, tmpParentSCO,
blnIsEdit,_oldSCOs);
        }

    }
    ScormVersion = eSCORM_Version.SCORM_2004;
}
```

All source code for the implementation is several thousand lines long, and is therefore too long to print. It is © Udutu Learning Systems, though is available on request to any thesis committee members. (contact me at knoesgaard@udutu.com and I'll email it to you)