NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

NEXTGEN NAVY ELEARNING TRACKING

by

William E. Miller

December 2014

Thesis Advisor: Man-Tak Shing
Co-Advisor: Arijit Das

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2014	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE
NEXTGEN NAVY ELEARNING TRACKING

5. FUNDING NUMBERS

6. AUTHOR(S) William E. Miller

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200words)

The Navy's eLearning (NeL) computer-based learning system relies on a Learning Management System (LMS) for content delivery and tracking learning information. One major obstacle with NeL's current LMS implementation is that tracking of learning can only be done while a user is on a desktop computer using an Internet browser to connect to the LMS software. However, not all learning takes place within an Internet browser on a desktop computer. The Experience-API (xAPI), also known as Tin Can API and SCORM 2.0, is a standard maintained by Advanced Distributed Learning (ADL) that decouples the tracking of learning information from the content delivery. Any piece of software implementing the xAPI standard running on any networked device can track learning activity and store that data inside of a Learning Record Store (LRS). A prototype system was developed in a virtual environment to showcase the use of the xAPI/LRS to track quiz data, and the quiz data could then be synced from the LRS to the LMS. The prototype showed that xAPI, along with its LRS, can overcome the NeL's AtlasPro LMS limitation of only tracking learning from a user's desktop computer using an Internet browser.

14. SUBJECT TERMS Experience API, xAPI, Tin Can API, SCORM, LRS, LMS, eLearning, NeL	**15. NUMBER OF PAGES** 65
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

NEXTGEN NAVY ELEARNING TRACKING

William E. Miller
Civilian, Department of Defence, Defense Manpower Data Center
B.S., California State University–Monterey Bay, May 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2014

Author: William E. Miller

Approved by: Man-Tak Shing
Thesis Advisor

Arijit Das
Co-Advisor

Peter J. Denning
Chair, Department of CS

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Navy's eLearning (NeL) computer-based learning system relies on a Learning Management System (LMS) for content delivery and tracking learning information. One major obstacle with NeL's current LMS implementation is that tracking of learning can only be done while a user is on a desktop computer using an Internet browser to connect to the LMS software. However, not all learning takes place within an Internet browser on a desktop computer. The Experience-API (xAPI), also known as Tin Can API and SCORM 2.0, is a standard maintained by Advanced Distributed Learning (ADL) that decouples the tracking of learning information from the content delivery. Any piece of software implementing the xAPI standard running on any networked device can track learning activity and store that data inside of a Learning Record Store (LRS). A prototype system was developed in a virtual environment to showcase the use of the xAPI/LRS to track quiz data, and the quiz data could then be synced from the LRS to the LMS. The prototype showed that xAPI, along with its LRS, can overcome the NeL's AtlasPro LMS limitation of only tracking learning from a user's desktop computer using an Internet browser.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
A. THESIS OBJECTIVE....1
B. THESIS OUTLINE....2
II. BACKGROUND....3
III. PROTOTYPE SYSTEM DESIGN....9
A. STAKEHOLDERS....9
B. USE CASES....9
C. COMPONENTS....13
D. ARCHITECTURE....16
E. SEQUENCE DIAGRAMS....16
1. Student User Quiz Submission....16
2. Administrator User Data Sync....17
3. Teacher User View Grade....18
F. CLASS DIAGRAMS....19
G. XAPI/LRS EXTENSIONS....22
IV. PROTOTYPE SYSTEM IMPLEMENTATION....25
A. STUDENT USER AND LRS DATA INSERT....28
B. ADMINISTRATOR USER AND DATA SYNC FUNCTION....34
C. TEACHER USER AND LMS VIEW GRADE....40
V. CONCLUSIONS AND FUTURE RESEARCH....41
A. CONCLUSION....41
B. FUTURE RESEARCH....42
APPENDIX. XAPI/LRS STATEMENT IN JSON FORMAT....43
LIST OF REFERENCES....45
INITIAL DISTRIBUTION LIST....47

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Current user to NeL LMS interaction. 3
Figure 2. Example implementation of the xAPI LRS tracking decoupled from content delivery. 5
Figure 3. JSON array of car objects. 6
Figure 4. xAPI/LRS statement structure (Advanced Distributed Learning, 2014). 7
Figure 5. Prototype system use case diagram. 10
Figure 6. Prototype system component diagram. 14
Figure 7. Client server three-tiered architecture of prototype system. 16
Figure 8. Student quiz submission sequence diagram. 17
Figure 9. Data Sync of LRS to LMS sequence diagram. 18
Figure 10. Teacher viewing student's quiz score sequence diagram. 19
Figure 11. Quiz website class diagram. 20
Figure 12. Data Sync website class diagram. 21
Figure 13. xAPI/LRS statement Extension element for quiz data tracking. 22
Figure 14. Prototype system implementation diagram. 26
Figure 15. Rustici Software's Basic Run-Time Calls SCORM package quiz (Rustici Software). 28
Figure 16. Quiz website quiz questions 29
Figure 17. Quiz website calculating Student score. 30
Figure 18. Quiz website building and inserting LRS statement. 31
Figure 19. Quiz website testing form for inserting data into LRS. 32
Figure 20. JSON statement in the LRS. 33
Figure 21. Data Sync website. 34
Figure 22. Data Sync function pulling data from LRS. 35
Figure 23. Data Sync function pulling data from LRS detail. 35
Figure 24. Data Sync function extracting, transforming, and setting default values. 36
Figure 25. Data Sync function SCO Moodle LMS java objects populated. 37
Figure 26. Data Sync function SCO inserted into Moodle database call. 38
Figure 27. Data Sync function Moodle LMS grade java objects populated. 39
Figure 28. Data Sync function Moodle grade history java objects populated. 39
Figure 29. Teacher viewing the Student's quiz grade in Moodle LMS before Data Sync function run. 40
Figure 30. Teacher viewing the Student's quiz grade in Moodle LMS after Data Sync function run. 40

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. xAPI/LRS statement elements description (Advanced Distributed Learning, 2014). 8
Table 2. Use Case 1 – Student – Access Quiz Questions. 11
Table 3. Use Case 2 – Student – Submit Quiz Answers. 11
Table 4. Use Case 3 – Administrator – Access Data Sync. 12
Table 5. Use Case 4 – Administrator – Start Data Sync. 12
Table 6. Use Case 5 – Teacher – Access Quiz Scores. 13
Table 7. Prototype system component descriptions and technologies. 15
Table 8. xAPI/LRS statement Extension elements descriptions. 23
Table 9. The devices used to access the prototype system. 27

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ADL	Advanced Distributed Learning
DOD	Department of Defense
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifier
JDBC	Java Database Connection
JSON	JavaScript Object Notation
LMS	Learning Management System
LRS	Learning Record Store
NeL	Navy eLearning
OS	Operating System
SCO	Shareable Content Object
SCORM	Shareable Content Object Reference Model
UC	Use Case
URL	Uniform Resource Locator
VPN	Virtual Private Network
xAPI	Experience Application Public Interface

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my entire family for their support and encouragement during my graduate studies. Without them, this endeavor would not have been possible.

Thanks to my advisors Professor Man-Tak Shing and Professor Arijit Das for obtaining a research grant and sponsor for this thesis, for help with scoping the thesis, for their feedback in every phase, and for general guidance.

Thanks to Virgil Hart and the folks from NETC for sponsoring the research for this thesis.

Thanks to Jason Haag and Andy Johnson from ADL for lending their expertise on SCORM and xAPI, coming out to meet with us at NPS, and discussing various directions the current and future research could take.

Thanks to Louis Algaze for his support and advice on the Sakai LMS that helped shape the direction of the thesis.

Thanks to Erik Lowney for his extensive networking knowledge and setting up the virtual environment that the prototype system resides in.

Thanks to my supervisors Ron Forbes and Michelle Rudolph at DMDC for their interest in my career progression, their approval to get a master's at NPS, and allowing me flexible work times for courses and thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The current Learning Management System (LMS) designs used for eLearning content delivery and data tracking are very centralized and high in complexity. This makes it time consuming and expensive to evolve them so that they can take advantage of emerging technologies.

Navy eLearning (NeL) uses a LMS called AtlasPro that does not have the capability to work from mobile device browsers and is limited in the way in tracking learning data. The Hypertext Markup Language (HTML) content that AtlasPro produces contains the HTML iframe tag, which is not supported by many mobile device browsers, causing the content to not behave and/or display correctly (Mobify, 2012). Without support for mobile devices, users of the AtlasPro system are limited to using desktop computer browsers. Not all learning takes place at a desktop computer, nor on a mobile device for that matter, but enabling the system to support mobile devices will broaden the scope of what learning can take place and what data about learning can be tracked.

There will always be a need of leveraging legacy software and data structures (legacy system) with modern software and data structures (modern system). Generally this need comes from trying to fulfill new requirements while controlling costs. Creating a modern system that leverages the legacy system can be more cost effective than a) creating a modern system that fulfils all of the legacy systems requirements and new requirements or b) modifying the legacy system to include the new requirements (Hyland Software, 2009).

A. THESIS OBJECTIVE

The technology to support the next generation of the NeL LMS, and LMSs in general, has not been solidified. This thesis provides a proof of concept study to identify the capabilities and issues of the Experience API (xAPI) as one potential technology. Specifically this thesis will look at how the xAPI, with its Learning Record Store (LRS), can help overcome the limitations of data tracking within a LMS. This is done using a

scenario based analysis to design a high level software model along with an implementation of a prototype system.

B. THESIS OUTLINE

Chapter II presents several topics to help better understand the problem, the work done, and the results for this thesis. The topics included are: the NeL, a general overview of LMSs, the Shareable Content Object Reference Model (SCORM) standard maintained by Advanced Distributed Learning (ADL), and details on the xAPI along with its use of the LRS software and JavaScript Object Notation (JSON) content format.

The prototype's design is discussed in Chapter III. It covers who the stakeholders of the system are, what use cases are needed to support the users, the various components that make up the system, how the components connect to one another, detailed sequence diagrams, and the system's class diagrams.

In Chapter IV, the prototype implementation is demonstrated to verify the prototype design. It discusses the environment that the prototype system is hosted in and a step by step demonstration with screen shots and details.

Lastly, Chapter V summarizes the research and suggests follow-on research that should be done but is outside the scope of this thesis.

II. BACKGROUND

The NeL is a system that "delivers computer-based learning designed to enhance professional and personal growth of Navy military members" (Navy eLearning, 2010). A core piece of NeL, as shown in Figure 1, is the LMS AtlasPro (Sea Warrior Program and Naval Education and Training Command Public Affairs, 2013).

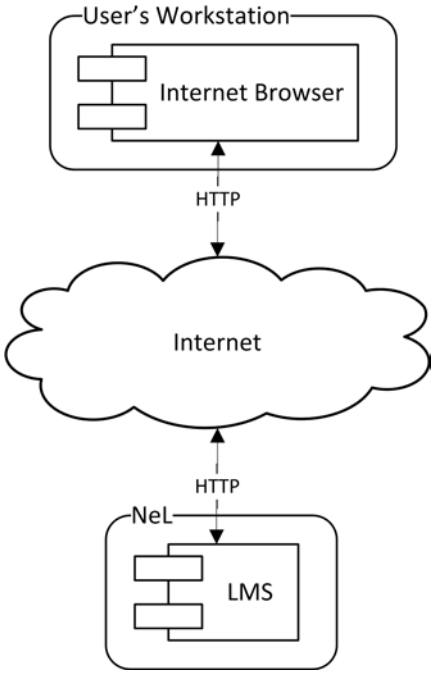


Figure 1. Current user to NeL LMS interaction.

A LMS is a software framework that consolidates every piece of the learning process into one system (Szabo & Flesher, 2002). Generally, a LMS will provide the following: content structure, security, user registration, content delivery, interaction/ navigation, assessment, tracking, reporting, record keeping, facilitating reuse, personalization, integration, and administration (Berking & Gallagher, 2013). Some examples of LMSs are AtlasPro, Sakai, Blackboard, and Moodle.

Typically a LMS supports content that implements the SCORM standard that is maintained by ADL. SCORM "defines the interrelationship of course components, data

models, and protocols so that learning content objects are sharable across systems that conform with the same model" (Berking & Gallagher, 2013, p. 34). The benefit of content built using SCORM is that it is not dependant on any one LMS and can be reused as a separate module.

There are several versions of the SCORM standard. This thesis focuses on SCORM 2004 3rd Edition, which has three major sub-standards: Content Aggregation Model, Run-Time Environment, and Sequencing and Navigation (Rustici, 2009). The Content Aggregation Model specifies how the content has to be structured and packaged and is read in by a LMS. The Run-Time Environment specifies "how [the] content should behave once it has been launched by the LMS." The Sequencing and Navigation specifies the users' movement through the content. With these three pieces it is possible to create a SCORM package as simple as a one page quiz or as complex as a war game simulation.

The lowest level of a piece of SCORM content is called an Asset (Rustici, 2009). Some examples of Assets are: text, images, video, and sound. An Asset is always a static piece of content. One or more Assets can be grouped into a Shareable Content Object (SCO) that "should represent the smallest unit of learning that the LMS should track." A SCO can communicate with the LMS but an Asset on its own cannot.

One major obstacle with NeL's current LMS implementation is how users have to interact with the LMS. Tracking of learning can only be done while a user is on a desktop computer using an Internet browser to connect to the LMS software (Poltrack, Haag, Hruska, & Johnson, 2012, p. 4). However, not all learning takes place within an Internet browser on a desktop computer.

ADL maintains the xAPI standard that is also known as Tin Can API and SCORM 2.0 (Advanced Distributed Learning). The main driving factor for the creation of xAPI is that SCORM is only able to track learning within a LMS but not all learning takes place within a LMS (Tin Can API). The xAPI is the "next generation of SCORM that allows e-learning to use modern technologies in an interoperable way" (Whitaker, 2012). It has been "designed to support existing SCORM use cases as well as enabling use cases that were difficult to meet with SCORM, such as mobile training and content

that is accessed outside of a web browser" (Advanced Distributed Learning, 2013). The xAPI decouples the tracking of learning information from the content delivery as shown in Figure 2.

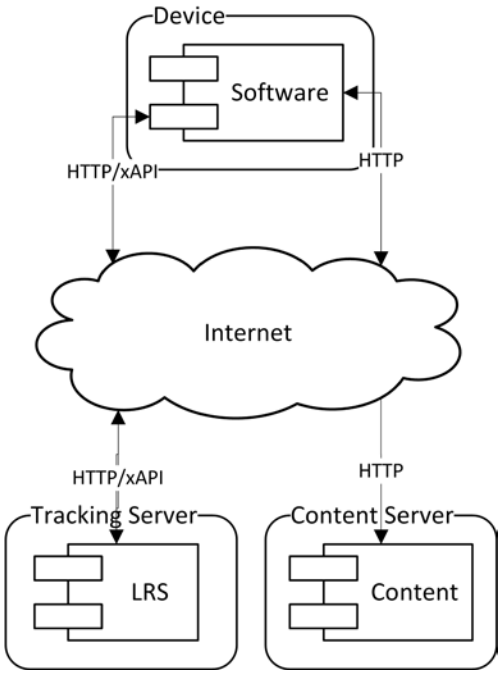


Figure 2. Example implementation of the xAPI LRS tracking decoupled from content delivery.

Since the xAPI is a standard, like SCORM, it can be implemented by any piece of software as a standalone system or as part of a larger system. In the next few years LMSs will probably start to incorporate xAPI into them similar to how they did with SCORM. Any piece of software implementing the xAPI standard running on any networked device can track learning activity and store that data inside of a LRS (Brusino, 2012).

A LRS is a key piece of the xAPI standard. It is the system that stores the learning activity information, referred to as a statement. However, it does not host or provide the content to the user (Experience API, 2014). Depending on the configuration of the LMS, the LRS could either be an internal or external component of the LMS (Brusino, 2012). The communication to and from the LRS is done over a network using the Hypertext Transfer Protocol (HTTP) with the content being JSON (Experience API, 2014).

JSON is a format for text that is easy to read by humans and able to be parsed by computers (Bray, 2014). See Figure 3 for example JSON of the car make, model, and year data. There are two structures that can be used together in JSON: object and array. An object contains one or more keys, which is a string of text (surrounded by double quotes), and each key relates to a value. An array is a list of values. The value for both the object key(s) and arrays can be one of the following: string of text (surrounded by double quotes), a number, another object, another array, a boolean (true or false), or a null (empty).

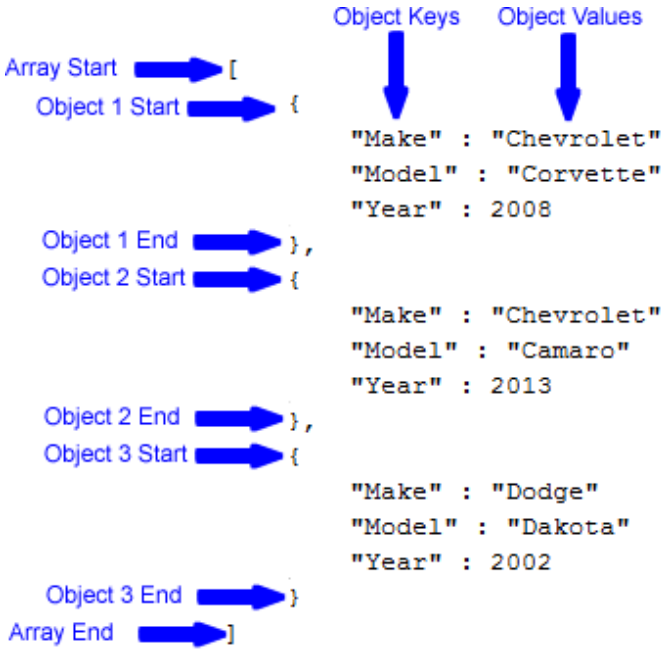


Figure 3. JSON array of car objects.

At a high level, a single statement stored in the LRS contains several different pieces of information (Experience API, 2014). For the purpose of this thesis, there are three top level data elements about the learning (Actor, Verb, and Object) and five top level data elements about the statement itself (Authority, Timestamp, Stored, Version, and ID). See Figure 4 for the xAPI/LRS statement structure, Table 1 for a description of each element, and the appendix for a sample JSON statement. (Note: This does not cover the full xAPI/LRS elements but only a sub-set of elements as relevant to this thesis. The

full list and explanation of xAPI/LRS elements can be found here: https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md.)

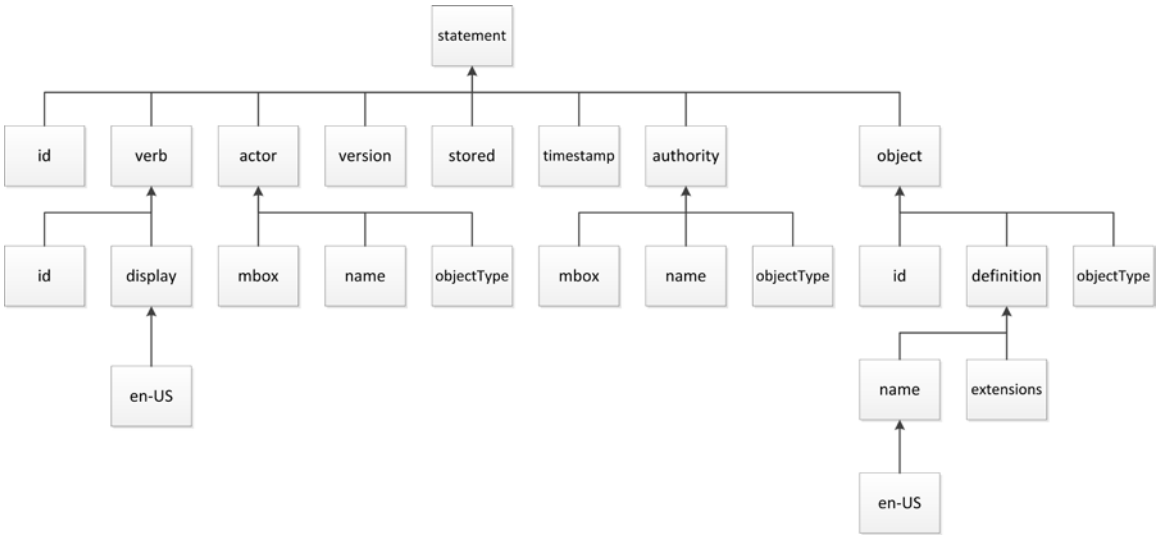


Figure 4. xAPI/LRS statement structure (Advanced Distributed Learning, 2014).

Element Name	Element Description
statement	The xAPI data is stored in this.
statement.id	A unique ID for this statement in the LRS.
statement.verb	The action being done by the Actor.
statement.verb.id	The full ID of the verb.
statement.verb.display	The short ID for displaying purposes.
statement.verb.display.en-US	The U.S. English short ID for displaying purposes.
statement.actor	The user doing the learning.
statement.actor.mbox	The email of the user.
statement.actor.name	The name of the user.
statement.actor.objectType	The type of object the user is.
statement.version	The xAPI version used for this statement.
statement.stored	The day and time this statement was stored in the LRS.
statement.timestamp	The day and time the learning took place.
statement.authority	The account that stored this statement in the LRS. This is generally the account of the system interacting with the LRS. It does not have to be the Actor's account although it could be.
statement.authority.mbox	The email of the authority account.
statement.authority.name	The name of the authority account.
statement.authority.objectType	The type of object the authority account is.
statement.object	The learning that took place by the Actor.
statement.object.id	The ID of the learning being tracked.
statement.object.definition	Additional data about the object.
statement.object.definition.name	The name of the additional object data.
statement.object.definition.name.en-US	The U.S. English name for displaying purposes.
statement.object.definition.extensions	Allows adding customized data elements for the project. This is a key feature used in this thesis and is covered in more detail in the XAPI/LRS Extensions section.
statement.object.objectType	The type of object the object is.

Table 1. xAPI/LRS statement elements description (Advanced Distributed Learning, 2014).

III. PROTOTYPE SYSTEM DESIGN

The goal of this prototype system design is to provide a high level concept for getting data elements that are stored in a LRS into a LMS. It does not cover the security aspects required for a production Department of Defense (DOD) system. For this study we chose to show this proof of concept using quiz information. The data elements of interest to move from the LRS to the LMS are: a few of the SCORM SCO data tracking elements, the student ID who took the quiz, and the course ID the quiz is in.

A. STAKEHOLDERS

There are three groups of stakeholders for the prototype system: Students, Administrators, and Teachers. In a future prototype the Administrator role, as it relates to the prototype in this thesis, could be replaced with automation/software.

- Student: Needs to be able to access quiz questions and submit quiz answers.
- Administrator: Needs to be able to initiate the copy of data from the LRS to the LMS.
- Teacher: Needs to be able to access quiz scores for Students in the LMS.

B. USE CASES

The use cases in Figure 5 are based on the stakeholders and their needs. The details of each use case (UC) can be found in Table 2 through Table 6. As this is a proof of concept prototype the exception/error use cases are not handled.

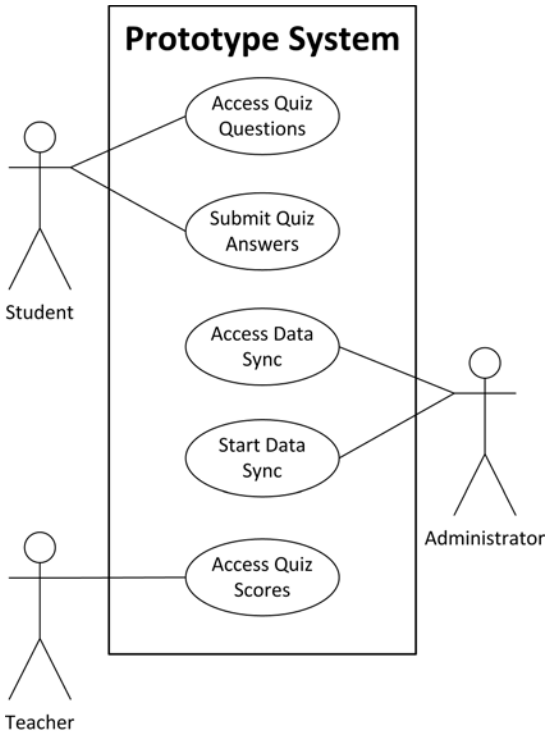


Figure 5. Prototype system use case diagram.

There are a few assumptions that apply to all the use cases, UC1-UC5. First is that the LMS has been set up with a course and that the course has been set up with a SCORM quiz content package. Second is that the LMS has been set up with the Student user and Teacher user and they have been enrolled in the course as a student roll and teacher roll, respectfully. Please note that LMS's behavior (code base) is not being modified in anyway so UC5 would be the same on any LMS. The UC1-UC4 are the new pieces for the prototype system.

Decoupling the data tracking and content hosting, that a LMS typically provides, allows for greater system flexibility. The decoupling can be done by having the content hosted by a server and the data tracked in the LRS. The website in this prototype is the Quiz website since this prototype focuses on quiz score data. This is shown in UC1 and UC2.

UC1	Student – Access Quiz Questions
Description	Steps taken for a Student to bring up the Quiz website that contains the quiz questions.
Assumptions	• Student has not attempted/completed the quiz. • Student is using either a mobile device or workstation (device) that has an Internet browser installed on it.
Base Course of Action	1. Student opens the Internet browser on their device. 2. Device displays Internet browser. 3. Student inputs the Quiz website URL into the Internet browser. 4. Device displays Quiz website content (quiz questions) via Internet browser.

Table 2. Use Case 1 – Student – Access Quiz Questions.

UC2	Student – Submit Quiz Answers
Description	Steps taken for a Student to submit the quiz answers on the Quiz website.
Assumptions	• Student has just completed UC1. • Student is using either a mobile device or workstation (device) that has an Internet browser installed on it.
Base Course of Action	1. Student selects answers to each quiz question displayed in Internet browser. 2. Device displays selected answers for each quiz question via Internet browser. 3. Student submits quiz answers in Internet browser. 4. Device displays submission success via Internet browser.

Table 3. Use Case 2 – Student – Submit Quiz Answers.

UC3	Administrator – Access Data Sync
Description	Steps taken for an Administrator to bring up the Data Sync website that contains the Data Sync functions.
Assumptions	• Student has just completed UC2. • Data Sync has not been run. • Administrator is using either a mobile device or workstation (device) that has an Internet browser installed on it.
Base Course of Action	1. Administrator opens the Internet browser on their device. 2. Device displays Internet browser. 3. Administrator inputs the Data Sync website URL into the Internet browser. 4. Device displays Data Sync website content (Data Sync functions) via Internet browser.

Table 4. Use Case 3 – Administrator – Access Data Sync.

UC4	Administrator – Start Data Sync
Description	Steps taken for an Administrator to start the Data Sync function from the LRS to the LMS on the Data Sync website.
Assumptions	• Administrator has just completed UC3. • Administrator is using either a mobile device or workstation (device) that has an Internet browser installed on it.
Base Course of Action	1. Administrator selects the Start Data Sync button in Internet browser. 2. Device displays Data Sync ran successfully in Internet browser.

Table 5. Use Case 4 – Administrator – Start Data Sync.

UC5	Teacher – Access Quiz Scores
Description	Steps taken for a Teacher to view the Student's quiz score data in the LMS.
Assumptions	• Administrator has just completed UC4. • Teacher is using a workstation that has an Internet browser installed on it.
Base Course of Action	1. Teacher opens the Internet browser on their workstation. 2. Workstation displays Internet browser. 3. Teacher inputs the LMS URL into the Internet browser. 4. Workstation displays LMS login screen via Internet browser. 5. Teacher inputs their username and password into the LMS logins fields and clicks the Login button in the Internet browser. 6. Workstation displays login successful by showing the LMS home page via the Internet browser. 7. Teacher selects the course they are enrolled in as the teacher role in the Internet browser. 8. Workstation displays the course home page via the Internet browser. 9. Teacher selects the grade book for the course in the Internet browser. 10. Workstation displays the grades for all students enrolled in the course via the Internet browser.

Table 6. Use Case 5 – Teacher – Access Quiz Scores.

C. COMPONENTS

Figure 6 shows all of the prototype system components, the users of the system, and how they relate to one another. This study is conducted within NPS's firewall in a virtualized environment.

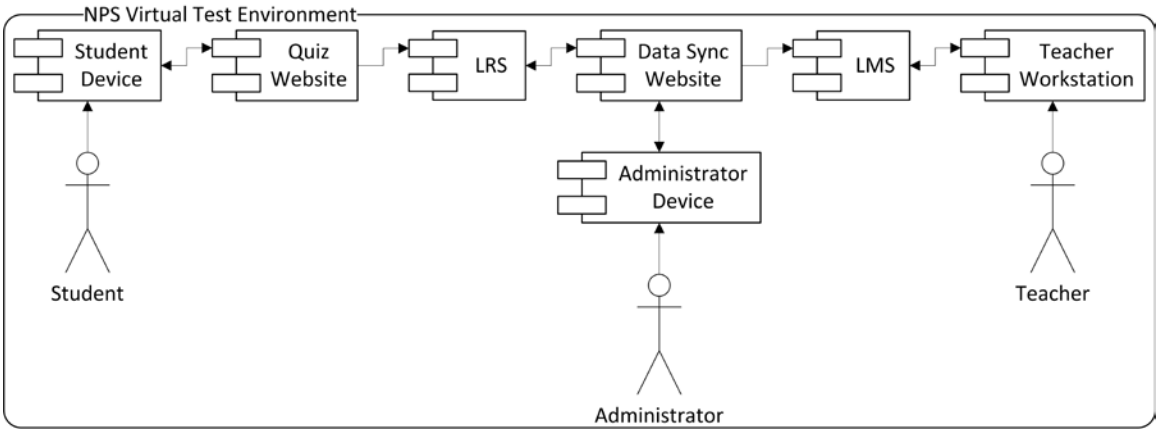


Figure 6. Prototype system component diagram.

In Table 7 the description and technology for each component is listed. The Quiz website, LRS, Data Sync website, and LMS can all reside on separate virtualized servers to show that they are separate components. However, they could all just as easily be put on a single server.

Component	Description	Technology
Student Device	Student uses this to access the Quiz website.	This can be any device that has an Internet browser and network capability such as a desktop computer or smartphone.
Quiz website	Hosts the quiz that includes the content such as: HTML, CSS, JavaScript, etc. When the quiz data is submitted, the tracking data is sent to the LRS.	Server OS: Ubuntu 12.04.4 (64 bit) website Language: Java 6.31 Hosting Software: Apache Tomcat 7.0.26
LRS	Tracks the resulting quiz data.	Server OS: Ubuntu 12.04.4 (64 bit) LRS Software: ADL LRS 1.0.0
Administrator Device	Administrator uses this to access the Data Sync website.	This can be any device that has an Internet browser and network capability such as a desktop computer or smartphone.
Data Sync website	Hosts the Data Sync function that pulls the quiz data from the LRS, transforms it to match the LMS data format, and then pushes the transformed data into the LMS.	Server OS: Ubuntu 12.04.4 (64 bit) website Language: Java 6.31 Hosting Software: Apache Tomcat 7.0.26
LMS	End point for the Student's quiz score data and where the Teacher can view the Student's quiz score.	Server OS: Ubuntu 12.04.4 (64 bit) LMS Software: Moodle 2.7 Hosting Software: Apache 2.4.9
Teacher Workstation	Teacher uses this to access the LMS.	This can be any desktop computer that has an Internet browser and network capability.

Table 7. Prototype system component descriptions and technologies.

D. ARCHITECTURE

The prototype system uses a client server three-tiered architecture as shown in Figure 7. The tiers are presentation, logic, and data/storage.

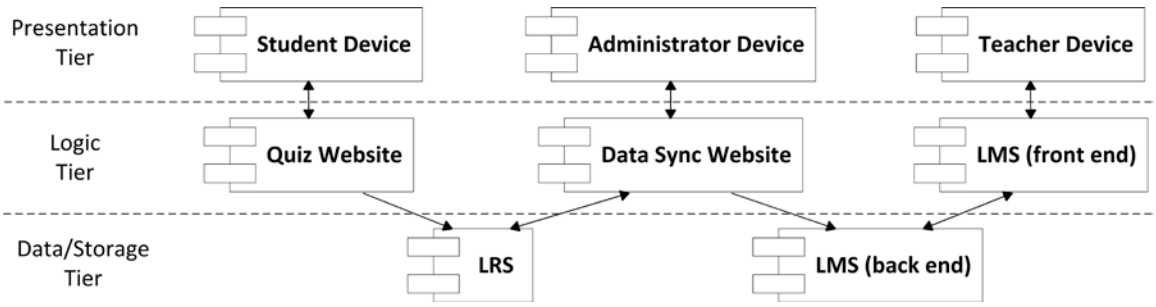


Figure 7. Client server three-tiered architecture of prototype system.

E. SEQUENCE DIAGRAMS

At a high level, there are three stages for the prototype system and each stage is triggered by a user. In the first stage, shown in Figure 8, is the Student user submits their quiz response to the Quiz website that in turn places the data inside the LRS. The second stage, shown in Figure 9, is the Administrator user runs the Data Sync program to copy the data from the LRS to the LMS. For the third stage, shown in Figure 10, the Teacher user access the LMS to view the Student's quiz grade.

1. Student User Quiz Submission

First the Student user using the Student Device's Internet browser accesses the HTML quiz content (HTTP get) on the Quiz website (java servlet). Next the Student Device, via the Student user, submits the quiz response (HTTP Post) to the Quiz website. The Quiz website calculates the quiz score and then constructs the JSON statement to be inserted into the LRS. This JSON statement includes the Extensions data elements that are covered in more detail in the XAPI/LRS Extensions section. Lastly the Quiz website pushes the JSON statement into the LRS (HTTP post). The LRS adds a few additional elements to the statement once it is stored (see the appendix for a sample).

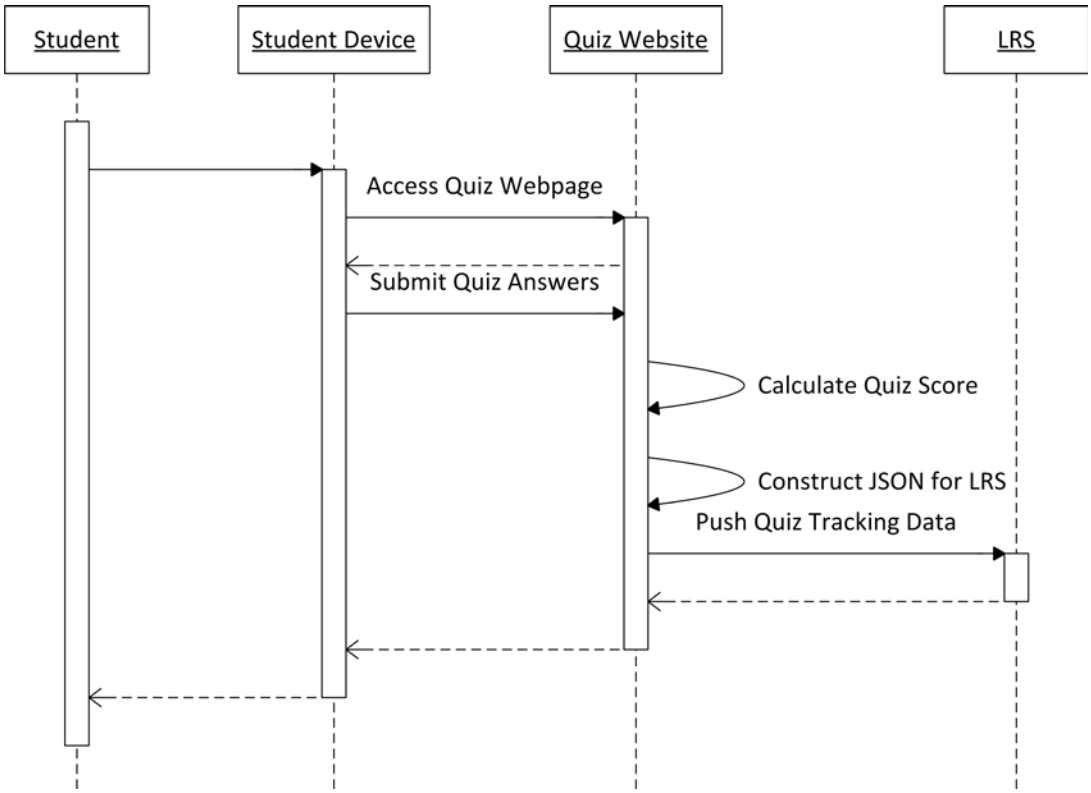


Figure 8. Student quiz submission sequence diagram.

2. Administrator User Data Sync

First the Administrator user using the Administrators Device's Internet browser accesses (HTTP get) the Data Sync website (java servlet). Next the Administrator Device, via the Administrator user, submits the request (HTTP post) to start the Data Sync process. The Data Sync website then pulls the quiz score and other tracking data (HTTP get) for the Student user as a JSON statement from the LRS (see the appendix for a sample), transforms the data into the LMS, and pushes the quiz tracking data (JDBC inserts) into the LMS.

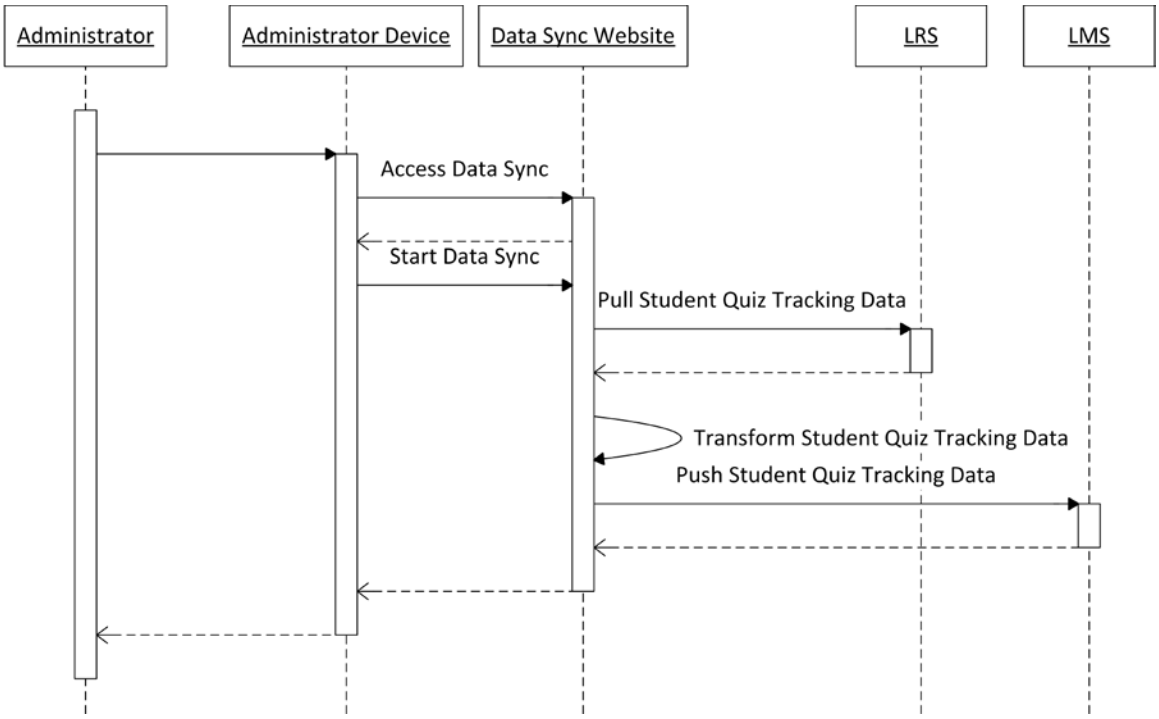


Figure 9. Data Sync of LRS to LMS sequence diagram.

3. Teacher User View Grade

The Teacher user using the Teacher Workstation's Internet browser logs into the LMS, selects the course, and opens the grade book to view the Student's quiz grade.

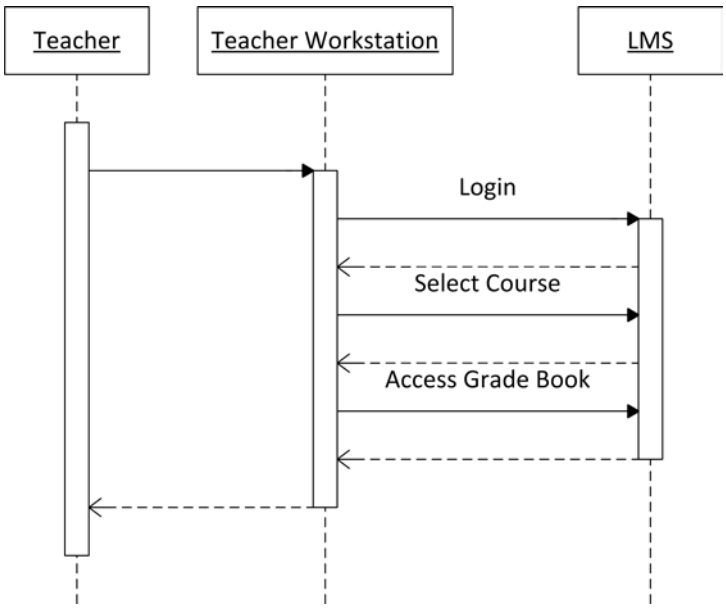


Figure 10. Teacher viewing student's quiz score sequence diagram.

F. CLASS DIAGRAMS

There are four separate server side software components that make up the prototype system: Quiz website, LRS, Data Sync website, and LMS. Of those, the LRS and LMS software existed before this thesis. The Quiz website and Data Sync website were built for this thesis and their class diagrams are shown in Figure 11 and Figure 12, respectfully.

Each HTTP request made to the Quiz website has a specific action name associated with it and goes through the MainFilter and then into the HomeController. When the HTTP request is just for the list of quiz questions, the HomeController directs the Student to the quiz question page. When the HTTP request is submitting the quiz answers, the HomeController calculates the quiz score, builds the InsertFormBean object, and then uses the LrsService to put the data into the LRS.

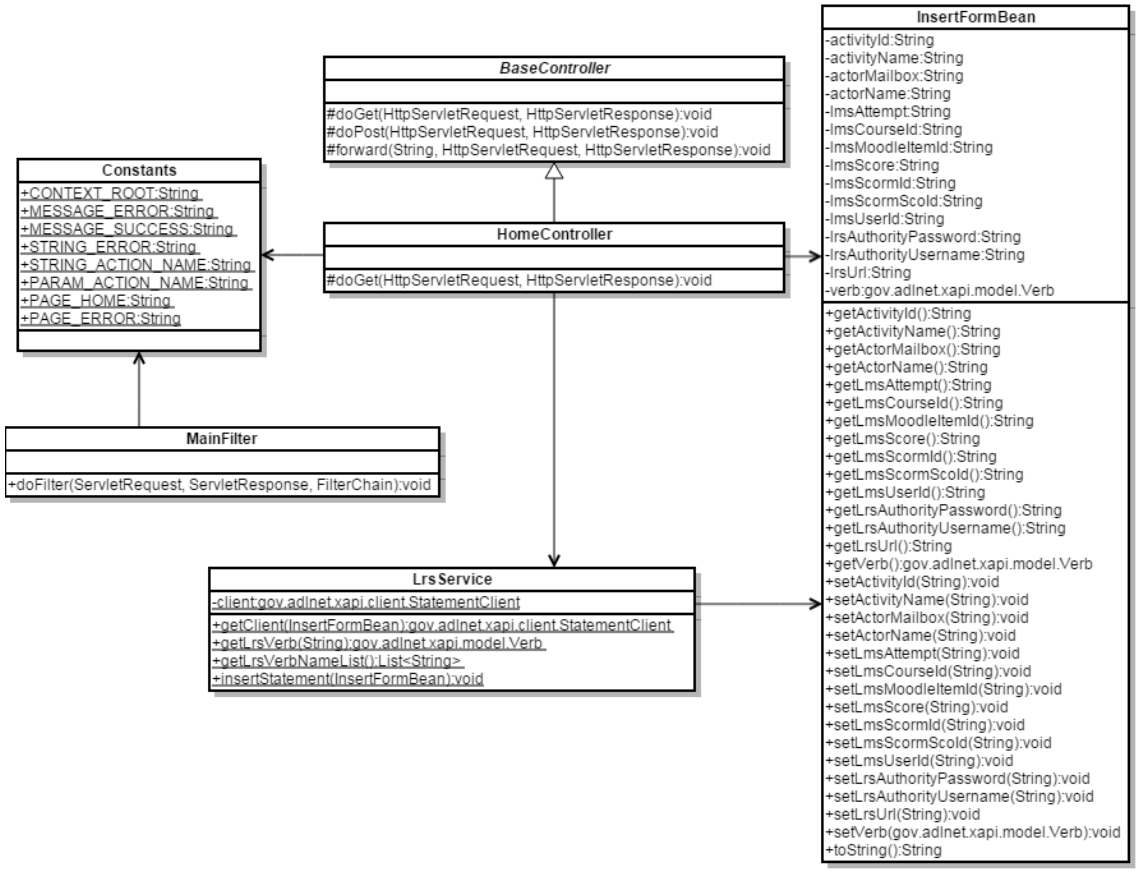


Figure 11. Quiz website class diagram.

The Data Sync website is designed with the ability to interact with both the LRS and LMS at various stages of the process to make development easier. However, since the goal of the Data Sync website is to sync the data from the LRS to the LMS, that is the process discussed here. Each HTTP request made to it has a specific action name associated with it and goes through the MainFilter. The first request to access the Data Sync website goes to the HomeController and redirects the Administrator to the main Data Sync website page that contains all the functions. The Administrator executes the Data Sync function from this page that goes to the DataSyncController that calls the DataSyncService. The DataSyncService calls the LrsService to pull the Student's quiz tracking data from the LRS, it transforms the data into the LMS format, and then calls the LmsService to insert the data into the LMS.

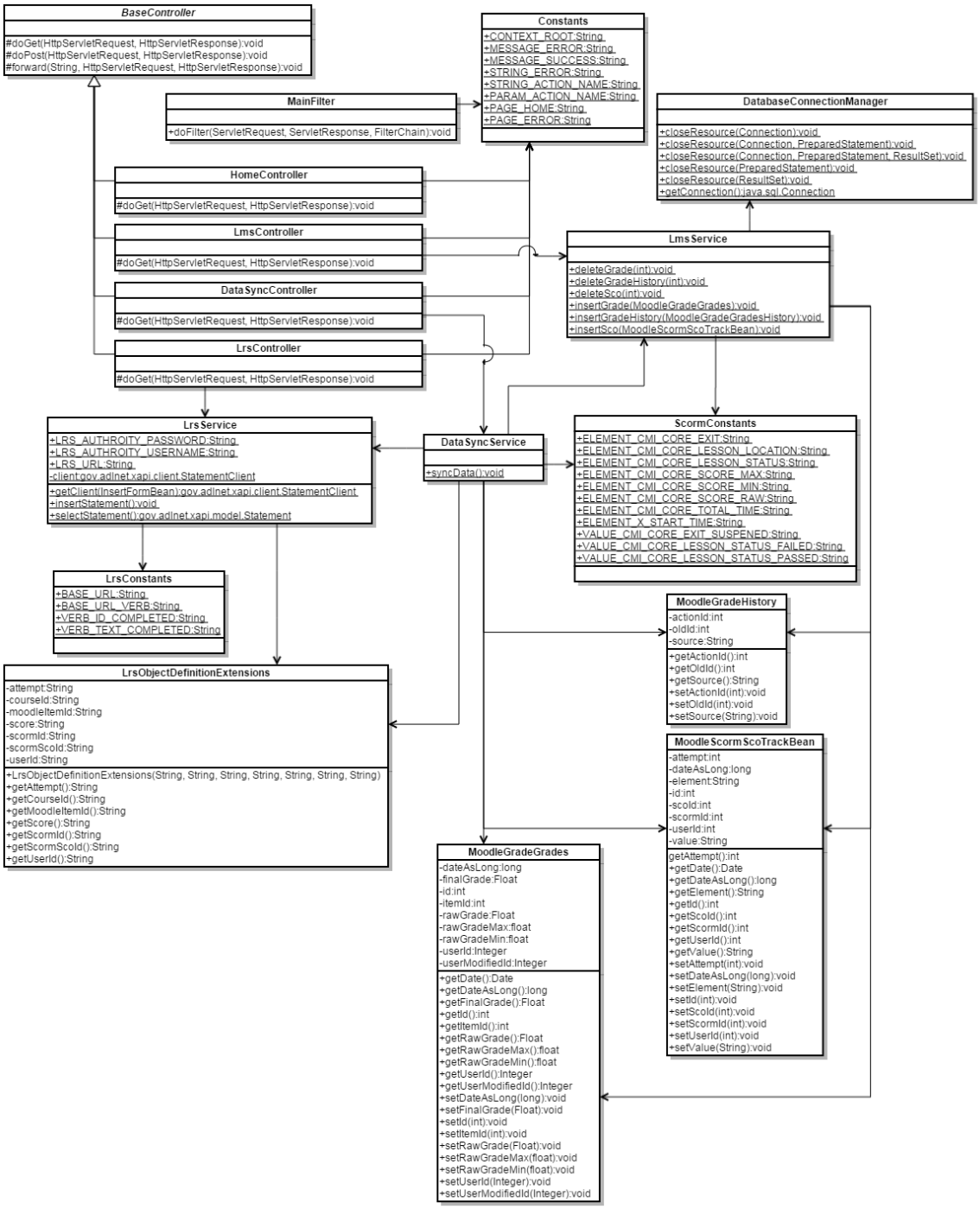


Figure 12. Data Sync website class diagram.

G. XAPI/LRS EXTENSIONS

The structure of a statement as used in this thesis can be found in Figure 4 in Chapter II – Background section. A feature to note about the statement.object.definition.extensions element is the ability to add customized data sub-elements within it known as Extensions. The Extension data element is critical to this thesis as it allows the statement to contain the quiz tracking data elements. These are the elements that are copied from the LRS into the LMS by the Data Sync function. See Figure 13 for a view of just the Extension data element structure, Table 8 for an explanation of each Extension data element that is being added for use in the prototype system, and the appendix for a sample full LRS JSON statement.

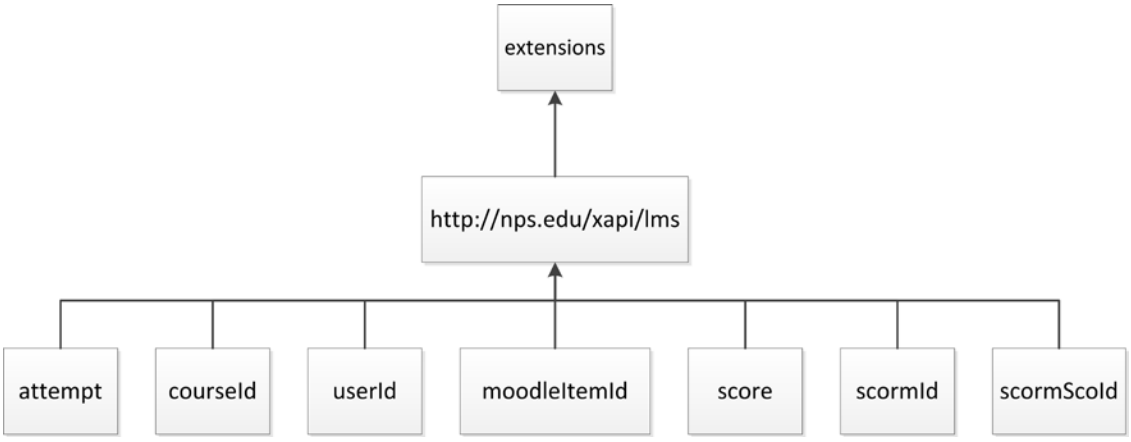


Figure 13. xAPI/LRS statement Extension element for quiz data tracking.

Element Name	Element Description
attempt	The quiz attempt this is for. Attempt 1 would be the first time the Student has taken the quiz.
courseId	The LMS course ID that the quiz is in.
userId	The LMS user ID of the Student that took the quiz.
moodleItemId	The LMS ID for the item in the Moodle LMS. An item could be anything from an assignment to a resource posting. In this case the ID is referring to the specific quiz in the LMS.
score	The score the Student received for the quiz.
scormId	The SCORM ID for the SCORM package that was uploaded into the LMS.
scormScoId	Within the SCORM package, the specific SCO ID this is for. In this case, it is the referring to the quiz SCO in the SCORM package.

Table 8. xAPI/LRS statement Extension elements descriptions.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PROTOTYPE SYSTEM IMPLEMENTATION

The prototype system is implemented in a virtual environment within the NPS firewall as shown in Figure 14. Table 9 lists the various devices used for accessing the prototype system. There are two ways the prototype system is accessed. The first way is with a Local Workstation running VMware Horizon View Client software, which creates a virtual private network (VPN), to connect through the NPS firewall to a virtual Remote Workstation in the NPS Cloudlab environment. The Remote Workstation is then used to develop and access the virtual servers of the prototype system. The second way is with a Mobile Device configured to use NPS's VPN to connect through the NPS firewall that then connects to the virtual servers of the prototype system. The Mobile Device is for accessing the prototype system as the Student user while the Remote Workstation, via the Local Workstation, is for accessing the prototype system as the Student, Administrator, or Teacher user.

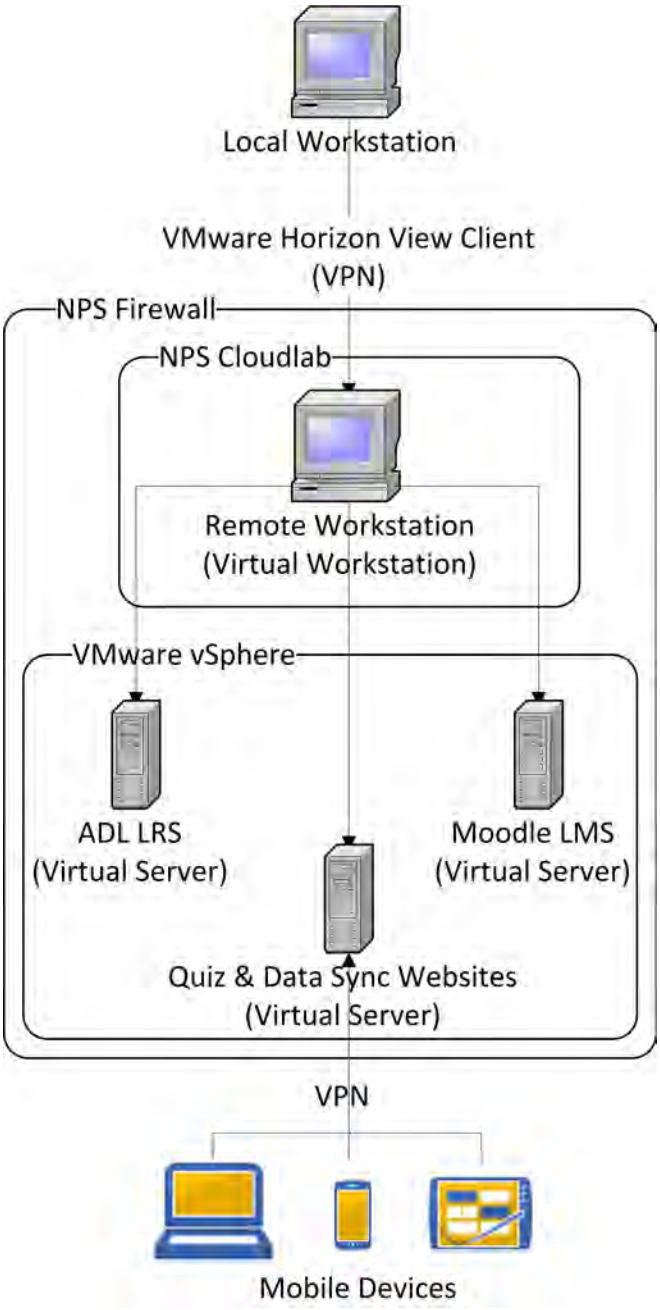


Figure 14. Prototype system implementation diagram.

Purpose	Type	Brand	Model	OS/Version
Local Workstation	Desktop	Dell	XPS 720	Windows 7
Mobile Device	Laptop	HP	Pavilion DV7	Windows 7
Mobile Device	Tablet	Google	Nexus 10	Android 4.4.4
Mobile Device	Smartphone	HTC	HTC One	Android 4.1.2

Table 9. The devices used to access the prototype system.

ADL's implementation of the xAPI standard, version 1.0.0, is used for the prototype system. The LRS built by ADL was installed as-is; no code modifications were made to the LRS for this prototype.

An alternative LMS was needed for this study because the AtlasPro LMS used by NeL could not be obtained within the time constraints of this thesis. Since the data being moved between the LRS and LMS is SCORM tracking data elements and since SCORM is the standard the AtlasPro LMS used for storing that data, the assumption is that any LMS that uses the SCORM standard to store the tracking data elements would be sufficient to use in place of the AtlasPro LMS. The LMS used for this study is Moodle version 2.7 and no code modifications were made to it.

The SCORM tracking data elements used in this thesis are for a very simple quiz of that the score data element is the most important. The sample SCORM package used for this thesis is the Basic Run-Time Calls, SCORM 2004 3rd Edition, from Rustici Software's Scorm.com website: http://scorm.com/wp-content/assets/golf_examples/PIFS/RuntimeBasicCalls_SCORM20043rdEdition.zip. At the very end of this SCORM package is a quiz. Figure 15 shows part of that quiz. The correct answers are in bold as this is a sample quiz. These same quiz questions and answers are used in the prototype system.

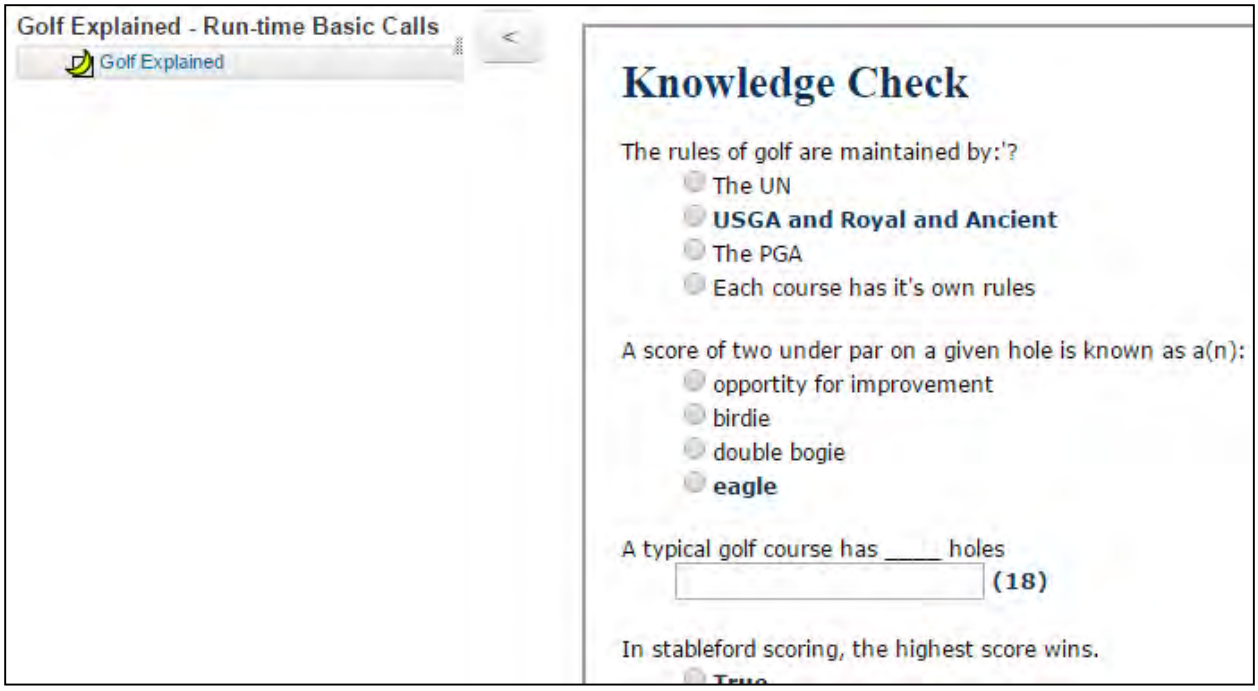


Figure 15. Rustici Software's Basic Run-Time Calls SCORM package quiz (Rustici Software).

A. STUDENT USER AND LRS DATA INSERT

The Quiz website that the Student accesses is shown Figure 16 with the screen shots side-by-side to condense the image. The correct answers are indicated only because it is a prototype. Once the Student is done filling in the answers they click the Submit Quiz button.

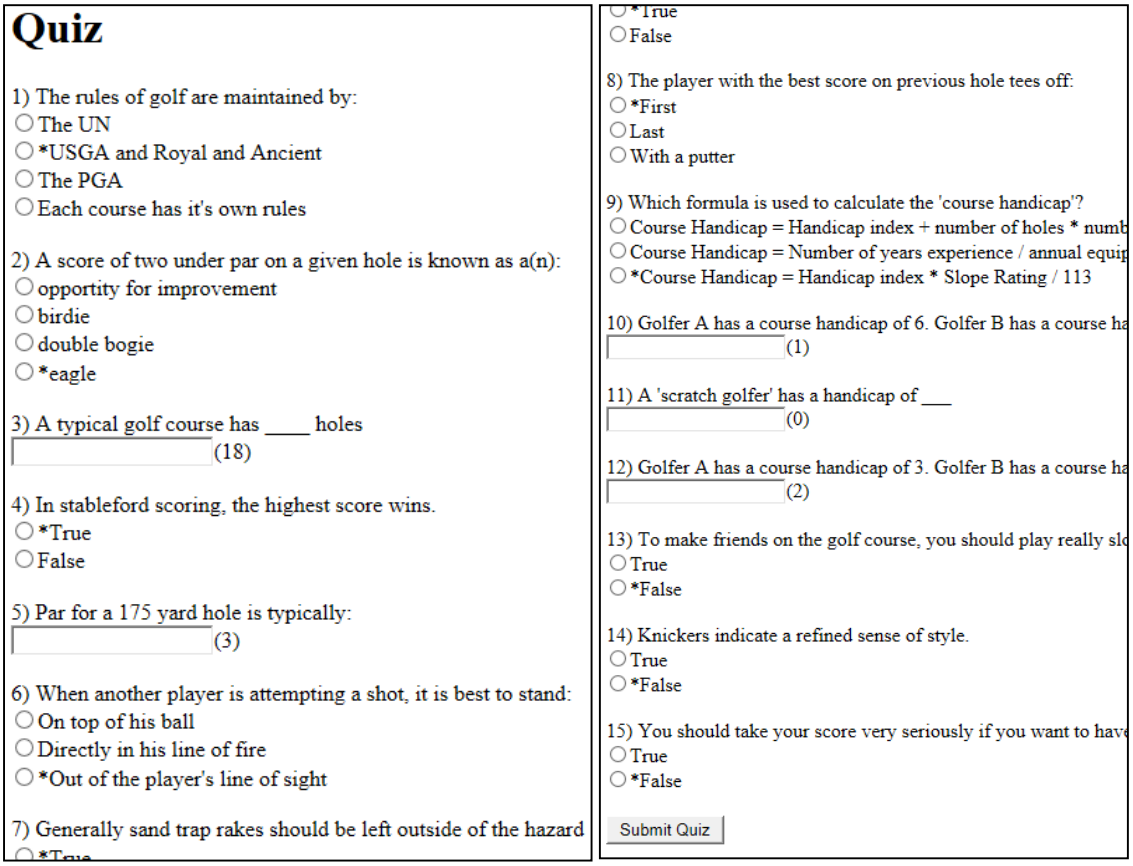


Figure 16. Quiz website quiz questions.

The Quiz website calculates the Student's score, builds the LRS statement object, and then inserts the statement into the LRS. Figure 17 shows the Student's score being calculated and Figure 18 shows the LRS statement being assembled and inserted. The java library used to build the java statement object, create the JSON statement, and then insert the JSON statement into the LRS is the Experience API Java Library (jxapi) and can be found here: https://github.com/adlnet/jxapi. The Extension data elements added for this prototype are converted from a java object into JSON by Google's JSON java library (GSON) and can be found here: https://code.google.com/p/google-gson/. The JSON produce by the GSON library is then used with the jxapi library to produce the complete JSON statement that is inserted into the LRS.

```
//Submit Quiz data
else if ("submitQuiz".equals(actionName)) {
    //Pull form data
    Integer totalRight = 0;
    totalRight += getAnswerValue(request.getParameter("q1"));
    totalRight += getAnswerValue(request.getParameter("q2"));
    if ("18".equals(request.getParameter("q3"))) {
        totalRight++;
    }
    totalRight += getAnswerValue(request.getParameter("q4"));
    if ("3".equals(request.getParameter("q5"))) {
        totalRight++;
    }
    totalRight += getAnswerValue(request.getParameter("q6"));
    totalRight += getAnswerValue(request.getParameter("q7"));
    totalRight += getAnswerValue(request.getParameter("q8"));
    totalRight += getAnswerValue(request.getParameter("q9"));
    if ("1".equals(request.getParameter("q10"))) {
        totalRight++;
    }
    if ("0".equals(request.getParameter("q11"))) {
        totalRight++;
    }
    if ("2".equals(request.getParameter("q12"))) {
        totalRight++;
    }
    totalRight += getAnswerValue(request.getParameter("q13"));
    totalRight += getAnswerValue(request.getParameter("q14"));
    totalRight += getAnswerValue(request.getParameter("q15"));
    logger.debug("totalRight: " + totalRight);

    //Calculate quiz score
    Float scoreRaw = Float.valueOf(totalRight)/15f * 100;
    logger.debug("scoreRaw: " + scoreRaw);
    DecimalFormat df = new DecimalFormat("#");
    String scoreFinal = df.format(scoreRaw);
    logger.debug("scoreFinal: " + scoreFinal);

    //Build LRS data
    InsertFormBean bean = new InsertFormBean();
    bean.setLrsUrl("http://172.20.56.126:8000/xapi");
    bean.setLrsAuthorityUsername("Rett Miller");
    bean.setLrsAuthorityPassword("********");
    bean.setActorMailbox("mailto:navyschooluser@gmail.com");
    bean.setActorName("NavySchoolUser");
    bean.setActivityId("http://nps.edu/moodle/quiz2");
    bean.setActivityName("Quiz 2");
    bean.setVerb(LrsService.getLrsVerb("completed"));
    bean.setLmsUserId("4");
    bean.setLmsAttempt("1");
    bean.setLmsScore(scoreFinal);
    bean.setLmsCourseId("2");
    bean.setLmsMoodleItemId("2");
    bean.setLmsScormId("1");
    bean.setLmsScormScoId("2");
    logger.debug("bean: " + bean);

    //Insert LRS data
    LrsService.insertStatement(bean);
}
```

Figure 17. Quiz website calculating Student score.

```
public static void insertStatement(InsertFormBean bean) throws Exception {
    try {
    StatementClient client = getClient(bean);
    Statement statement = new Statement();
    Agent agent = new Agent();
    agent.setMbox(bean.getActorMailbox());
    agent.setName(bean.getActorName());
    statement.setActor(agent);
    statement.setId(UUID.randomUUID().toString());
    statement.setVerb(bean.getVerb());
    Activity a = new Activity();
    a.setId(bean.getActivityId());
    statement.setObject(a);
    ActivityDefinition ad = new ActivityDefinition();
    ad.setName(new HashMap<String, String>());
    ad.getName().put("en-US", bean.getActivityName());
    HashMap<String, JsonElement> map = new HashMap<String, JsonElement>();
    Gson gson = new Gson();
    JsonElement je = gson.toJsonTree(new LrsObjectDefinitionExtensions(
            bean.getLmsUserId(),
            bean.getLmsAttempt(),
            bean.getLmsScore(),
            bean.getLmsCourseId(),
            bean.getLmsMoodleItemId(),
            bean.getLmsScormId(),
            bean.getLmsScormScoId()
            ));
    map.put("http://nps.edu/xapi/lms", je);
    ad.setExtensions(map);
    a.setDefinition(ad);

    gson = new Gson();
    String statementJson = gson.toJson(statement);
    logger.debug("statementJson: " + statementJson);

    String publishedId = client.publishStatement(statement);
    logger.debug("publishedId: " + publishedId);
    }
    catch (ProtocolException e) {
        if (e.getMessage().contains("Server redirected too many")) {
            throw new Exception("Problem connecting to the LRS server. Check that the "
                    + "LRS URL (" + bean.getLrsUrl() + "), "
                    + "Authority Username (" + bean.getLrsAuthorityUsername() + "), "
                    + "and Authority Password (" + bean.getLrsAuthorityPassword() + ") "
                    + "are correct and work on the LRS server.", e);
        }
        else {
            throw e;
        }
    }
}
```

Figure 18. Quiz website building and inserting LRS statement.

In addition to the quiz on the Quiz website, for testing purposes a form was built that can insert various values into the LRS as shown in Figure 19. Other than the "LRS URL" field that just points to the LRS location and the "LRS Authority Password" that is used to connect to the LRS, all the data ends up in the statement in the LRS. The fields that are prefixed with "LRS" are required values for the LRS statement where as the fields prefixed with "LMS" are the Extension data elements added specifically for this prototype. The Extension data elements eventually end up in the LMS after the Data Sync function is run.

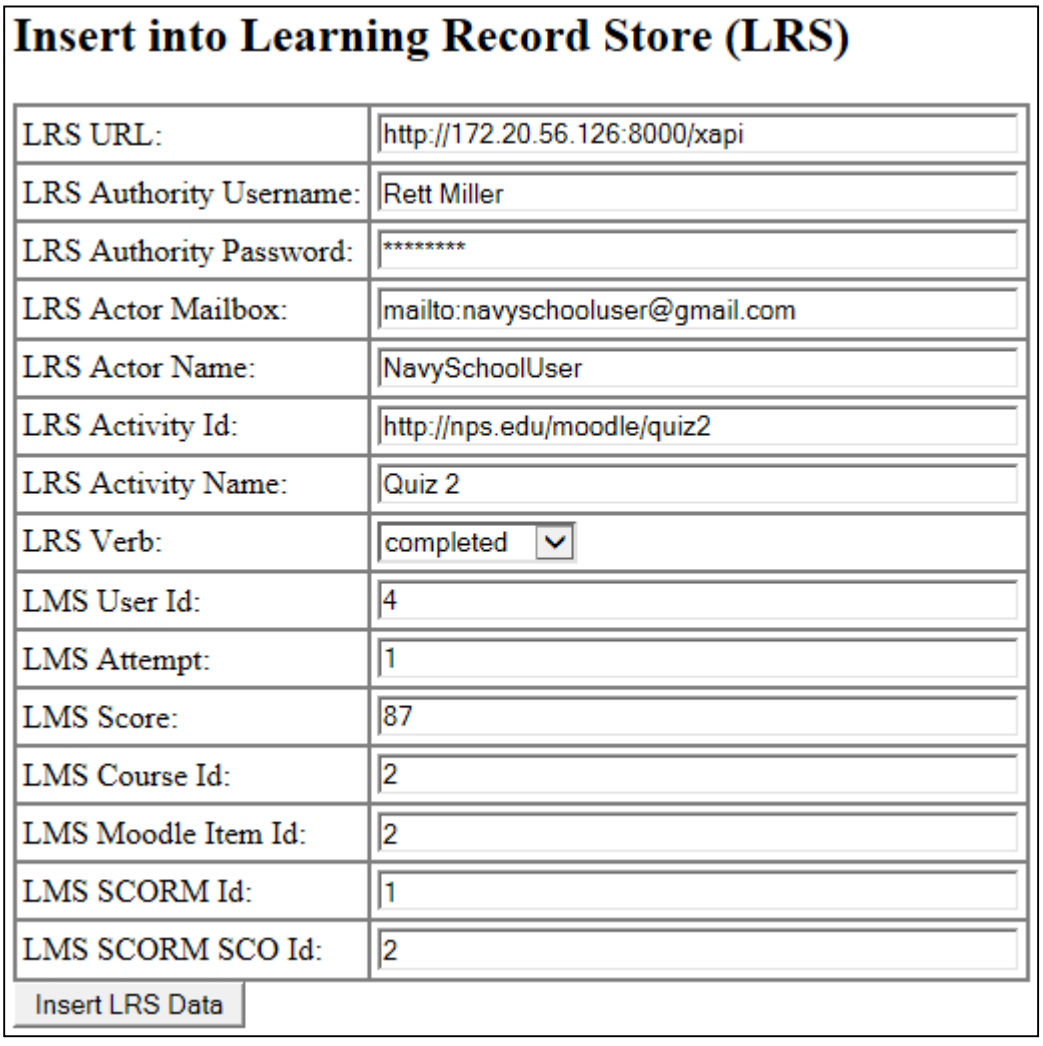


Figure 19. Quiz website testing form for inserting data into LRS.

Figure 20 shows the JSON statement once it is in the LRS. In addition to installing the ADL LRS, the ADL Experience API Client Examples were installed: https://github.com/adlnet/experienceapi_client_examples. This included a Report Sample project that can be used to view the statements in the LRS and is where the image in Figure 20 was taken from.

Figure 20. JSON statement in the LRS.

B. ADMINISTRATOR USER AND DATA SYNC FUNCTION

Once the Student's quiz data is in the LRS, the Administrator can access the Data Sync website as shown in Figure 21. To run the Data Sync function, the Administrator would click on the "Sync Data from LRS to LMS" button at the very bottom. For testing purposes, the Data Sync website also can test various aspects of both the LRS and LMS. One item to note is that the xAPI standard does not provide a way to delete statements from a LRS. The steps of the Data Sync function are broken out and discussed below.

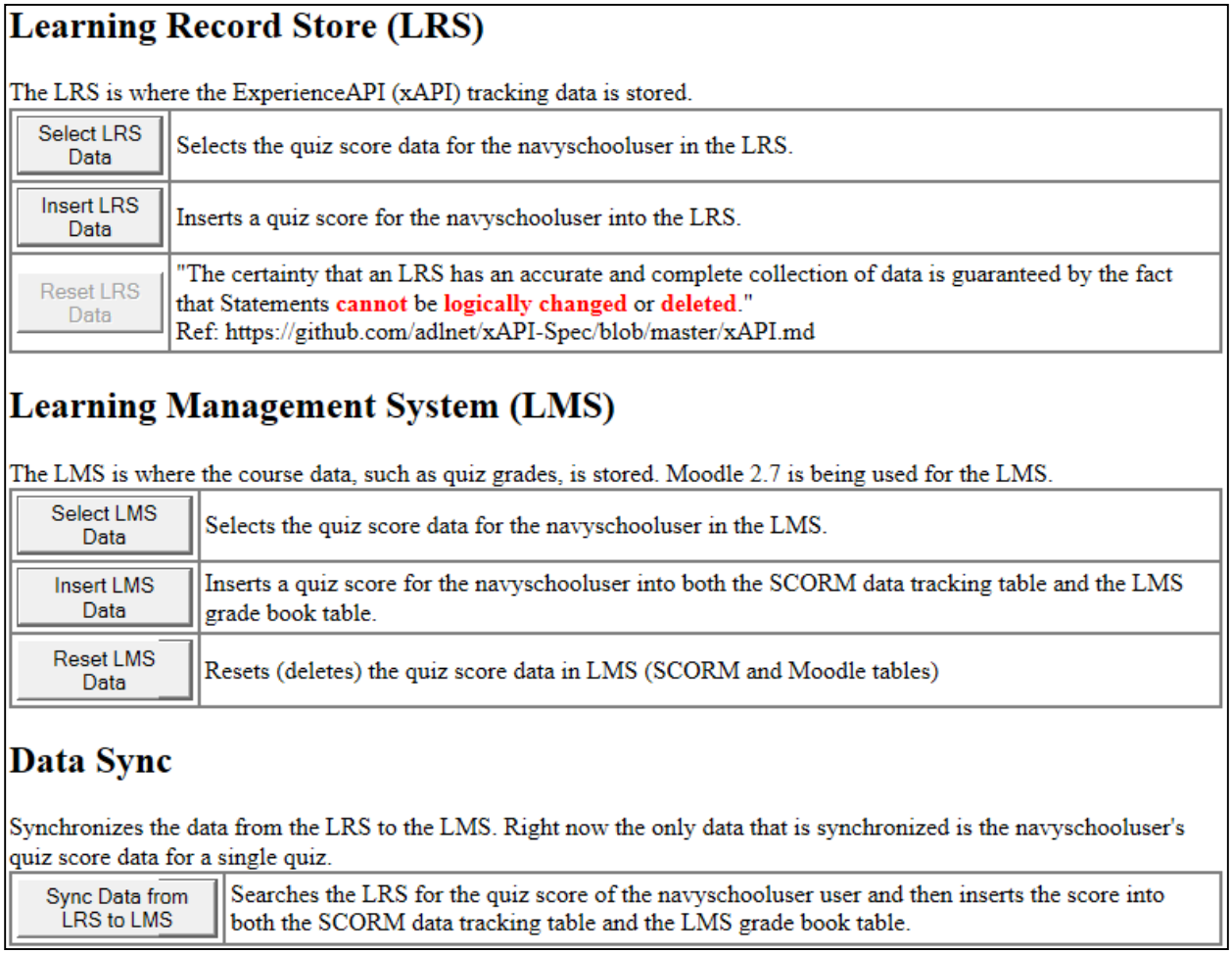


Figure 21. Data Sync website.

First the data is pulled from the LRS as shown in Figure 22. In Figure 23 the details of the LRS call is shown. The jxapi library is used for connecting to the LRS to

pull the data. The data is converted by the jxapi from a JSON statement into a java object statement. By default the LRS will return the last 100 statements with a pointer that can be used to get the next 100 statements (and so on) if no filter parameters are added. For the prototype system the LRS data is filtered by the specific quiz (http://nps.edu/moodle/quiz2) of interest. The assumption is the quiz has only been taken by a single Student so at this point the list of statements returned are only for the specific quiz and the test Student. Since the xAPI does not allow deleting of statements from the LRS, after the first run of the prototype there will always be more than one statement returned by this filter. Since the statements are returned in chronological order, the first statement (element 0) of the returned list of statements is the most recent quiz data for the test Student and that is the data used.

```
public static void syncData() throws Exception {
    logger.debug("Getting LRS data...");
    Statement statement = LrsService.selectStatement();
    logger.debug("statement: " + ToStringBuilder.reflectionToString(statement));
    logger.debug("statement.getObject(): " + ToStringBuilder.reflectionToString(statement.getObject()));
    Activity activity = (Activity) statement.getObject();
    logger.debug("activity: " + ToStringBuilder.reflectionToString(activity));
    ActivityDefinition activityDefinition = activity.getDefinition();
    logger.debug("activityDefinition: " + ToStringBuilder.reflectionToString(activityDefinition));
    HashMap<String, JsonElement> extensionsMap = activityDefinition.getExtensions();
    logger.debug("extensionsMap.size(): " + extensionsMap.size());
    JsonElement jsonElement = extensionsMap.get("http://nps.edu/xapi/lms");
    logger.debug("jsonElement: " + ToStringBuilder.reflectionToString(jsonElement));
    Gson gson = new Gson();
    LrsObjectDefinitionExtensions lrsObjectDefinitionExtensions = gson.fromJson(jsonElement, LrsObjectDefinitionExtensions.class);
    logger.debug("lrsObjectDefinitionExtensions: " + ToStringBuilder.reflectionToString(lrsObjectDefinitionExtensions));
```

Figure 22. Data Sync function pulling data from LRS.

```
public static Statement selectStatement() throws Exception {
    StatementClient client = getClient();
    StatementResult results = client.filterByActivity("http://nps.edu/moodle/quiz2").getStatements();
    int counter = 0;
    for (Statement element : results.getStatements()) {
        counter++;
        logger.debug("counter: " + counter);
        logger.debug("element: " + ToStringBuilder.reflectionToString(element));
    }
    return results.getStatements().get(0);
}
```

Figure 23. Data Sync function pulling data from LRS detail.

Second the data is extracted and transformed from the java object statement and default values are set that are not of interest for this study but required by the Moodle LMS as shown in Figure 24.

```
//Extracts data
logger.debug("Extracting LRS data...");
int userId = Integer.valueOf(lrsObjectDefinitionExtensions.getUserId());
logger.debug("userId: " + userId);
int scormId = Integer.valueOf(lrsObjectDefinitionExtensions.getScormId());
logger.debug("scormId: " + scormId);
int scoId = Integer.valueOf(lrsObjectDefinitionExtensions.getScormScoId());
logger.debug("scoId: " + scoId);
int attempt = Integer.valueOf(lrsObjectDefinitionExtensions.getAttempt());
logger.debug("attempt: " + attempt);
int score = Integer.valueOf(lrsObjectDefinitionExtensions.getScore());
logger.debug("score: " + score);
String scoreAsString = String.valueOf(score);
logger.debug("scoreAsString: " + scoreAsString);
float scoreAsFloat = Float.valueOf(score);
logger.debug("scoreAsFloat: " + scoreAsFloat);
int scoreMin = 0;//TODO could make this another field in the LRS
logger.debug("scoreMin: " + scoreMin);
String scoreMinAsString = String.valueOf(scoreMin);
logger.debug("scoreMinAsString: " + scoreMinAsString);
int scoreMax = 100;//TODO could make this another field in the LRS
logger.debug("scoreMax: " + scoreMax);
String scoreMaxAsString = String.valueOf(scoreMax);
logger.debug("scoreMaxAsString: " + scoreMaxAsString);
String lessonLocation = "15";//TODO could make this another field in the LRS
logger.debug("lessonLocation: " + lessonLocation);
String totalTime = "00:03:34.00";//TODO could make this another field in the LRS
logger.debug("totalTime: " + totalTime);
//Dates
String dateAsStringFull = statement.getTimestamp();//Example: 2014-06-15T18:35:10.976673+00:00
logger.debug("dateAsStringFull: " + dateAsStringFull);
String dateAsStringShort = dateAsStringFull.split("\\.")[0];//Example: 2014-06-15T18:35:10
logger.debug("dateAsStringShort: " + dateAsStringShort);
String dateAsStringDatePart = dateAsStringShort.split("T")[0];//Example: 2014-06-15
logger.debug("dateAsStringDatePart: " + dateAsStringDatePart);
int year = Integer.valueOf(dateAsStringDatePart.split("-")[0]);//Example: 2014
logger.debug("year: " + year);
int month = Integer.valueOf(dateAsStringDatePart.split("-")[1]);//Example: 06
logger.debug("month: " + month);
int day = Integer.valueOf(dateAsStringDatePart.split("-")[2]);//Example: 15
logger.debug("day: " + day);
String dateAsStringTimePart = dateAsStringShort.split("T")[1];//Example: 18:35:10
logger.debug("dateAsStringTimePart: " + dateAsStringTimePart);
int hour = Integer.valueOf(dateAsStringTimePart.split(":")[0]);//Example: 18
logger.debug("hour: " + hour);
int minute = Integer.valueOf(dateAsStringTimePart.split(":")[1]);//Example: 35
logger.debug("minute: " + minute);
int second = Integer.valueOf(dateAsStringTimePart.split(":")[2]);//Example: 10
logger.debug("second: " + second);
Calendar cal = Calendar.getInstance();
cal.set(year, month, day, hour, minute, second);
long dateAsLongInsert = cal.getTimeInMillis();
logger.debug("dateAsLongInsert: " + dateAsLongInsert);
String dateAsStringStart = String.valueOf(dateAsLongInsert); //TODO could make this another field in the LRS
logger.debug("dateAsStringStart: " + dateAsStringStart);
```

Figure 24. Data Sync function extracting, transforming, and setting default values.

Lastly the data is put into the LMS java objects and inserted into the LMS. The LMS used in this study is Moodle version 2.7 and it does not provide web services for

altering SCORM related data in the LMS. Because of this, the study reverse engineered how Moodle handled SCORM data related for a SCORM quiz package within the database. The discussion below is the result of that reverse engineering to mimic how Moodle stores the SCORM data in its database.

The first LMS java object populated is the SCO data as shown in Figure 25. For this prototype, the only value of interest is the quiz score data (cmi.core.score.raw) but several other values are set as required by the Moodle database. The generic code used to insert a SCO into the LMS is shown in Figure 26.

```
logger.debug("Inserting LMS data...");
MoodleScormScoTrackBean sco = new MoodleScormScoTrackBean();
sco.setUserId(userId);
sco.setScormId(scormId);
sco.setScoId(scoId);
sco.setAttempt(attempt);
sco.setDateAsLong(dateAsLongInsert);

sco.setElement(ScormConstants.ELEMENT_X_START_TIME);
sco.setValue(dateAsStringStart);
LmsService.insertSco(sco);

sco.setElement(ScormConstants.ELEMENT_CMI_CORE_SCORE_RAW);
sco.setValue(scoreAsString);
LmsService.insertSco(sco);

sco.setElement(ScormConstants.ELEMENT_CMI_CORE_SCORE_MIN);
sco.setValue(scoreMinAsString);
LmsService.insertSco(sco);

sco.setElement(ScormConstants.ELEMENT_CMI_CORE_SCORE_MAX);
sco.setValue(scoreMaxAsString);
LmsService.insertSco(sco);

sco.setElement(ScormConstants.ELEMENT_CMI_CORE_LESSON_LOCATION);
sco.setValue(lessonLocation);
LmsService.insertSco(sco);

sco.setElement(ScormConstants.ELEMENT_CMI_CORE_LESSON_STATUS);
sco.setValue(ScormConstants.VALUE_CMI_CORE_LESSON_STATUS_PASSED);
LmsService.insertSco(sco);

sco.setElement(ScormConstants.ELEMENT_CMI_CORE_TOTAL_TIME);
sco.setValue(totalTime);
LmsService.insertSco(sco);
```

Figure 25. Data Sync function SCO Moodle LMS java objects populated.

```
public static void insertSco(MoodleScormScoTrackBean sco) throws Exception {
    Connection con = null;
    PreparedStatement ps = null;
    try {
        con = DatabaseConnectionManager.getConnection();
        String query = "INSERT INTO mdl_scorm_scoes_track ( " +
                            "userid, " +
                            "scormid, " +
                            "scoid, " +
                            "attempt, " +
                            "element, " +
                            "value, " +
                            "timemodified " +
                       ") " +
                       "VALUES (?,?,?,?,?,?,?)";
        ps = con.prepareStatement(query);
        ps.setInt(1, sco.getUserId());
        ps.setInt(2, sco.getScormId());
        ps.setInt(3, sco.getScoId());
        ps.setInt(4, sco.getAttempt());
        ps.setString(5, sco.getElement());
        ps.setString(6, sco.getValue());
        ps.setLong(7, sco.getDateAsLong());
        int recordsInserted = ps.executeUpdate();
        logger.debug("recordsInserted: " + recordsInserted);
    }
    finally {
        DatabaseConnectionManager.closeResource(con, ps);
    }
}
```

Figure 26. Data Sync function SCO inserted into Moodle database call.

As part of the Moodle database reverse engineering to have the Student's grade show up via the Moodle website for the Teacher, several grade related objects and tables need to be populated as shown in Figures 27 and Figure 28. (Note: These are not SCORM related data elements or tables.)

```
//Grade
//Enter a itemid 1 grade with no rawgrade score.
//Doesn't appear to affect anything without it, but Moodle does this.
MoodleGradeGrades grade1 = new MoodleGradeGrades();
grade1.setUserId(userId);
grade1.setItemId(1);
grade1.setRawGradeMin(scoreMin);
grade1.setRawGradeMax(scoreMax);
grade1.setFinalGrade(scoreAsFloat);
grade1.setDateAsLong(dateAsLongInsert);
LmsService.insertGrade(grade1);
//Grade actually used score
MoodleGradeGrades grade2 = new MoodleGradeGrades();
grade2.setUserId(userId);
grade1.setUserModifiedId(userId);
grade2.setItemId(2);
grade2.setRawGrade(scoreAsFloat);
grade2.setRawGradeMin(scoreMin);
grade2.setRawGradeMax(scoreMax);
grade2.setFinalGrade(scoreAsFloat);
grade2.setDateAsLong(dateAsLongInsert);
LmsService.insertGrade(grade2);
```

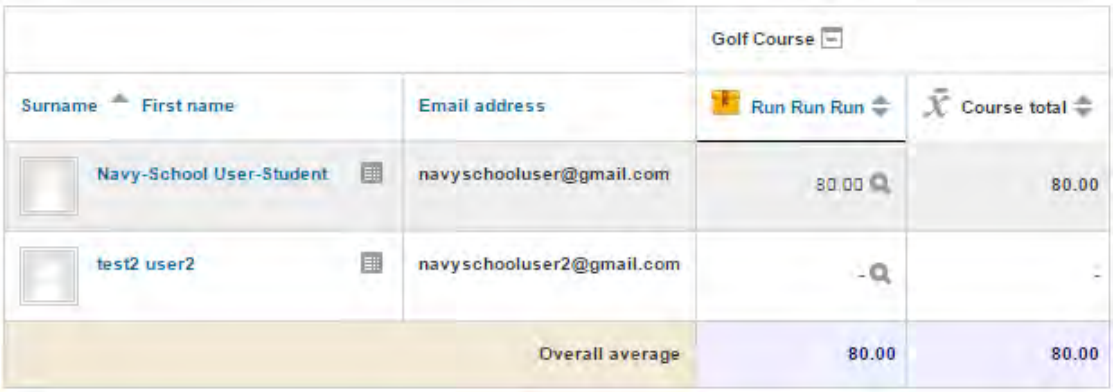
Figure 27. Data Sync function Moodle LMS grade java objects populated.

```
//Grade History
//Source: mod/scorm
MoodleGradeGradesHistory gradeHistoryScorm = new MoodleGradeGradesHistory();
gradeHistoryScorm.setUserId(userId);
gradeHistoryScorm.setUserModifiedId(userId);
gradeHistoryScorm.setItemId(2);
gradeHistoryScorm.setRawGrade(scoreAsFloat);
gradeHistoryScorm.setRawGradeMin(scoreMin);
gradeHistoryScorm.setRawGradeMax(scoreMax);
gradeHistoryScorm.setFinalGrade(scoreAsFloat);
gradeHistoryScorm.setDateAsLong(dateAsLongInsert);
gradeHistoryScorm.setActionId(1);
gradeHistoryScorm.setOldId(1);
gradeHistoryScorm.setSource("mod/scorm");
LmsService.insertGradeHistory(gradeHistoryScorm);
//Source: system
MoodleGradeGradesHistory gradeHistorySystem = new MoodleGradeGradesHistory();
gradeHistorySystem.setUserId(userId);
gradeHistorySystem.setItemId(1);
gradeHistorySystem.setRawGradeMin(scoreMin);
gradeHistorySystem.setRawGradeMax(scoreMax);
gradeHistorySystem.setDateAsLong(dateAsLongInsert);
gradeHistorySystem.setActionId(1);
gradeHistorySystem.setOldId(2);
gradeHistorySystem.setSource("system");
LmsService.insertGradeHistory(gradeHistorySystem);
//Source: aggregation
MoodleGradeGradesHistory gradeHistoryAggregation = new MoodleGradeGradesHistory();
gradeHistoryAggregation.setUserId(userId);
gradeHistoryAggregation.setItemId(1);
gradeHistoryAggregation.setRawGradeMin(scoreMin);
gradeHistoryAggregation.setRawGradeMax(scoreMax);
gradeHistoryAggregation.setFinalGrade(scoreAsFloat);
gradeHistoryAggregation.setDateAsLong(dateAsLongInsert);
gradeHistoryAggregation.setActionId(2);
gradeHistoryAggregation.setOldId(2);
gradeHistoryAggregation.setSource("aggregation");
LmsService.insertGradeHistory(gradeHistoryAggregation);
```

Figure 28. Data Sync function Moodle grade history java objects populated.

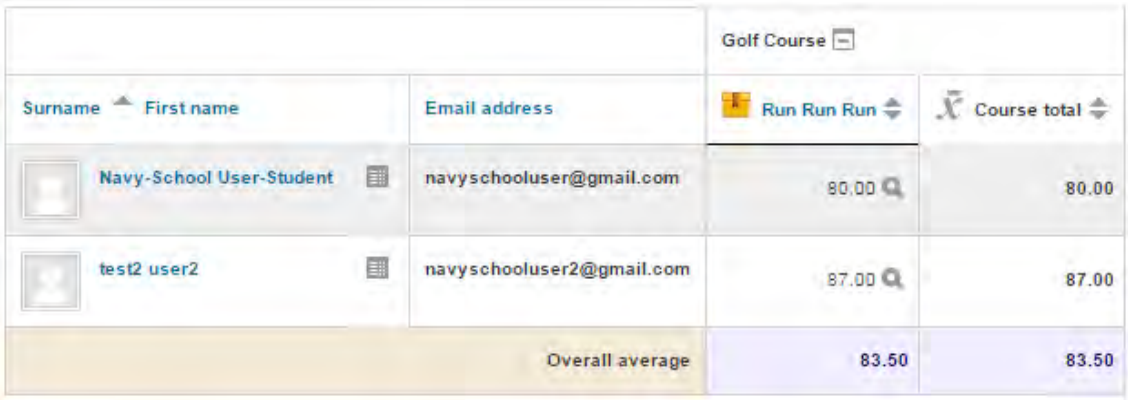
C. TEACHER USER AND LMS VIEW GRADE

For the prototype system two students were created in the Moodle LMS. The first student (Navy-School User-Student) took the SCORM sample golf quiz within Moodle to get a score. The second student (test2 user2) took the quiz via the Quiz website. Figure 29 shows Moodle grades before the second student's quiz grade was copied into Moodle via the Data Sync function. Figure 30 show the Moodle grades after the Data Sync function has run.



		Golf Course	
Surname First name	Email address	Run Run Run	Course total
Navy-School User-Student	navyschooluser@gmail.com	80.00	80.00
test2 user2	navyschooluser2@gmail.com	-	-
	Overall average	80.00	80.00

Figure 29. Teacher viewing the Student's quiz grade in Moodle LMS before Data Sync function run.



		Golf Course	
Surname First name	Email address	Run Run Run	Course total
Navy-School User-Student	navyschooluser@gmail.com	80.00	80.00
test2 user2	navyschooluser2@gmail.com	87.00	87.00
	Overall average	83.50	83.50

Figure 30. Teacher viewing the Student's quiz grade in Moodle LMS after Data Sync function run.

V. CONCLUSIONS AND FUTURE RESEARCH

A. CONCLUSION

The xAPI along with its LRS can overcome the NeL's AtlasPro LMS limitation of only tracking learning from a user's desktop computer using an Internet browser. It is possible to track SCORM related data elements in a LRS and then extract that same data and place it into a LMS. This can enable tracking from much larger pool of devices and software. Another benefit of using the xAPI is that it separates the tracking of learning from the delivery of learning content that makes the overall system more modular and flexible.

There are several areas specific to the prototype system that could be improved upon. Obtaining and installing the AtlasPro LMS in place of the Moodle LMS would more closely align the prototype to the NeL production environment. Placing the Quiz website and Data Sync website each on their own virtual server would better decouple the components. Having to reverse engineer the Moodle LMS SCORM database interaction was not ideal and would be better if Moodle provided a web service for SCORM interactions. Automating the triggering of the Data Sync function would remove the need for it being manually triggered by the Administrator user.

While the prototype system has the Student Device interact with the LRS via the Quiz website, it is possible to have the Student Device interact with a LRS directly. The Student Device doesn't even have to use an Internet browser for the interaction. This is because the xAPI works over HTTP. As long as the Student Device has software that can communicate over HTTP and a network connection to the LRS, then the Student Device can interact with the LRS. Because of this, the Student Device doesn't have to be a desktop, laptop, tablet, or even a smartphone but could any type of computing device with a network connection such as a hand-held scanner or pilot tactical helmet. It may even be possible to put the LRS on the computing device itself.

B. FUTURE RESEARCH

The prototype system used only a few of the SCORM standard data elements but there are many more. It could be beneficial to have a complete mapping of SCORM data elements from a LRS to a LMS. Is it possible to put every type of SCORM data element into a LRS? Can those same SCORM data elements then be transferred into a LMS?

HTML5 provides some powerful new features such as local (offline) device storage, and geographical location of the device, and native (no plug-in) support for audio, video, interaction, and drawing (MacDonald, 2011). Can HTML5, along with the xAPI, provide the same features and functionality that training material in the existing NeL LMS provides? What new features and functionality are possible?

As the prototype system is a proof of concept, it does not take into account many of the DOD related security concerns. What are the security concerns with the xAPI/ LRS? How can these security concerns be mitigated?

While the xAPI can augment SCORM, the xAPI isn't limited to just storing SCORM data elements. What non-SCORM data elements would be useful to track and store in a LRS?

There is no requirement that the computing device must connect to the LRS over a network connection. It may be possible to put the LRS on the computing device itself that may be beneficial in certain situations. Can a LRS be installed directly on the computing device that is going to communicate with the LRS? If so, what possibilities does this open up?

These are just a few directions the research could be taken.

APPENDIX. XAPI/LRS STATEMENT IN JSON FORMAT


```
{
	"id": "f7c4fe63-c314-4525-883a-407f2fe06dd4",
	"verb": {
		"id": "http://adlnet.gov/expapi/verbs/completed",
		"display": {
			"en-US": "completed"
		}
	},
	"actor": {
		"mbox": "mailto:navyschooluser@gmail.com",
		"name": "NavySchoolUser",
		"objectType": "Agent"
	},
	"object": {
		"definition": {
			"extensions": {
				"http://nps.edu/xapi/lms": {
					"attempt": "1",
					"courseId": "2",
					"userId": "4",
					"moodleItemId": "2",
					"score": "95",
					"scormId": "1",
					"scormScoId": "2"
				}
			},
			"name": {
				"en-US": "Quiz 2"
			}
		},
		"id": "http://nps.edu/moodle/quiz2",
		"objectType": "Activity"
	},
	"authority": {
		"mbox": "mailto:wemiller@nps.edu",
		"name": "Rett Miller",
		"objectType": "Agent"
	},
	"timestamp": "2014-11-02T21:43:51.227355+00:00",
	"stored": "2014-11-02T21:43:51.227355+00:00",
	"version": "1.0.0"
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Advanced Distributed Learning. (2013). *Background and history*. Retrieved September 13, 2014, from Advanced Distributed Learning: http://www.adlnet.gov/tla/experience-api/background-and-history/

Advanced Distributed Learning. (2014, May 28). *Experience API*. Retrieved June 1, 2014, from GitHub: https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md

Advanced Distributed Learning. (n.d.). Training & learning architecture (TLA): Project Tin Can. Retrieved September 13, 2014, from Advanced Distributed Learning: http://www.adlnet.gov/tla/tin-can/#tab-research

Berking, P., & Gallagher, S. (2013, May 14). Retrieved September 20, 2014, from Advanced Distributed Learning: http://www.adlnet.gov/wp-content/uploads/2013/05/Choosing_an_LMS.pdf

Bray, T. (2014, March). *The javascript object notation (JSON) data interchange format.* Retrieved October 26, 2014, from Internet Engineering Task Force: http://tools.ietf.org/html/rfc7159

Brusino, J. (2012, June 1). *The next generation of SCORM: A Q&A with Aaron Silvers*. Retrieved November 3, 2013, from astd.org: http://www.astd.org/Publications/Newsletters/Learning-Circuits/Learning-Circuits-Archives/2012/06/The-Next-Generation-of-SCORM-a-Q-and-a-with-Aaron-Silvers

Experience API. (2014, January 2). Advanced Distributed Learning (ADL) co-laboratories. Retrieved January 20, 2014, from GitHub: https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md

Hyland Software. (2009, Febuary 18). *The trouble with legacy systems.* Retrieved August 16, 2014, from OnBase: https://www.onbase.com/~/media/Files/Hyland/WhitePaper/wp_trouble-with-legacy-systems.ashx

MacDonald, M. (2011). *HTML5*: *The missing manual*. Sebastopol : O'Reilly Media, Inc.

Mobify. (2012, October 25). *Working with iframes*. Retrieved August 23, 2014, from Mobify: https://support.mobify.com/customer/portal/articles/547042-working-with-iframes

Navy eLearning. (2010). Retrieved November 3, 2013, from MilitarySpot.com: http://www.militaryspot.com/navy/navy-elearning-nel/

Poltrack, J., Haag, J., Hruska, N., & Johnson, A. (2012). *The next generation of SCORM: Innovation for the global force*. Retrieved November 16, 2013, from Advanced Distributed Learning: http://www.adlnet.gov/wp-content/uploads/2012/12/12114.pdf

Rustici Software. (n.d.). *Golf examples*. Retrieved February 7, 2014, from Rustici Software: http://scorm.com/scorm-explained/technical-scorm/golf-examples/

Rustici, M. (2009, January 12). *SCORM 2004 overview for developers*. Retrieved October 26, 2014, from Rustici Software: http://scorm.com/scorm-explained/technical-scorm/scorm-2004-overview-for-developers/

Sea Warrior Program and Naval Education and Training Command Public Affairs. (2013, February 22). New learning management system improves navy e-learning efficiency. Retrieved November 3, 2013, from Navy.mil: http://www.navy.mil/submit/display.asp?story_id=72301

Szabo, M., & Flesher, K. (2002). CMI Theory and Practice: Historical Roots of Learning Management Systems. *World conference on e-learning in corporate, government, healthcare, and higher education* (pp. 929–936). Montreal: Association for the Advancement of Computing in Education.

Tin Can API. (n.d.). *SCORM vs the tin can API*. Retrieved September 13, 2014, from Tin Can API: http://tincanapi.com/scorm-vs-the-tin-can-api/

Whitaker, A. (2012, July 19). *An introduction to the tin can API*. Retrieved November 3, 2013, from The Training Business: http://www.thetrainingbusiness.com/softwaretools/tin-can-api/

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California