# ULAM Headless Course Format

ver. b7305bd

Mateusz Wojczal. November 2021

Latest version of this document is available here: PDF, web

# Contents

# Glossary

**ADL** Advanced Distributed Learning (ADL) Initiative from the Office of the United States Secretary of Defense.

**cmi5** cmi5 is an xAPI Profile that provides rules defining how online courses are imported, launched and tracked using a Learning Management System (LMS) and xAPI.

**JSON** JavaScript Object Notation (JSON) is a lightweight data-interchange format, that is both easy to read and write for humans and easy to parse and generate by machines.

**JSON Schema** JSON Schema is a vocabulary that allows you to annotate and validate JSON documents.

**LMS** Learning management system (LMS) is a type of software used for creation, administration and presentation of educational courses and their contents.

**LRS** A Learning Record Store (LRS) is a data store system that serves as a repository for learning records collected from connected systems where learning activities are conducted.

**Monolith architecture** In software engineering, a monolithic application describes a single-tiered software application in which the user interface and data access code are combined into a single program from a single platform.

**PWA** A progressive web application (PWA), is a type of application software delivered through the web, built using common web technologies. It is intended to work on any platform that uses a standards-compliant browser, including both desktop and mobile devices.

**SCORM** Shareable Content Object Reference Model (SCORM) is a set of technical standards for eLearning software products.

**Separation of concerns** In computer science, separation of concerns (SoC) is a design principle for separating a computer program into distinct sections, each of which addresses a separate concern.

**SPA** A single-page application (SPA) is a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of a web browser loading entire new pages.

**xAPI** The Experience API (xAPI) is an e-learning software specification that allows learning content and learning systems to speak to each other in a manner that records and tracks all types of learning experiences.

**XML** Extensible Markup Language (XML) is a markup language for coding documents, that allows designers to create their own customized tags for structuring document contents. XML is widely used as a data-interchange format in web services.

**ZIP** ZIP is an archive file format that supports lossless data compression.

# The Abstract

Current e-learning software comes with a huge technological debt and does not respond to market needs as fast as other IT segments can. The main reason is dependency on obsolete formats like SCORM that are still widely used, and which do not separate data layer from the presentation layer. There is a need from market for existence of better designed and better implemented formats.

# The Introduction

Current e-learning formats do not separate data from presentation layers. Current e-learning content is not portable and is not designed to age well.

With separation of layers the content could be presented in a modern way every time a new device is introduced. If SCORM courses were designed in this fashion back in 2000s it would be straightforward to convert them to any of commonly used devices, like smartphones, smartwatches, smart TVs, and anything else capable of running current software. But because of a few wrong design decisions made without necessary foresight, we are now stuck with this format and with obsolete courses.

Back in the days when Advanced Distributed Learning was creating SCORM (adapting older AICC HACP desktop format) there was little variance in commonly available devices capable of interacting with eLearning software - most personal computers used the same web browser, same operating system and had similar displays with common 1024x768 pixel resolution. If there were any exceptions, they were minimal and negligible. Browsers were not able to do much more than to show server response in HTML format after (synchronous) client request. Everything displayed in browser window was rendered by server, and even if there was any separation of layers, it existed only on the server side.

Organizations that are working on e-learning standards are responding to market needs very slowly. Their latest specification `cmi5`, which does solve many of the shortcomings of previous standards, is already 6 years old and not commonly adopted - the most popular format SCORM 2004 4th Edition was published in 2009.

The headless approach seems to be solving all of the issues that modern e-learning and LMSes do have. The separation of content and it's players allows to create courses that works well on any device and do age well. Courses designed in this favour most likely will be able to be played on smart devices not yet in existence.

# Evolution of e-learning

## History of e-learning formats

The most popular e-learning formats are created and managed by the Advanced Distributed Learning (ADL) Initiative from the Office of the United States Secretary of Defense.

Before e-learning was used in the web browser environment there was AICC's format created in 1993. First widely used format was AICC HACP (released in 1998) which later evolved into SCORM 1.0 that was released in year 2000. Since then, up to the latest SCORM 2004 4th Edition version (released in 2009) it is the most popular and adopted collection of standards and specifications for web-based e-learning.

SCORM (which is an abbreviation of Sharable Content Object Reference Model) describes communication between client side content and a host system, and describes package structure, defining how whole course, consisting of resource files and XML manifest can be packaged into ZIP archives that are called "Package Interchange Format"[1].

Since SCORM has many shortcomings, The Experience API, also known as Tin Can API or `xAPI` was released, and later cmi5 format was created, providing a set of rules intended to achieve interoperability in a traditional Learning Management System environment.

xAPI specification removes content for it description, and allows the content to send "statements" based around [actor] [verb] [object], or "I – did – this" to a Learning Record Store (LRS) which can be a part of a Learning Management System, but can also live on its own, or as part of another separate system.

[1] Technical Specification 4th Ed.. SCORM. Retrieved 2017-05-22.

The table below [2] summarizes the comparison of each standard:

| Format | Released | Pages | Widely Used | Run-Time | Pack-aging | Meta-data | Sequen-cing | Works Cross Domain |
|---|---|---|---|---|---|---|---|---|
| AICC HACP | Feb 1998 | 337 | Yes | Yes | Yes | No | No | Yes |
| SCORM 1.0 | Jan 2000 | 219 | No | Yes | Yes | Yes | No | No |
| SCORM 1.1 | Jan 2001 | 233 | No | Yes | Yes | Yes | No | No |
| SCORM 1.2 | Oct 2001 | 524 | Yes | Yes | Yes | Yes | No | No |
| SCORM 2004 "1st Edition" | Jan 2004 | 1,027 | No | Yes | Yes | Yes | Yes | No |
| SCORM 2004 2nd Edition | Jul 2004 | 1,219 | Yes | Yes | Yes | Yes | Yes | No |
| SCORM 2004 3rd Edition | Oct 2006 | 1137 | Yes | Yes | Yes | Yes | Yes | No |
| SCORM 2004 4th Edition | Mar 2009 | 1162 | Yes | Yes | Yes | Yes | Yes | No |
| IMS Common Cartridge | Oct 2008 | 135 | No | No | Yes | Yes | No | Yes |
| IMS LTI | May 2010 | 25 | In Academic LMSs | Yes | No | No | No | Yes |
| The Experience API (xAPI) | April 26, 2013 | 85 | Not Yet | Yes | Partial | No | No | Yes |
| cmi5 (a companion to xAPI) | June 1, 2016 | 48 | Not Yet | Yes | Yes | No | No | Yes |

## What is Learning Management System - LMS

The most popular LMS in the world is Moodle [3], released on 20 August 2002, as Open Source software, distributed under the GNU General Public License, which is probably one of the reasons of its popularity.

Moodle is an application written in PHP that is being served by a machine that uses PHP. That means that all of the actions for administrators, course creators, students and any other roles does require to connect to the hosting machine (server) that serves Moodle. This is an example of monolith architecture, which means that all Moodle components are PHP based working on one machine that parses moodle source code every time there is a request from the browser. Components of the program are interconnected and interdependent in a tightly-coupled architecture.

Most other popular LMS work very similarly, as the monolith architecture is the most prevalent design among available LMS software.

## LMS Monolith Architecture

The diagram below show Moodle monolith architecture

[2] A timeline and description of the eLearning standards.. SCORM
[3] Moodle - Open-source learning platform | Moodle.org

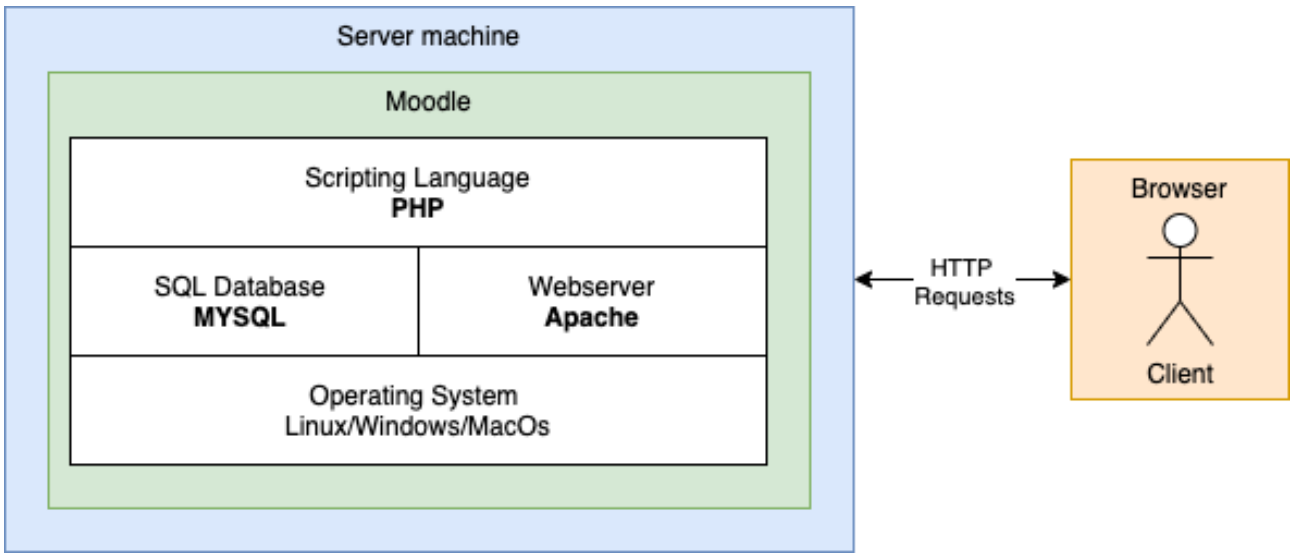


Figure 1: LMS monolith architecture. Moodle technical architecture

All the LMS Features, including

- Managing courses, users and roles
- Online assessment
- User feedback
- Synchronous and Asynchronous Learning
- Learning Analytics

are handled directly from the server, the response is prepared before being sent in HTML format by PHP preprocessor, and the client gets the final rendered HTML document.

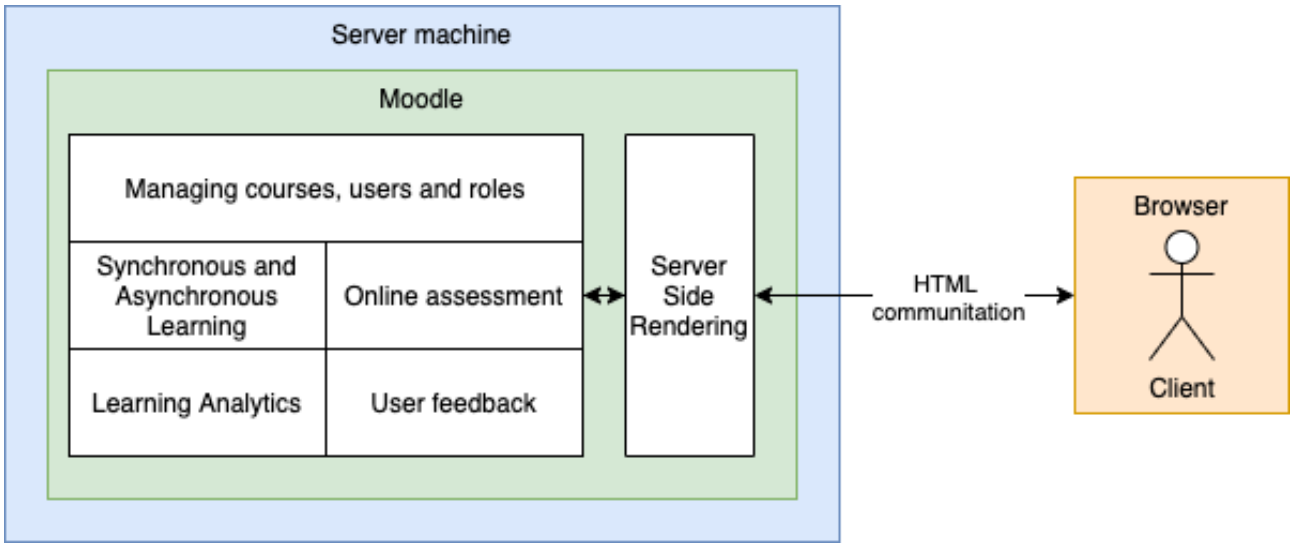


Figure 2: LMS monolith architecture. Moodle functional architecture

All the above means that Moodle and dedicated server is required all the time for all e-learning activities.

## Process of publishing the course

Standard way of creating and publishing SCORM compliant course is described by these four steps:

1. Creating of a course in an e-Learning authoring tool (like Adobe Captivate [4]) or using tools integrated with the LMS software
2. Course is published as a SCORM package (a ZIP file)
3. SCORM package is being uploaded to the LMS, using an upload form, and is prepared to be published
4. LMS publishes the course to the students; all results of activities are stored in the LMS

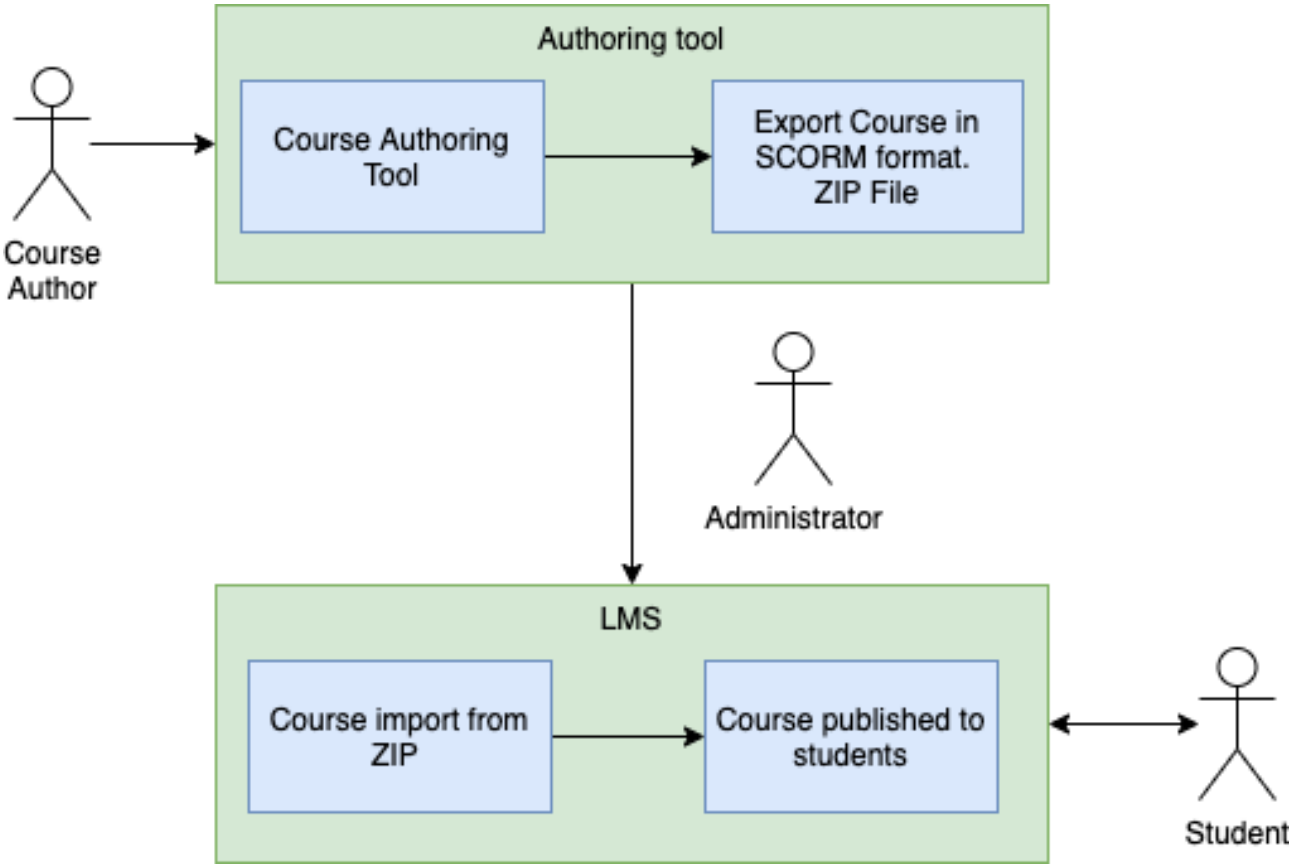


Figure 3: Process of publishing the course

The process above is one directional - which means that SCORM package is a closed format, once published it cannot be edited. In order to make any changes, even correcting simple typo, the whole process must be repeated - course needs to be changed in authoring tool, packaged into zip, then uploaded and (re)published.

## Introduction of Experience API (xAPI) and related technologies

One of the shortcomings of SCORM that influenced the introduction of extended formats was the limitation of capability to track and analyse activities of students to only within the same LMS.

That means that the course, LMS and student progress were inseparable.

xAPI specification removes content for it description, and allows the content to send "statements" based around [actor] [verb] [object], or "I – did – this" to a Learning Record Store (LRS) which can be a part of a Learning Management System, but can also live on its own (or as a part of another, separate, system). This was the first step for **Separation of concerns** in e-learning.

[4] Adobe Captivate

### Learning Record Store (LRS)

A Learning Record Store is an application, external to course runtime, that receives and sends data in JSON format from and to the course runtime - it is an essential component in Experience API process flow. What is a big difference from previous e-learing standards, is that the specification does not tell how the course should be played (how course runtime should work), it just defines how the runtime must communicate with the interface (LRS) though xAPI Statements. The statements are open to be extended and each implementation can introduce its own statements.

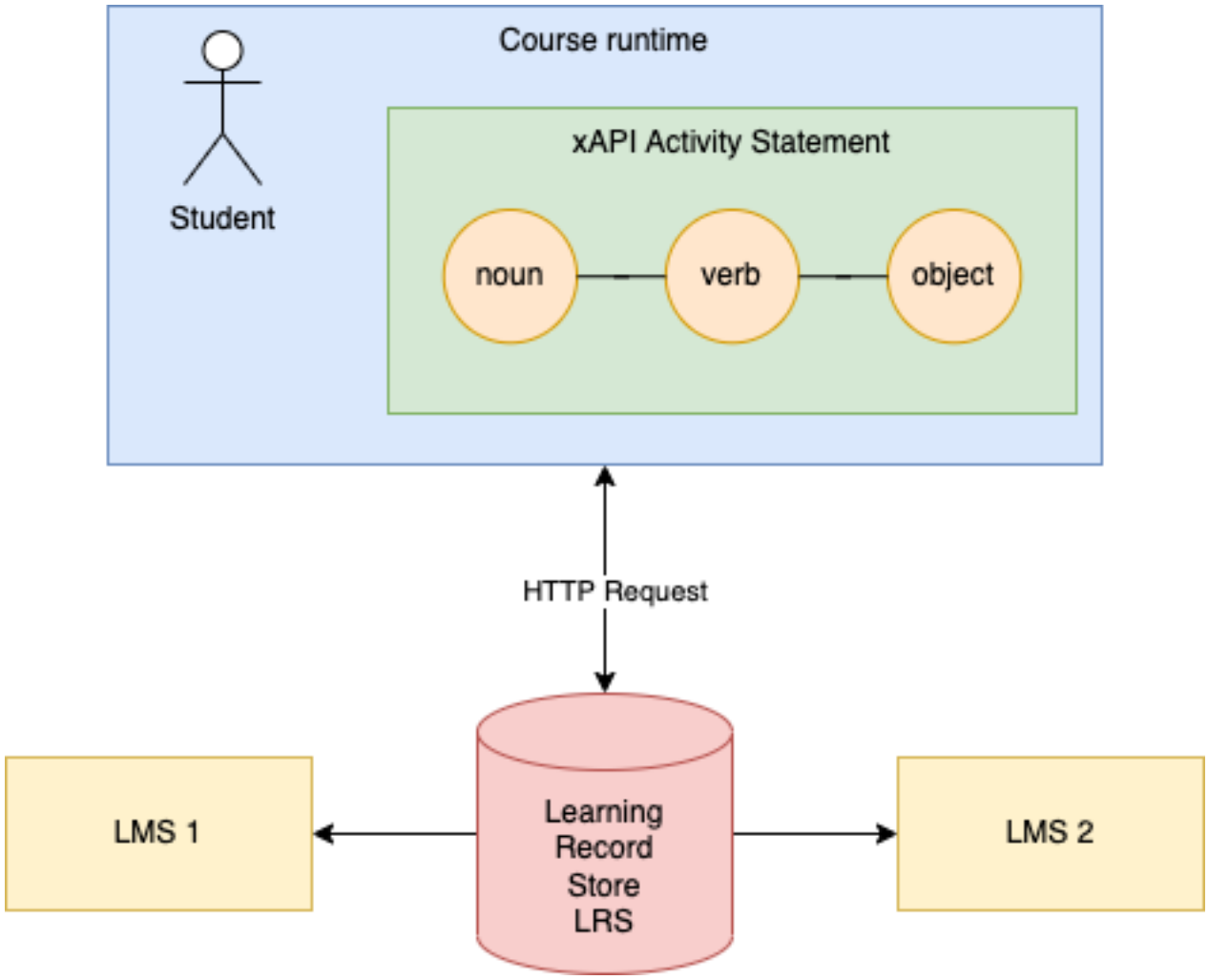


Figure 4: Experience API (xAPI) process flow with Learning Record Store (LRS)

### cmi5 Specification

> cmi5 is a "profile" for using the xAPI specification with traditional learning management (LMS) systems. [5]

The specification of cmi5 is a set of rules providing all the capabilities of SCORM and xAPI. It is similar to SCORM in a way, as it also requires a XML file manifest, yet it does introduce the Assignable Unit (AU) - separately launchable learning content presentation. The AU is the unit of tracking and management. The AU collects data on the learner and sends it to the LMS (through an LRS).

[5] cmi5: Technical 101 Terminology.
[6] Conceptual Overview of cmi5

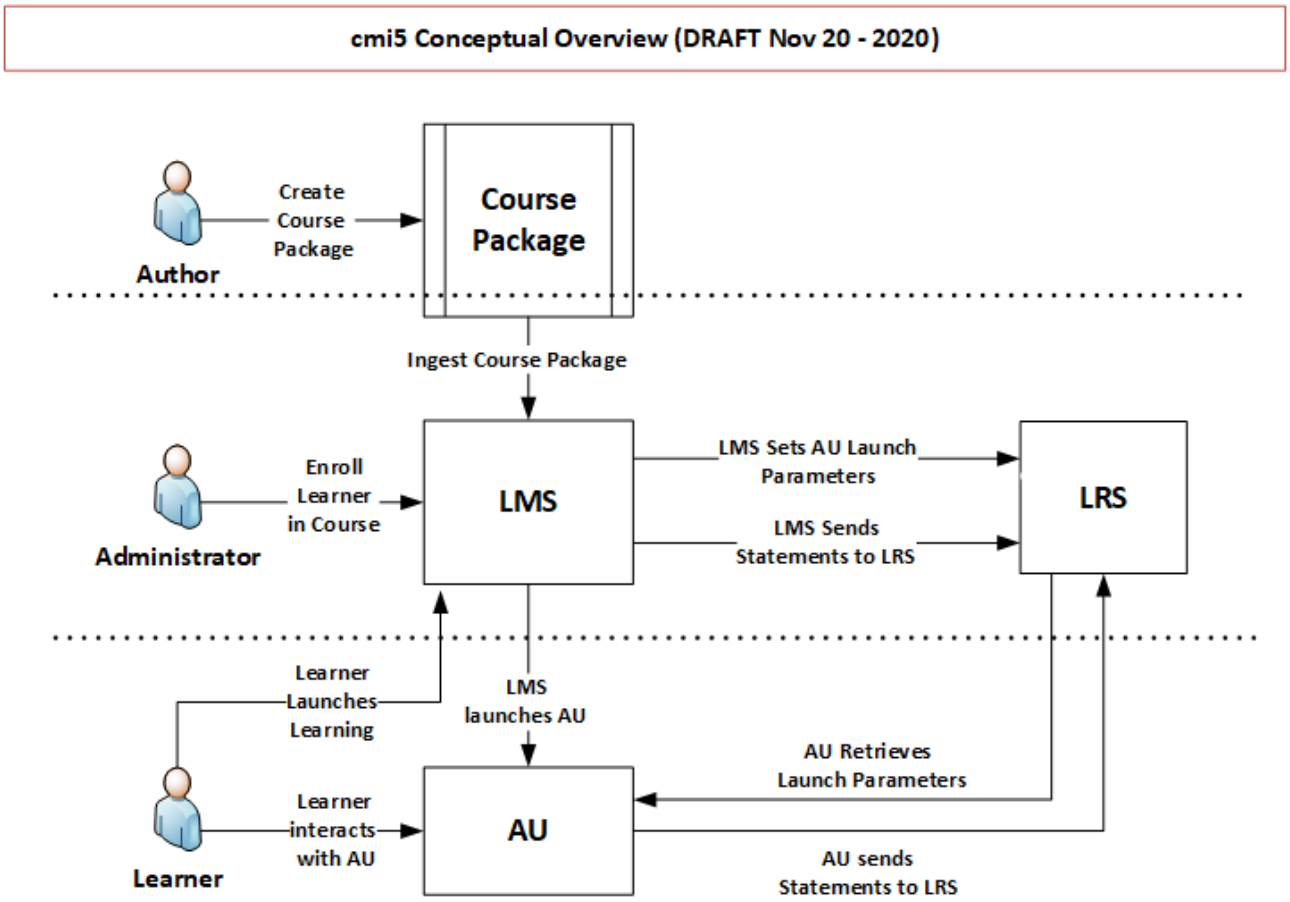


Figure 5: Conceptual Overview of cmi5 [6]

cmi5 also requires Determine Launch Mode, defining some xAPI statements that must appear in correct order. The course itself describes `moveOn` rules

> Setting that captures how a learner moves through the AUs/Blocks of a Course. Determines what is required for an AU to be considered "Satisfied". Blocks are "Satisfied" when all of their direct descendent AUs or Blocks are "Satisfied". The Course is "Satisfied" when all of its direct descendent AUs or Blocks are "Satisfied". [7]

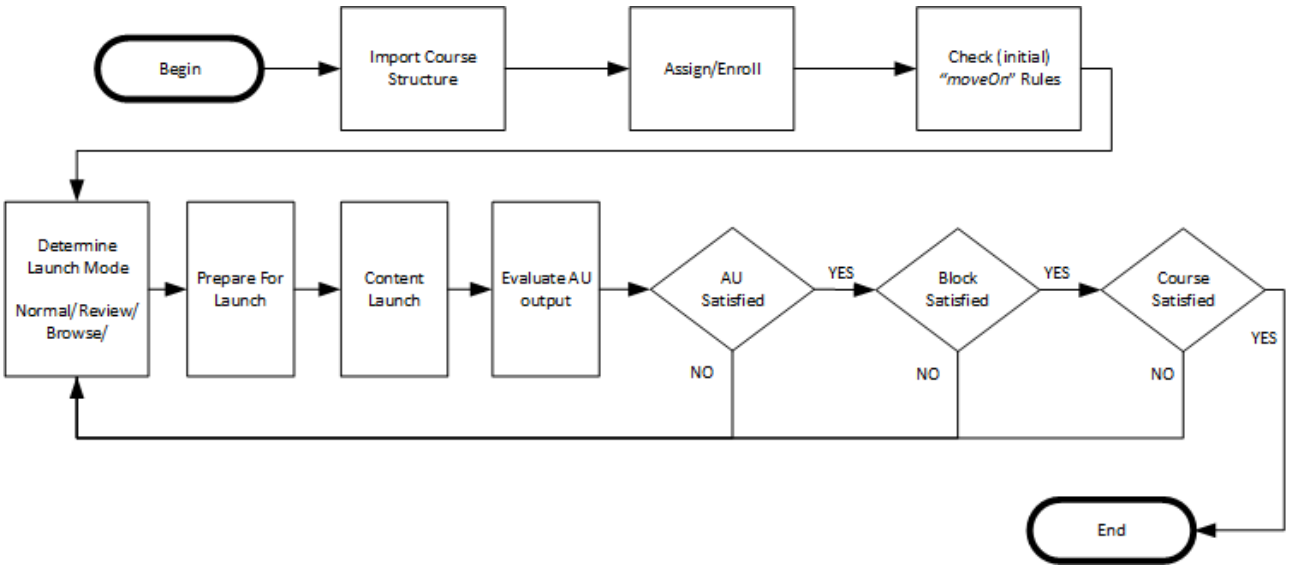


Figure 6: cmi5 Implementation Flow for an LMS [8]

**Limitations parts of current standards**

Regardless of all the efforts for evolution of e-learning standard there are still limitations:

- SCORM is limited by design, there is no way to improve it implementing **Separation of concerns** design pattern.
- Even with latest standard `cmi5` the **Separation of concerns** is not complete.
- Assignable Unit defines only entry URL for the content, but it does not define the content structure in any way.
- Specification of Assignable Unit (AU) require to have `launchURL` that basically is course starting point. There is no way to extend this to replace `launchURL` with the content itself.
- There is no separation of layers in content delivery. Presentation, data and logic layers are inseparable.
- Courses cannot be played offline as server is required all the time.
- Even though cmi5 provides **Mobile app launch support** functionality there is no specification for that, it is possible yet not defined.

[7] cmi5: Technical 101 Terminology.
[8] cmi5 Implementation Flow for an LMS

# Separation of concerns

The main motivation of introducing new e-learning format is to allow to separate all of the e-learning components into independent elements and follow the **separation of concerns (SoC)** computer science design principle.

> A design principle for breaking down an application into modules, layers, and encapsulations, the roles of which are independent of one another. [9]

## Headless

Regular websites and web application works in the way that their own back-end (server side component) and front-end (graphical user interface). Each piece use the same code base and communicate directly on the server machine with each other, making the website as a whole.

Headless web application is an implementation of **separation of concerns (SoC)** design principle of the front-end as stand-alone piece of software, and the back-end that doesn't know anything about way the data that is served will be presented. All the communication happens through API as the bridge between both parts. All parts works separately technically (placed on separate servers) and functionally.

### Headless LMS.

In opposition to Monolith LMS Architecture headless LMS is build upon API as a main component. All other components does communicate though this interface. In most cases API and Database are the only parts that require dedicated server.

In Monolithic architecture, frontend component, a presentation layers, requires specific know-how, example: you are obliged to use Moodle template system called Mustache In Headless architecture, frontend is framework agnostic. You can use any frontend framework you want. Furthermore you can use few at once, like React on one domain, vue for course details and Angular for admin panel on other domains.

**Headless LMS Architecture**

- **separation of concerns (SoC)** design principle, separate all of the components.
- only API require server
- admin panel is serverless
- user app is serverless
- application and admin panel are easy to replicate
- other view layouts (eg native mobile application) are easy to add without changes to other layers
- to implement courses for students there is no need to specialization knowledge.

A Headless LMS is a "Course Repository" that makes content accessible to any platform via an API. We provide blocks to build one, yet you're free to change those od use your own. Unlike a traditional LMS such as Moodle, a Headless LMS does not dictate where or how content is shown. Also you don't need any additional software to show a course - it's just a matter of API communication

[9] Blockchain Networks: Token Design and Management Overview NISTIR 8301. National Institute of Standards and Technology

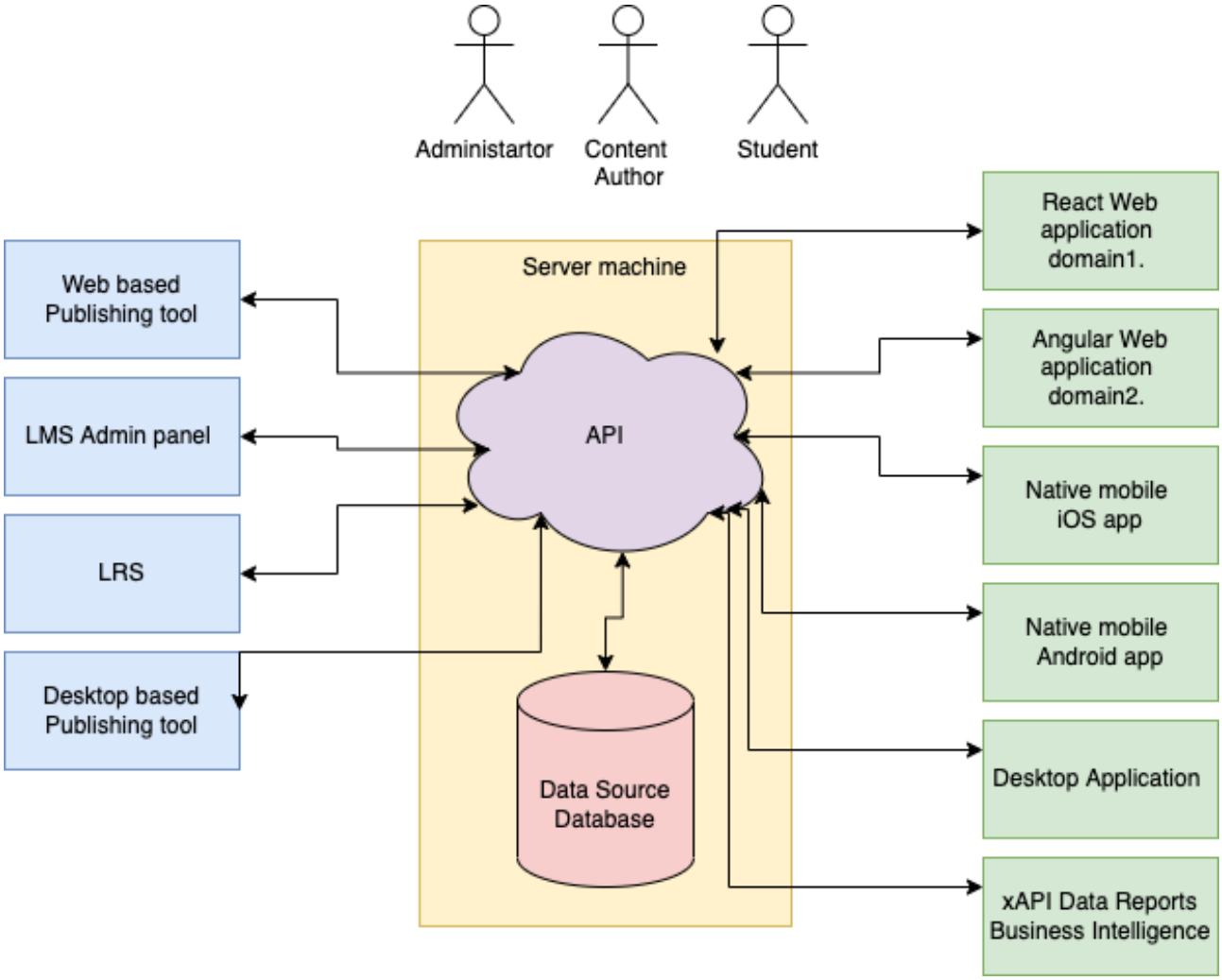


Figure 7: Architecture of headless LMS

A Headless LMS enables teams to deliver omnichannel experiences at scale, globally, without restrictions like templates, devices, or pre-defined technologies. A Headless LMS allows brands and companies to engage with users on any device and format. White label was never easier then with headless. A Headless LMS fits into any preferred tech stack or framework, including most popular ones like React, Angular, and Vue.

### Limitations parts of current standards that Headless can improve

- Implementing **Separation of concerns** design pattern is complete.
- New headless formats can defined content structure.
- Presentation, data and logic layers are separable.
- Courses cannot can be played offline as server is not required all the time.
- Other presentation layouts are easy to add without changes to other layers so **Mobile app launch support** functionality is easily achievable.

## ULAM Format.

Because of the existing limitation new format **ULAM Universal Learning Asynchronous Model** for course content is needed.

The main motivation for the above are the following statements.

**Separation content layer from presentation layer.**

All of the most popular e-learning format, mostly SCORM mix view, data and logic layer into one as inseparable. New format should separate all the layers to follow separation of concerns (SoC) design principle.

**Use of JSON format instead of XML.**

When SCORM was published for the first time JSON format did not exists. During evolution of standards ADL use JSON for `xAPI`. Comparing to XML JSON is light-weight and has an easy to parse data format requiring no additional code for parsing. It is more popular then XML [10]

**Easy implementation.**

Since SCORM first version implementing the format wasn't trivial for implementation. Until this day SCORM is the most popular, and latest format cmi5 has problems with adaptations [11]. New format must be easy to implement.

**Format that age well.**

SCORM does not age well. There are many courses that were created in 2010s using obsolete Flash technologies, which are useless now because none of the browsers still support Flash Player. New format must be able to prevent this kind of issues and **Separation of concerns** design pattern is a solution, because it does separate course pure data from presentation.

**Simple but open for extension.**

New format must be open for extensions which implies that it is possible to create new features without changing old ones.

[10] Google Trends XML vs JSON
[11] An exciting time to watch xAPI and cmi5 adoption numbers

**Standalone.**

New format package must be able to be played as course without any external services.

**Headless.**

The course itself does not know how it will be displayed for a student.

**Using well design standards, reject obsolete ones.**

**From SCORM**

- Content packaging as ZIP file
- Importing from ZIP
- Exporting as ZIP
- Sequencing

**From xAPI**

- usage of JSON Format
- xAPI Statements
- Learning Record Store

**From cmi5**

- Launch Method with `moveOn` Rules
- Manifest with blocks and Assignable Units (AU)
- 9 xAPI Verbs (Launched, Initialized, Completed, Passed, Failed, Abandoned, Waived, Terminated, Satisfied)

## Definition

The course is packed in ZIP file that contains `content.json` manifest in main folder.

The course itself consists of at least one lessons which consists of at least one topic. Topic is describes by type and value.

The manifest contains the following:

### General description of Course attributes

Course is defined by

| Attribute name | type | description | required |
|---|---|---|---|
| id | number \| string | Unique ID of the course. Can be used to identify during import process whether this is a new course or update | no |
| description | string | Description of the course | no |
| title | string | Title of the course | yes |
| base_price | number | Base price of the course. Value 0 means that it's free | no |
| lessons | array | List of lessons | yes |
| language | string | Unique ID of the language of the course. Preferred in ISO 639-1 format | no |
| subtitle | string | Subtitle of the course | no |
| summary | string | Summary | no |

| Attribute name | type | description | required |
|---|---|---|---|
| duration | string | How long does the course take. Preferred in ISO 8601 duration format | no |

Except of listed attributes the course can be described by unlimited additional fields.

**General description of Lesson attributes**

| Attribute name | type | description | required |
|---|---|---|---|
| id | number \| string | Unique ID of the course. Can be used to identify during import process whether this is a new lesson or update | no |
| title | string | Title of the lesson | yes |
| order | number | Sorting order | no |
| duration | string | How long does the lesson take. Preferred in ISO 8601 duration format | no |
| summary | string | Summary of the lesson | no |
| topis | array | List of Topics | yes |

Except of listed attributes the lesson can be described by unlimited additional fields.

**General description of Topic attributes**

| Attribute name | type | description | required |
|---|---|---|---|
| id | number \| string | Unique ID of the course. Can be used to identify during import process whether this is a new topic or update | no |
| title | string | Title of the topic | yes |
| order | number | Sorting order | no |
| duration | string | How long does the lesson take. Preferred in ISO 8601 duration format | no |
| summary | string | Summary of the lesson | no |
| preview | boolean | Can user preview this Assignable Unit without launching the course | no |
| description | string | Description of the lesson | no |
| type | string | Type of Topic | yes |
| value | object | Value of the Topic depending on the Type | yes |

**General description of Topic types**

Topic is describes by type and value. In the first version of the format there are following Topic Types.

Topic type attribute is in string format with namespace, with default namespace `ULAM`.

Topic type is an abstract type and polymorphic. This means that has different meaning based on the `type` value.

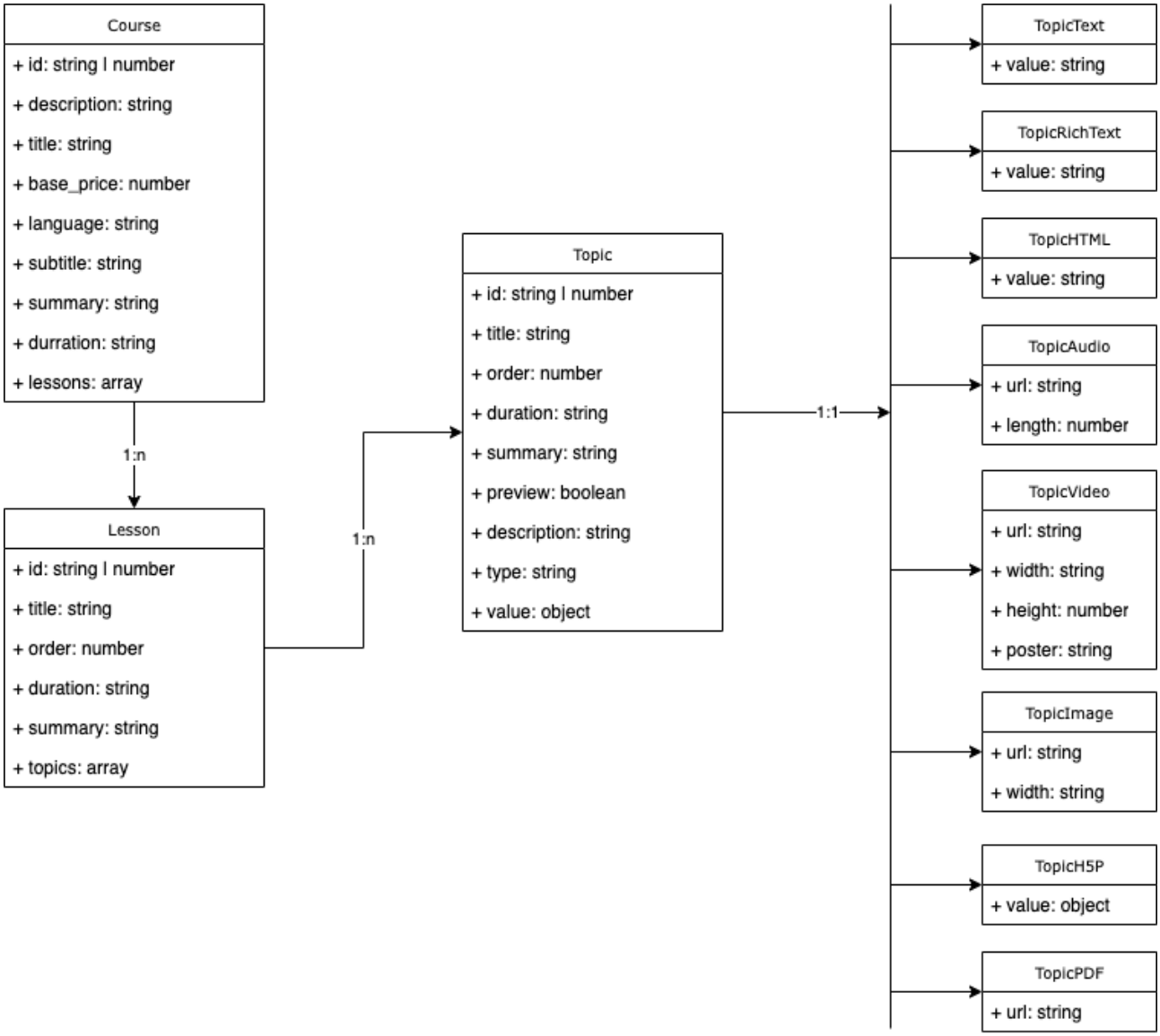


Figure 8: Structure of `ULAM` course manifest

If the value of the Topic contains references (url) to assets (audio, video, images, etc), they must be placed with relative paths to main `content.json` file.

**Text**

- Namespace: `ulam\text`
- Value: Unformatted text with newlines

**RichText**

- Namespace: `ulam\richtext`
- Value: Formatted with Markdown text with newlines

**HTMLText**

- Namespace: `ulam\html`
- Value: Formatted with HTML text

**Audio**

- Namespace: `ulam\audio`
- Value: Reference to audio file

**Video**

- Namespace: `ulam\video`
- Value: Reference to Video file

**PDF**

- Namespace: `ulam\pdf`
- Value: Reference to PDF file

**Image**

- Namespace: `ulam\image`
- Value: Reference to Image file

**H5P**

- Namespace: `ulam\h5p`
- Value: Reference to H5P content

## Validating

The manifest must be validated against JSON Schema file. The latest schema is always available in github repository[12].

[12] Ulam Headless Format Github repository.

## Extending

Except of listed attributes the manifest can be extended by unlimited additional fields.

For example if there is need to describe course by video path or image it's a matter of adding new fields to the course attributes.

### Topic Types

One of the most important motivation introducing new format was the ability to add without limits new type of Assignable Unit (AU) in `ULAM` called TopicTypes.

Each organization that wants to add new type should use it's own namespace for `type`.

This allows to add new topic types like

- Augmented Reality assets
- 3D Models
- 3D Panoramas
- 3D Movies
- Metaverse assets
- Specialized format for particular industries
- etc.

## Example

The example below show a course with extended fields.


```
{
    "id": "f7e84b25e6f1f8d5e7dd3f1f438dd5f5",
    "title": "The title of course.",
    "summary": "Summary.\n this is new line",
    "image_path": "course\/2\/repudiandae.jpg",
    "image_url": "https:\/\/api.escolalms.com\/\/storage\/course\/2\/repudiandae.jpg",
    "video_path": "course\/2\/repudiandae.mp4",
    "video_url": "https:\/\/api.escolalms.com\/\/storage\/course\/2\/repudiandae.mp4",
    "base_price": 0,
    "duration": "12H",
    "active": true,
    "subtitle": "Ratione nulla voluptate consequatur qui atque et rerum.",
    "language": "pl",
    "description": "Rerum numquam ut praesentium nostrum aut officia consequuntur",
    "level": "expert",
    "lessons": [{
        "title": "voluptas",
        "summary": "Qui aliquam aliquam dolor nihil.",
        "topics": [{
            "title": "beatae",
            "active": true,
            "preview": true,
            "type": "ulam\\richtext",
            "value": {
                "created_at": "2021-10-14T15:50:29.000000Z",
                "updated_at": "2021-10-14T15:50:29.000000Z",
```

```
                "value": "Aperiam magni saepe labore accusantium totam animi.\n=========
            },
            "summary": "Sit aut fuga repellendus velit harum esse.",
            "order": 5

        }, {
            "title": "esse",
            "active": true,
            "preview": false,
            "type": "ulam\\image",
            "value": {
                "value": "topic\/23\/doloribus.jpg",
                "width": 640,
                "height": 480,
                "url": "https:\/\/api.escolalms.com\/\/storage\/topic\/23\/doloribus.jpg
            },
            "summary": "Ipsa laboriosam.",
        }]
}
```


More examples are available in github repository[13].

## Packaging.

Ulam package is similar to SCORM Package. It is zip file that consist of:

- validated `content.json` in main folder
- all the other assets that `content.json` attributes refer to. Those are relative paths

### Import

LMS should be able to parse ZIP package, copy essential files, update theirs paths and save course, lessons and topics to database.

### Export

LMS should be able to create ZIP package, with essential files, update theirs paths and save course, lessons and topics to `content.json`

## comparison with `cmi5`

| | cmi5 | ULAM |
|---|---|---|
| use xAPI | yes | yes |
| Manifest format | xml | json |
| Defined course type structure | no | yes |
| Separation of concerns in course data | no | yes |
| Connection to LRS | required | not required |
| Mobile friendly | no (only tracking) | yes |
| Run-Time required | yes | no |

[13] Ulam Headless Format Github repository.

| | cmi5 | ULAM |
|---|---|---|
| Content Package format | yes | yes |
| Definition of Course launch | yes | yes, same as cmi5 |
| Client Agnostic | yes | yes |
| Distributed Content | yes | no |
| Advanced activity tracking | yes | yes |
| Serverless | no | yes |

## Frontent. Implementation

The implementation of ULAM courses are Frontend agnostinc. This means that the format itself doesn't describe how to render the course.

The player should follow `cmi5` lanuch mode and all `xAPI` 9 verbs.

Once the frontend application receice the ULAM content from the LMS endpoint is should parse it and create tree of lessons and topics.

### Single Page Application

Modern web application does not require server side component to render its content, since evolution of JavaScript all of rendering is possible only in browser on the client side.

A Single-page application (SPA) is a web app implementation that loads only a single web document, and then updates the body content of that single document via JavaScript API, which results in websites without loading whole new pages from the server, that finally result in performance gains and a more dynamic experience. Those kind off appliaction can load more quickly, fetching data in the background. Support of incremental updates, saving partially completed forms or documents without the user having to click a button to submit a form and wait for the whole page to reload. Also support of rich frontend behaviors, such as drag-and-drop, copy and paste, file picker, local files preview and editing and many more. Those kind of application can run in a disconnected mode, making updates to a client-side model that are eventually synchronized back to the server once a connection is re-established, which sometimes is names offline mode.

The client-side implementation of ULAM format can be Single Page Application becase the format itself just defines content data, leaving presentation model up to the implementation phase. The SPA would handle a logic and presentation layer.

### Progressive Web App

Progressive Web Apps are web apps that use modern web technologies like service workers, manifests, in combination with progressive enhancement to give users an experience on par with native apps. They provide a number of advantages to users like the following

- **being installable** - users to have app icons on their home screen, and be able to tap to open apps into their own native container that feels nicely integrated with the underlying platform.
- **progressively enhanced** - provide an excellent experience to fully capable devices, and an acceptable experience to less capable browsers.
- **responsively designed** - appliaction User Interfaces will fit any form factor: desktop, mobile, tablet, or whatever comes next.

- **re-engageable** - ease with which users can be re-engaged by updates and new content, even when they aren't looking at the app or using their devices, eg Push Notifications.
- **linkable** - ability to link to an app at a specific URL without the need for an app store or complex installation proces
- **discoverable** - better representation in search engines, be easier to expose, catalog and rank, and have metadata usable by browsers to give them special capabilities. Also beong able to be installed be Operation System package managers and shops, eg Android Play.
- **network independent** - can work when the network is unreliable, or even non-existent
- **secure** - working in browser sandbox and takich take advantage of HTTPS make them secure.

The client-side implementation of ULAM format fits all capabilities to be a Progressive Web App without any special preparation.

### Topics Types

`topic_type` is like `xAPI` word, it can be anything, the standard doesn't specify this except of the reserver `ulam` namespace.

### Topics Types Content Players

Once course is rendered, frontend should parse each topic type and display content depending on the type.

The course itself is headless so it doesn't know what kind of enviroment and device it is being played on.

Let take a video topic type as and example:

```
{
  "title": "esse",
  "type": "ulam\\video",
  "value": {
    "value": "topic/23/doloribus.mp4",
    "width": 640,
    "height": 480,
    "url": "https://api.escolalms.com//storage/topic/23/doloribus.mp4"
  },
  "summary": "Ipsa laboriosam."
}
```

The snippet above taken from ULAM format means that this topic type is a video that has a dimention 640 width and 480 height.

There are many scenarios that this can be handled, as this depenend on what is the course current context. Some examples include:

- HTML5 build-in video player (tag )
- iOS native application `AVPlayer`
- React/Vue/Angular video component
- React Native video component.

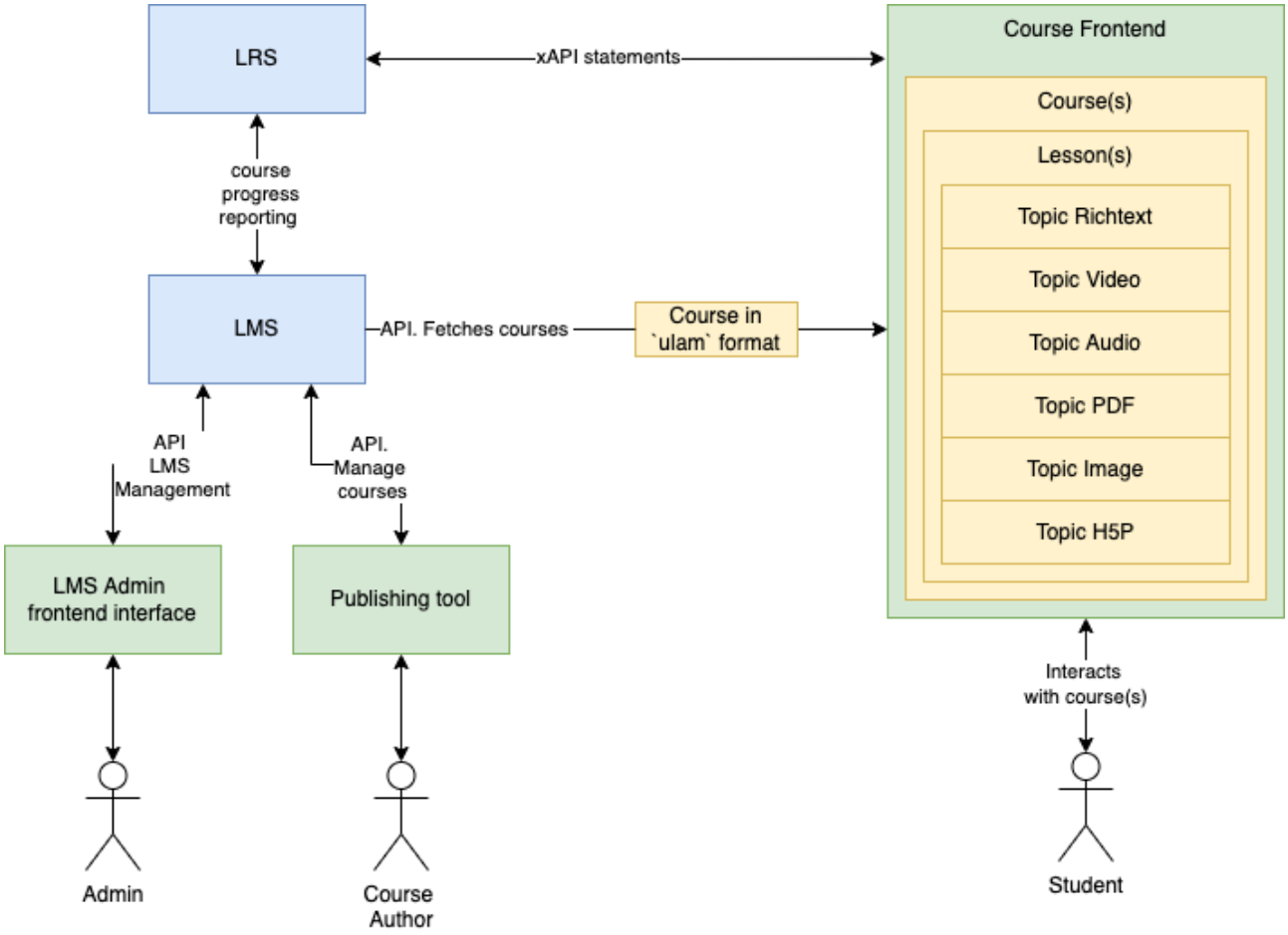


Figure 9: Implementation of Courses with `ULAM` format

# The Conclusions

The new format should be able top solve issue that occured during common e-learning lifespam. ULAM format is easy to use, implement, does age well and it's designed for extesions and plugins.

## Future Work

The main focus on the ULAM format is in WELLS LMS system designed and implement by EsolaLMS Ltd. The project is open source and available on github[14].

At the moment there is no versioning of ULAM format as this is still proof of concept.

## The Acknowledgements

Thank you XXX who edited this article.

Name ULAM comes from Stanislaw Ulam a Polish scientist in the fields of mathematics and nuclear physics, who worked with von John von Neumann on very first computer based computation methods.

[14] Wellms LMS Github repositories.