Moodle
Conceptual Architecture, Concrete Architecture and Architecture Enhancement Proposal
2019-12-09

Authors:

Chengxu Li - 17cl10@queensu.ca
Henry Van Herk - 15hgvh@queensu.ca
Runze Yi - 17ry5@queensu.ca
Tony Chen - 17gc1@queensu.ca
Tyler Elliott - tyler.elliott@queensu.ca

Contributions:

Chengxu Li - Context View, Abstract, Concrete System Architecture, Architecture styles, Enhancement Proposal description and how current architecture can support it

Henry Van Herk - Purpose and Scope, Audience, Architectural Views, Introduction, System Scenarios, Enhancement Proposal description and how current architecture can support it

Runze Yi - Introduction, Quality properties, Concrete System Architecture, Architecture styles, Conclusion, How current architecture can support Enhancement Proposal

Tony Chen - Stakeholders and Requirements, Why system is architecturally interesting, Functional View of the Top-Level and Subsystem, Perspectives, Sequence Diagrams, Enhancement Proposal UML Sequence Diagrams

Tyler Elliott - Stakeholders and Requirements, Why system is architecturally interesting, Functional View of the Top-Level and Subsystem, Perspectives, Sequence Diagrams, Enhancement Proposal UML Use Case Diagrams, Functional and Quality Requirements

Abstract:

This report is a discussion of the architecture of Moodle and some interesting aspects of the system. We will discuss things like the purpose, scope and audience of Moodle as well as functional and non-functional requirements for the system through text and diagrams.

The purpose of Moodle is to provide a free, convenient educational platform for both students and instructors. Its scope has been discussed within a UML diagram on the eleventh page. The main audience consists of students and teachers, with teachers designing their course content and distributing materials through Moodle to students.

This report covers important Moodle functional and quality requirements. The concrete architecture is also analyzed to provide in-depth information about architectural scenarios, viewpoints, and perspectives. While the current architecture provides many features, there are always new improvements being made, and we have proposed an enhancement to the architecture in this report. It was discovered that Moodle uses a plug-in based architectural style. Important use-cases have also been included and visualized using UML sequence diagrams.

Introduction

The purpose of this report is to describe the conceptual architecture of Moodle, which is an open source Learning Management System (LMS). It is a scalable system that allows external plugins, and is highly accessible.

First, we have an overall analysis of the system, which will focus on the purpose and scope, the audience, why it is architecturally interesting and its quality properties. The purpose and scope explains why the system exists, its functionalities and limitations and its business goals and architectural scope. The audience establishes Moodle's intended users and gives a brief overview of the stakeholders. The reason Moodle is architecturally interesting and why it is chosen as the target of this report will be identified. Lastly we will go through the software quality attributes with contributed to Moodle's overall success. An analysis on the top-level system and an interesting subsystem is analyzed. A dependency diagram is extracted from the Understand tool. The overall architectural style of Moodle is also studied as a plug-in architectural style. Examples of plug-ins are also provided.

Next, we go deeper into the individual roles of Moodle's most important stakeholders, such as the developers and system administrators, and provide an overview of the concerns of the stakeholders which are presented as requirements. These are broken down into functional requirements, for example being able to connect to online lessons, and non-functional requirements, such as availability, as the system shall be able to carry out a task when needed and to recover from failures. These are all presented with shall statements, and a use case diagram is provided for the functional requirements. Additionally, we have multiple functional scenarios and quality scenarios that apply to the Moodle system, and reflect the identified functional and non-functional requirements.

Afterwards, we gathered the architectural views and have presented them as viewpoints and analyzed their individual importance to the system as different perspectives. We used a table to organize the importance of these perspectives, followed by a brief explanation on the choice of the importance level for perspective as they apply to each viewpoint. Then we analyzed the context views of the system to figure out it's main components and architectural elements. After a quick overview we gathered into them into a context diagram.

Finally, we have proposed an enhancement to Moodle's current architecture. The value and benefits of the proposed enhancement to Moodle's architecture are identified. The functional requirements and corresponding non-functional requirements are presented with the aid of UML use case diagrams and textual descriptions. We then explain how the current architecture can support the enhancement using existing components and interfaces. Lastly, sequence diagrams we have created show the flow of the enhancement's use case diagrams.

Through this process we have obtained a general overview of the system and the system requirements that inform the architecture. For other developers or people interested in open source projects, it is very interesting to study such a well established and stable project, and we hope this analysis can help with that.

System Overview

a. Purpose and Scope

Moodle is an open source web application Learning Management System. Originally developed by Martin Dougiamas in 1999 and initially released in 2002, the project is now operated under Moodle HQ, an Australian company which is financially supported by a network of eighty-four Moodle Partner service companies worldwide (1.a.1). The mission statement of the Moodle project is “Empowering educators to improve our world.” (1.a.2). Used in education, training, and development, as well as in business settings, the purpose of Moodle is to provide educators and students with the best tools available to teach and learn, with a focus on interactive and collaborative content. Their vision statement is “To give the world the most effective platform for learning.” (1.a.3) To date, Moodle powers tens of thousands of learning environments in 231 countries, and currently has almost 200 000 000 total users, making it the most widely used learning platform (1.a.4). To clearly define the scope of Moodle, the business goals and drivers are when Martin Dougiamas started a PhD to examine "the use of open source software to support a social constructionist epistemology of teaching and learning within Internet-based communities of reflective inquiry." (1.a.5). The architectural scope is the system handles account management, course content, enrollment and access control, differentiates between students and teachers, supports external payment methods, and meets the needs of educational organizations. The architectural concerns are to provide an all in one LMS that is easy to use, completely free, fully customizable, scalable, secure and is accessible from remotely anywhere on any computer or mobile device. The architectural principles are that the system must be accessible and scalable to groups exceeding one million users. Lastly, other architectural constraints are meeting the GNU General Public License so that it can be provided as open-source.

b. Audience

Moodle’s audience is educators, students and educational organizations. The project is open source, which means “anyone can adapt, extend or modify Moodle for both commercial and non-commercial projects without any licensing fees and benefit from the cost-efficiencies, flexibility and other advantages of using Moodle.” (1.b.1). This also allows individual developers to contribute to the software by creating plugins and integrating external applications to achieve specific tasks, which users can pick and choose to tailor their Moodle experience. As of October 2019, there are 1,619 plugins available for download to extend Moodle’s core functionality (1.b.2). It is also a web application, meaning that it is executed through a web browser as opposed to traditional software which is executed by a user’s operating system. Since it is web based, it can be accessed from virtually anywhere, and is also compatible with mobile devices. Additionally, Moodle is available in over 120 languages (1.b.3), and this number continues to grow as users in the community contribute in the translation process.

c. Why the System is Architecturally Interesting

There are many reasons we find Moodle architecturally interesting, most notably its highly modular organization which enables the development and use of external plugins. Furthermore, its scalability and multilingualism allows it to be used in groups ranging in size from dozens of users to over a million, in 231 countries around the world. Its foundation on open-source development and the fact that it is free makes it a product that everyone can use and expand. By building on a common theme of accessibility and using assistive

technologies, such as screen-magnifiers, it can accommodate all users, including those with disabilities. Finally, Moodle's architecture allows easy integration with existing academic systems, encouraging established schools to use it to modernize their teaching practices.

d. Quality Properties

All features are established using help from lecture notes[2.d.1].

Performance:

As an e-learning tool, Moodle needs to be able to respond fast enough for both educators' and students' needs, as well as for anyone else that might use the system. When the educator posts materials or makes changes to existing content, Moodle should quickly respond to that and complete the educator's task. Users should not, and will not, wait for a very long time to use the system. Otherwise the waste of time would make the system very inefficient and users unhappy, and it would eventually be retired or abandoned.

Security:

Moodle targets schools, universities, workplaces and other sectors that need a platform for individuals to learn online. At least a large portion of the materials posted on Moodle for that sector will be copyrighted, and should not be shared with anyone outside of that course. For the students, or learners, their personal information including the account and password should be protected. If a leak of accounts ever happened, someone else will have access to information that is critical to the sector the learner is in. Moodle must provide a secure environment for both educators and learners such that their information will not be spread or seen by others.

Safety:

Safety as a software quality attributes means the ability of a system to avoid hazardous situations and act when components fail. This means Moodle should be able to detect and avoid critical failures in the system, while having an executable plan to act upon when system failures occur. As mentioned earlier, Moodle is a platform that works with multiple sectors and different languages. Situations like a collapsed database or broken interface would result in catastrophic results for both Moodle and the users of it. The users might have all their schedule delayed, even abandoned, and Moodle will be the one responsible for that. Despite the money it takes to fix the system after failure, the reputation of Moodle will certainly fall.

Availability:

As an online learning management tool, Moodle needs to be available all the time. The costs of service downtime for Moodle might not be as significant as for other large scale system, and it would definitely cost time of both educators and learners that are expecting certain functionality from the system at that time. For example, sudden changes to the course plan, task submissions, online tests, etc. All of these will be denied while service downtime, and there will be critical consequences. Additionally, the system should be available on different platforms such as Windows, Mac, mobile phones, tablets, etc. As a system of such

scale, failing to deliver such properties will result in losing potential users and negative feedback of current users.

Modifiability:

As time progresses, new features will be added to learning strategies. Moodle needs to be able to adapt to changes of the contents it provides. For example, a new operating system is developed and suddenly get as many users as Microsoft and Apple. In this case Moodle will need to offer their service on that OS as well. Therefore new features should be added to Moodle according to the needs of users. When failing to deliver such properties, the system will lose advantage to other e-learning tools that are of option to Moodle's current users.

Usability:

Moodle as a system that targets multiple sectors and provides e-learning to them, should be easy to use and learn. A user should not spend weeks just to figure out how the system operates and how he/she should view the slides. Nor should the user go through hundreds of links to finally log in. These are exaggerated but do carry the essential property that Moodle should have. Failing to deliver such properties will result in inefficient system along with negative user feedback, and eventually Moodle will lose existing or potential users.

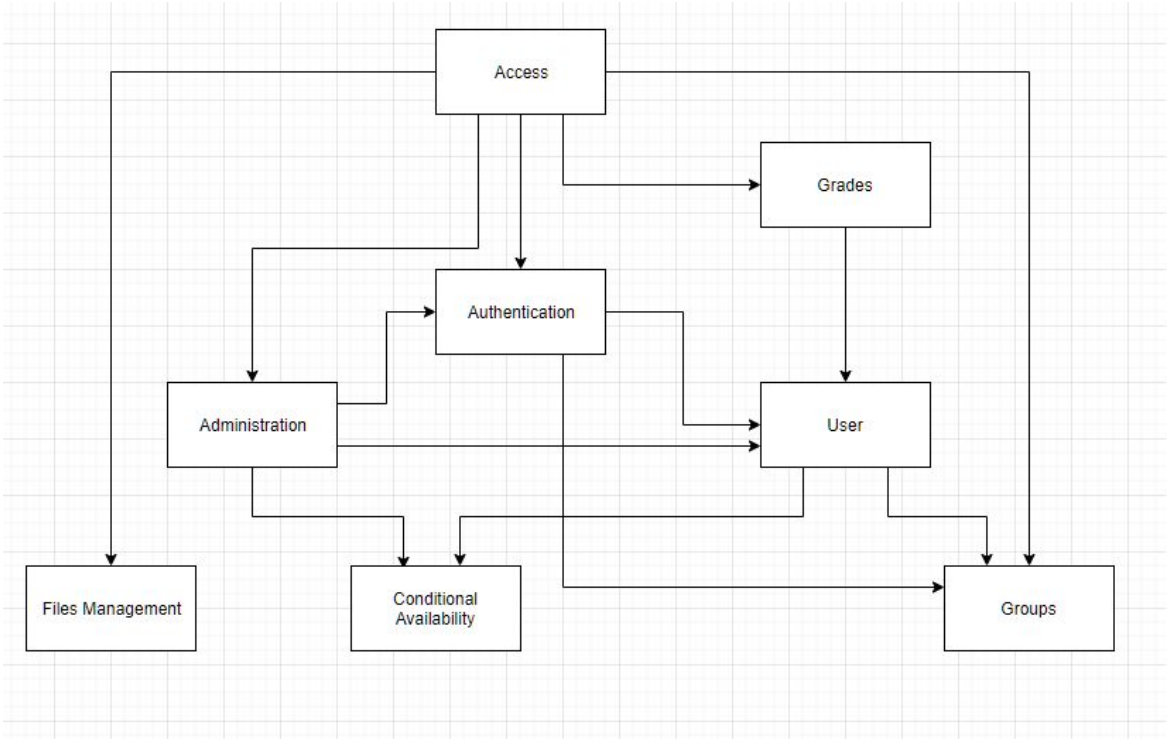
Interoperability:

Moodle is able to interact with systems of its users, for example, the management system of a university, and provide functionality to them. This allows users from different entities to make use of desired features according to the needs of that entity, and would also allow Moodle to manage users form that entity as a whole, and address their needs. An example of this is to log in to Moodle with a Facebook account. Failing to deliver such properties will result in losing potential users and dissatisfaction of current users.

e.1. Top-Level System:

The following are based on the Moodle.rog development page[2.e.1] and Understand Tool[2.e.2].

Dependencies Graph:



Top Level API Intro:

Access API: This API provide Moodle with functions that are capable of determining what current user is allowed to access, it also allows modules to extend Moodle with new capabilities. This subsystem contains Access, Login, Course, Enrol, Admin and User.

Data Manipulation API: This API aims to provide Moodle with functions of managing data in the database, functions should be used EXCLUSIVELY.

Exporter API: This API ensures that all data exported can be easily maintained. Also. They are used to generate signatures of external functions.

File API: This API manages all the files that are stored and owned by Moodle.

Form API: Web Forms in Moodle are created using Forms API, it supports all HTML elements.

Navigation API: This API functions in the way of manipulating the navigation system used in Moodle and such also all of the navigation activities from users.

Page API: This API allows developers to develop the pages you are seeing now in Moodle.

Persistent API: This API is responsible for updating / deleting / creating every object in the database of Moodle.

String API: This API handles all the text users might see in the Moodle interface. It also provides general string functions you might see in high-level programming languages like substr, strlen ect.

e.2. Sub system--Access

The access subsystem takes care of authorization to users to be able to view the content and make use of materials in certain areas.

A role based access control model is used. Role is a set of capability definitions, each usually represent the ability of a user to do a certain thing, including accessing a file or enrolling in a course. All users that has not logged in yet will be defined as $CFG->notloggedinroleid, and no other id will be able to be assigned to such users. There is also a $CDF->guestroleid, which is for guest log in accounts. Similarly, no other role can be assigned to guest accounts. All others users that have logged in will receive their default role id according to their authentication.

There are several factors inside the sub system. Access, Login, Course,Enrol, Admin and User. All of these are elements of the whole Moodle program. Access will require certain functionally from other elements to check and verify the capability of the current user.

f. Architectural Style

The following is based on lecture notes on Architectural Styles and Moodle.org plugin directory.

The architectural style that Moodle used is Plug-in. This architectural style works based on the existence of a core system and several independent plug-ins which extends the core system.

The core system is a high level abstract system that defines the operation of the system and the basic business logic. It has no specific implementation, no customization and knows nothing about the actual implementation of the plug-ins. The core just needs knowledge of the existences of the plug-ins and their required inputs and outputs so it can manage that.

The plug-ins are stand-alone, independent components that contain specialized processing, additional features, and custom code that is meant to enhance or extend the core system. Since each plug-in is independent of each other, it allows the system to add, remove or modify the plug-ins very quickly and without having to consider further impact it has on other plug-ins.

The current number of plug-ins that Moodle has is 1626, including Commander/quick navigation, Quiz fault tolerant model, Multi-Langauge content, etc. Some of these plug-ins provide essential functionality to the system while others provide more quality-based features

Stakeholders and Requirements

a. Stakeholders

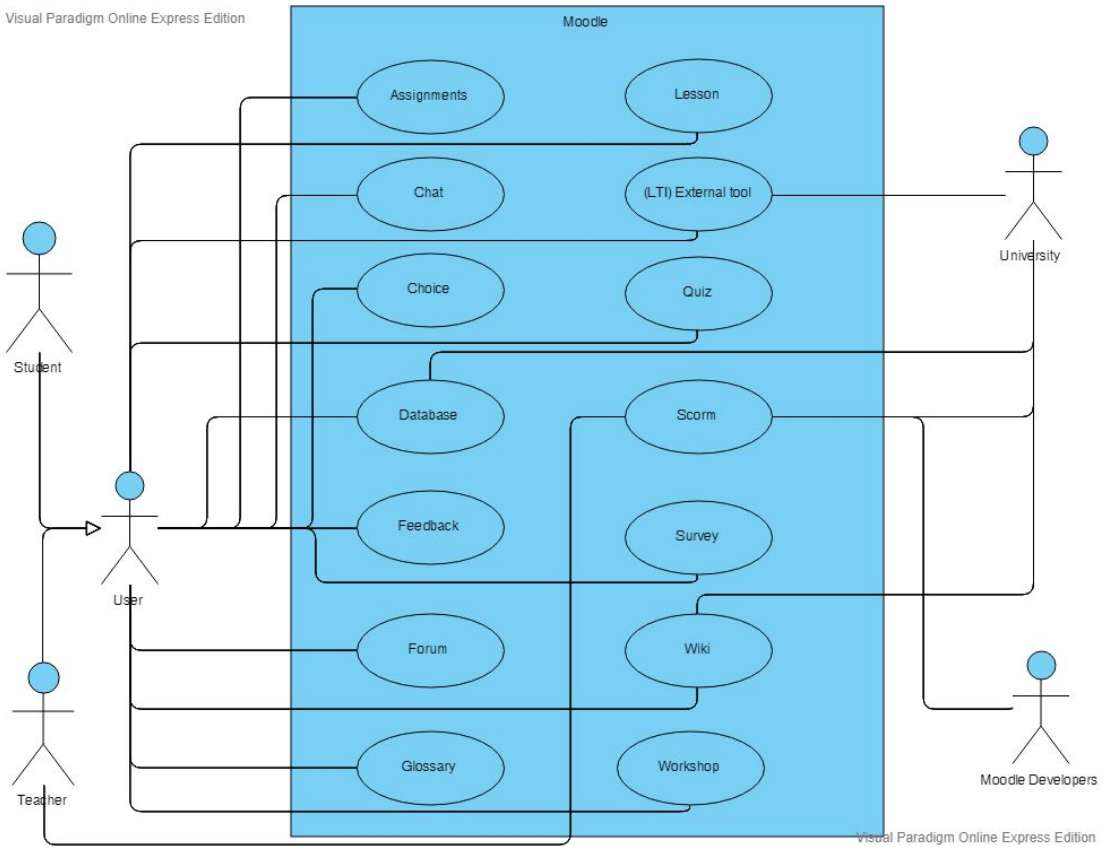
We used all relevant stakeholder categories from the lectures, including these additional types: competitors, and investors. [3.a.1]

Acquirers	Moodle HQ leads and coordinates the Moodle Project. It was originally written by Martin Dougiamas - Founder and CEO of Moodle Pty Ltd. [3.a.2]
Assessors	Ensure compliance with GNU General Public License (GPL). [3.a.3]
Communicators	Different teams of developers write the documentation for Moodle. Plugin developers write their own, and Martin Dougiamas provides overall guidance on architecture. [3.a.2]
Competitors	Blackboard, Canvas, Desire2Learn, Edmodo, and Google Classroom.
Developers	Moodle HQ is an Australian company with more than 20 developers. It is also developed by the open-source programming community through modules. [3.a.2]
Investors	Financially supported by a network of eighty-four Moodle Partner service companies worldwide. The Moodle Partners offer paid services to make it easier to use the free software. [3.a.4]
Maintainers	Moodle HQ developers and open source community work together with educators and users to determine the highest priority items. These decisions are voted on and then implemented to correct errors or improve Moodle. [3.a.5]
Suppliers	Moodle runs without modification on Unix, Linux, FreeBSD, Windows, OS X, NetWare and any other systems that support PHP and a database, including web host providers. [3.a.6]
Support staff	Teaching assistants, discussion forum moderators.
System administrators	IT departments of the schools, governments, and companies that use Moodle.

Testers	Volunteer testers from the Moodle community test each feature in Moodle to see if it works in the current version. These tests are repeated in a series of cycles, usually 4 weeks before a major release. Developers should be writing their own unit tests and/or behavioural tests for all submitted patches. These issues are then tested by Moodle HQ developers (i.e. by a human) on Wednesdays. [3.a.7]
Users	Educators, students, and the organization's IT department.

b. Overview of Requirements

Functional requirements



The following points are based on the Moodle activities page. [3.b.1]

1. Assignments - The system shall allow students to submit assignments and teachers to grade them.
2. Chat - The system shall allow real-time synchronous discussion between all users.
3. Choice - The system shall allow teachers to ask a question and specify a choice of multiple responses, students should be able to select a response.
4. Database - The system shall allow teachers and students to create a bank of record entries, and allow the university to make common banks of entries.

5. Feedback - The system shall allow teachers to create surveys in order to collect feedback, and allow students to respond to surveys and provide feedback.
6. Forum - The system shall allow synchronous discussion between all users.
7. Glossary - The system shall allow users to create and maintain a list of definitions
8. Lesson - The system shall allow teachers to upload lesson plans and allow students to view and download those lessons.
9. (LTI) External Tool - The system shall allow teachers to create external tool activities, or for university administrators to create them for teachers, and then allow users to interact with these LTI compatible tools.
10. Quiz - The system shall allow teachers to design and build quizzes using different question types, and allow students to submit answers which then may be automatically graded.
11. Scorm - The system shall allow the teacher to upload any scorm packages to include in the course, the university to upload common packages and the moodle developers to create new packages
12. Survey - The system shall allow teachers to gather data from students to help them learn about their class, reflect on their teaching and potentially make improvements
13. Wiki - The system shall allow users to create, view, and edit wiki pages, as well as allowing the university to create common wikis.
14. Workshop - The system shall enable peer assessment by allowing students to upload their work, and teachers to upload examples of good work.

Non- functional requirements (Architectural Requirements)

The following points are based on Lecture 4 notes. [3.b.2]

Type	Description
Availability	System shall be able to carry out a task when needed and to recover from failures. Matches all functional requirements.
Capacity	System shall handle over 1 000 000 users while maintaining its performance objectives. Matches 2, 4, 6, 10 from functional requirements.
Interface	System shall have an easy to follow and visually appealing interface. Matches all functional requirements except for 11.
Modifiability	System shall allow enhancement of the software over time by fixing issues and adding features. Matches 9 and 11 from functional requirements.
Performance	System shall meet timing requirements and respond quickly to events. Matches all functional requirements.
Reliability	System shall be reliable and robust, allowing students and teachers to complete their tasks whenever required. Matches all functional requirements.

Safety	System shall avoid hazardous situations and act when components fail. Matches all functional requirements.
Security	System shall protect information such as usernames and passwords from unauthorized access. Matches all functional requirements except for 9 and 11.
Supportability	System shall be supported by all operating systems. Matches all functional requirements.
Testability	System shall be easily testable and have tests for all code, due to allowing open-source contributions. Matches 9 and 11 from functional requirements.
Usability	System shall cater to users of all ability levels and be easy to pick up. Matches all functional requirements except for 11.

c. System Scenarios

Quality Based Scenarios

1. Security

Overview: A user attempts to login to a Moodle system but login fails due to an unrecognized username, an incorrect password or due to system unavailability.

System / Environment State: System is working correctly under normal load; user has initiated an authentication for their login credentials to the Moodle system.

External Stimulus: A user fails to login to the Moodle system.

Required System Behaviour: The login authentication is unrecognized, and the user is not granted access to the Moodle System. The user is prompted to try again or given the option to reset their password if they are an existing user.

Response Measure: The user is not logged in to the Moodle system, after multiple failed attempts to login the user is forced to reset their password.

2. Availability

Overview: One of the user-facing web servers fails during transmission of user page update.

System / Environment State: System is working correctly under normal load; user has requested access to system database which was routed to web server □ in transaction pool.

External Stimulus: Web server □ crashes during response generation.

Required System Behaviour: Response page may be corrupted on client browser. System records web server □ failure and notifies the web server manager. Web server □ is removed from the pool of available servers until it is deemed operational again. The requested page reload is directed to an alternate server and page will be correctly displayed.

Response Measure: Time for re-routed refresh is equivalent to “standard” refresh (<1 second 95% of the time). [4.1]

3. Performance

Overview: Check system responsiveness for grading submitted tests under normal operating conditions.

System / Environment State: System is working correctly under normal load.

External Stimulus: User submits test to be graded.

Required System Behaviour: System cross checks user submission with solution to test and calculates corresponding grade which is stored in the database. Web page is refreshed, and the user’s grade is displayed on the screen.

Response Measure: In 95% of requests, web page is loaded and displayed to user within 1 second. In 99.9% of requests, web page is loaded and displayed to user within 5 seconds. [4.2]

Functional Scenarios

1. Database [4.1]

Overview: How the Moodle system deals with a change to the existing database.

System State: The Moodle system database’s have enough space to cope with the processing required for this change.

System Environment: The deployment environment is operating normally, without problems.

External Stimulus: A user or administrator makes an update to the existing Moodle system database.

Required System Response: The incoming update to the database should be handled immediately, and the old data should stay available until the update is complete.

2. Forum

Overview: How the Moodle system handles synchronous discussion between all users.

System State: The Moodle system has all previous forum posts stored and available for authenticated users.

System Environment: The deployment environment is operating normally, without problems.

External Stimulus: A user or administrator makes a discussion post in one of the Moodle system forums.

Required System Response: The incoming update to the forum should be handled immediately. Active users in a forum should be notified when other users have made a new post in the forum.

3. Quiz

Overview: How the Moodle system handles quiz submission.

System State: The Moodle system has a quiz with solutions created by a teacher in the system that is available for students to complete and submit for evaluation.

System Environment: The deployment environment is operating normally, without problems.

External Stimulus: A student submits a quiz for grading.

Required System Response: The Moodle system cross checks user submission with provided solutions to test and calculates corresponding grade which is stored in the database. Web page is refreshed, and the user's grade is displayed on the screen.

Architectural Views

a. Context View

Brief Description of Included Elements

1. Account management – Information storage, Login verification
This element is commonly used in any software with the concept of "Accounts".
It stores personal information of users, such as profile photo, name, user ID, password and so on.

2. Course content – Course materials storage, Assessments, Searching index
This section is designed for the management of all course materials, assessments and the search index of those courses.

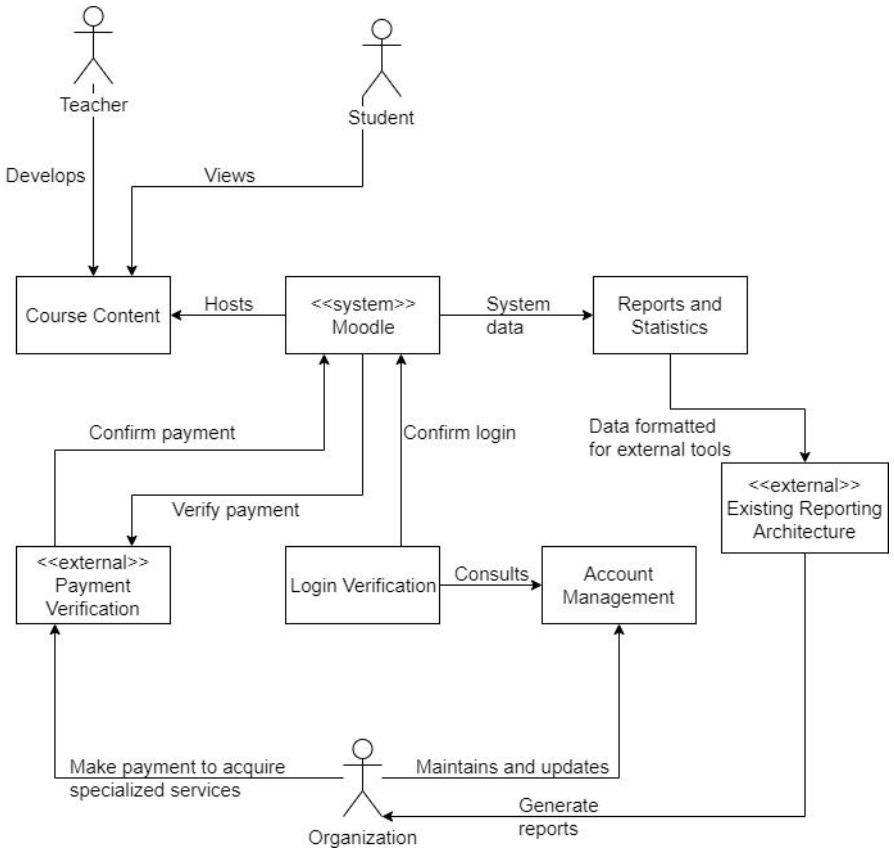
3. Enrollment and access control
This element verifies the access of specific users to certain courses. Hence it has to interact with Account management and Payment verification.

4. Students
Students use the course provided by teachers through Moodle.org to acquire the knowledge they desire, and sometimes they will be assessed.

5. Teachers
Teachers are responsible for designing the course content and walking the student through the course materials. They can also control students' access of course content.

6. External – payment verification
This element is used when a user is purchasing a course. If the payment information is confirmed, the purchase would be successful. Then the user gains access to the purchased course.

7. Organizations with educational needs
For example, if Queen's University would like to use Moodle.org, Queen's University has to provide the verification method of Queen's students.



b. Overview of Viewpoints and Perspectives

Perspectives				
Views	**Security**	**Performance and Scalability**	**Availability and Resilience**	**Evolution**
Functional	Low	Medium	Low	High
Information	Medium	Medium	Low	High
Concurrency	Low	High	Medium	Medium
Development	Medium	Medium	Low	Medium
Deployment	Medium	High	High	Low
Operational	Medium	High	High	Low

Above is a table we have created showing the view and perspective applicability for the Moodle project. The perspectives we have chosen to apply to our views (in no particular order) are security, performance and scalability, availability and resilience, and evolution, because they are the most relevant to the needs of the stakeholders.

The first perspective is security, and its desired quality is defined as "the ability of the system to reliably control, monitor, and audit who can perform what actions on what resources and to detect and recover from failures in security mechanisms." (6.1) Moodle has to identify and authenticate its users and manage who has access and can edit data in a given learning environment. Examining the fundamental organization viewpoints, which are the functional viewpoint, the information viewpoint and the concurrency viewpoint, there isn't a big demand for security. Since Moodle is open source, how the functional elements interact, how data is handled and how concurrent process are coordinated and controlled is available for everyone to see if they choose, and to modify as they see fit. However, sensitive information, such as copyrighted material must be handled with more care. It's when we get into the live environment viewpoints, which are the development viewpoint, the deployment viewpoint and the operational viewpoint, where security is a bigger issue, because we have to regulate who has access to the ability to modify a given learning environment.

The second perspective is performance and scalability, and it's desired quality is defined as "the ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes."(6.2) From the Development viewpoint, performance and scalability aren't the highest concern when building, but are important in maintaining the system. For the functional and information viewpoints, it's important that Moodle is able to efficiently handle the storage, manipulation and distribution of information, which rely on the functional elements in place, as users on the system increases and the performance must stay sufficient. However, what is most important in terms of performance and scalability are the concurrency, deployment and operational viewpoints. Moodle learning environments can have anywhere from a few to over a million concurrent users, making it paramount for the system to accommodate such a large number of users, without having to sacrifice overall performance.

The third perspective is availability and resilience, and its desired quality is defined as "the ability of the system to be fully or partly operational as and when required and to effectively handle failures that could affect system availability." (6.3). Since Moodle has so many end users, it is extremely important in the deployment and operational viewpoints because the system must be available to all users at all times, meaning any interference has the potential to affect a large group of people. From the concurrency viewpoint, it is important that the system is available even when large amounts of users are interacting with the learning environment, and can handle failures resulting from these large groups of users. Lastly, the functional, information and concurrency viewpoints although still important, are not as crucial as the others previously listed.

The last perspective is evolution, and its desired quality is defined as "the ability of the system to be flexible in the face of the inevitable change that all systems experience after deployment, balanced against the costs of providing such flexibility." (6.4). Both the functional and information viewpoint have a high importance in evolution, because as Moodle changes, the system at its core, how the functional elements interact and how data is

stored, manipulated and distributed, must be able to meet and facilitate these changes. As these changes are integrated, they will also affect the concurrency and the development viewpoints in that they must adhere to the new functionality of the system. In terms of deployment and operational viewpoints, there is less attention here because at this stage any changes the system experiences will most likely affect the functional and informational viewpoints.

c. Functional View of the Top-Level System

Functional Elements	Responsibilities	Interfaces	Primary Interactions
Assignments	- Allows students to submit work for teachers to grade and receive feedback - Allows submission of one or more files as a group or as one [TC1]	- Input is submission of work - Output is feedback and grades from teacher - Pre-conditions include files of any type the teacher specifies, maximum number and size of files uploaded - Post-conditions include rendering of feedback and grades from teacher [TC1]	Independent
Chat	- Allow participants to have a real-time synchronous discussion in a Moodle course [T1] - Allow users to join and participate in real-time communication -Allow teachers to view past chat sessions [T2]	- input: user messages - output: messages on all connected accounts, a chat log - pre-condition: user will enter input consisting of smilies, links, emoting, beeps, talk, or HTML [T2] - post-condition: Rendered output on all connected chat terminals	Independent
Choice	- Allows teacher to set up questions using radio buttons - Allows students to make selection from possible responses [TC2]	- Input is selection of response from student - Output is students' response results - pre-condition: user selects from one of the radio buttons - post-condition: rendering of classmates' responses and names, up to the discretion of the teacher [TC2]	Independent
Database	- Allow users to build, display and search a bank of record entries [T3]	- input: A new database entry, or a query - output: A returned view or success/error message	Can be used in any of the other modules,

	- View entries - Add entries - Link entries - Import and export entries - Create a new database - Delete an existing database [T4]	- pre-condition: The data is well-formed and of the proper format for the record entry type - post-condition: The output will be correct in relation to what's stored in the database	whenever access to a stored data record is necessary
Feedback	- Allows the creation and conducting of surveys to collect feedback [TC3]	- input: creation of survey - output: feedback from learners - pre-condition: Customizable questions that are either graded or non-graded - post-condition: Working survey that learners can take [TC3]	Independent
Forum	- Allow asynchronous communication between users by posting comments on threads - Allow teachers to create different types of forums: the "Single, simple discussion", the "Question and Answer forum", and the "Standard forum" [T5] - Control settings such as the ability to post attachments, block posts, or grade and rate them [T6]	- input: text or attachment for posting, or administrative commands - output: the desired text in forum-format, or a change in the forum settings - pre-condition: well-formed text, with the option of adding images, sound, and video - post-condition: well-formed output rendered to all browsers	Independent
Glossary	- Allows users to create and maintain a list of definitions - Allows collaborative work or only by teacher [TC4]	- input: an entry - output: addition of entry in glossary - pre-condition: well-formed text - post-condition: entries that can be searched or browsed in different formats]TC4]	Independent
Lesson	- Add lecture content, including questions of	- input: A lesson page, or student activity choice	Independent

	multiple types, to teach students [T7] - Content pages consist of a title, content, multiple options, and then a “jump” to another page in the lesson [T8] - Output detailed reports and statistics for teachers [T9]	- output: The lesson page requested, or a transfer to the next page - pre-condition: valid input - post-condition: valid link to a lesson page	
(LTI) External Tool	- Allows teacher to create an external tool activity or use one configured by site administrator - Stores user’s name or email and receive grades [TC5]	- input: configuration of external tool - output: creation of external tool - pre-condition: launch URL, consumer key and shared secret - post-condition: Generation of external tool available for use to provide new learning materials and activity types [TC5]	Enables users to interact with LTI - compliant learning resources and activities on other web sites [TC5]
Quiz	- Allow teachers to design and build quizzes with multiple question types [T10] - Allow students to submit answers, possibly with automatic grading - Produce statistics and reports on quizzes - Allow control of time spent on each quiz [T11]	- input: info for making a new quiz, or answers to an existing quiz - output: a new quiz generated, or a submitted student quiz - pre-condition: well-formed input - post-condition: well-formed output and reports	Independent
Scorm	- Allows teacher to upload SCORM or AICC package to include in course [TC6]	- input: uploading of SCORM package - output: inclusion of SCORM module in course - pre-condition: SCORM -compliant LMS using the same version of SCORM - post-condition: interoperability, accessibility and reusability of	Independent

		web-based learning content [TC6]	
Survey	- Allow teachers to create surveys to assess and stimulate learning in online environments [T12] - Output statistics and reports to allow tracking of student responses [T13]	- input: a new survey, a response to a survey question, a request for statistics - output: a new or updated survey, a completed survey, a downloadable report - pre-condition: well-formed commands - post-condition: well-formed reports	Independent
Wiki	- Allows collaboration of entire class on single document - Allows students to have their own wiki only visible to them and their teacher [TC7]	- input: creation of page - output: page added to wiki - pre-condition: well-composed document - post-condition: addition of document in wiki [TC7]	Allows community to contribute, edit and develop to wiki [TC7]
Workshop	- Allow students to submit their own work and receive other student's work to grade [T14]	- input: student work - output: grades - pre-condition: well-formed student work in text format - post-condition: accurate grading reports	Independent

d. Functional View of the Chosen Subsystem

Functional Elements	Responsibilities	Interfaces	Primary Interactions
has_capability()	- The most important function in Access API - Check whether a user has a particular capability in a given context - Can check the capabilities of current user or those of userid [T15]	- input: the userid, the context, and the capability - output: true if they have the capability otherwise, false - pre-condition: valid userid, context, and capability - post-condition: true or false	Called from outside of Access API
require_capability()	- Throws access control exception if user does not	- input: capability, context, userid as	Called when user attempts to access

	have the capability [TC8]	null, doanything as true, errormessage as ‘nopermissions’ and stringfile as empty string - output: throws control exception if true, otherwise false - pre-condition: valid parameters - post-condition: true or false [TC8]	but does not have the capability
require_login()	- verify that the user is logged in before accessing any course or activities - verify access to hidden courses and activities - specify availability restrictions for activities - verify user enrollment in course - log access to courses [T15]	- input: nothing, or the course id [T16] - output: force the user to login for all (or a particular) course - pre-condition: valid course id, or nothing - post-condition: user required to log in	Should be called by all plugin scripts after setting up Page->url [T15]
require_course_login()	- Allows read access to content on frontpage without logging in [TC8]	- input: calling of function - output: return true if activity is allowed read access, otherwise false - pre-condition: activity is allowed read access - post-condition: read access is granted [TC8]	Called when user attempts to gain access to content on frontpage without logging in
isguestuser() isloggedin() is_siteadmin()	- Originally necessary for limiting access of special accounts, this is now handled by has_capability() [TC8]	- input: nil or userid - output: true if the user id has the specified property,	Called in the same locations, now deprecated

		false otherwise - pre-condition: valid userid - post-condition: true or false [TC8]	
is_guest() is_viewing() is_enrolled()	- Allows access of course data if either is_enrolled(), is_viewing(), or is_guest() return true [TC8]	- input: nil, or userid [T17] - output: user gains access to course data - pre-condition: is_enrolled() or is_viewing() or is_guest() returns true - post-condition: true or false [TC8]	In order to access course data one of these functions must return true for user [T15]
get_users_by_capability()	- Return a list of users with given capability [T15]	- input: a context, and capability - output: list of users - pre-condition: valid context and capability - post-condition: list of users with given capability [T15]	Used only in course context [T15]

Perspectives

i. Performance & Scalability

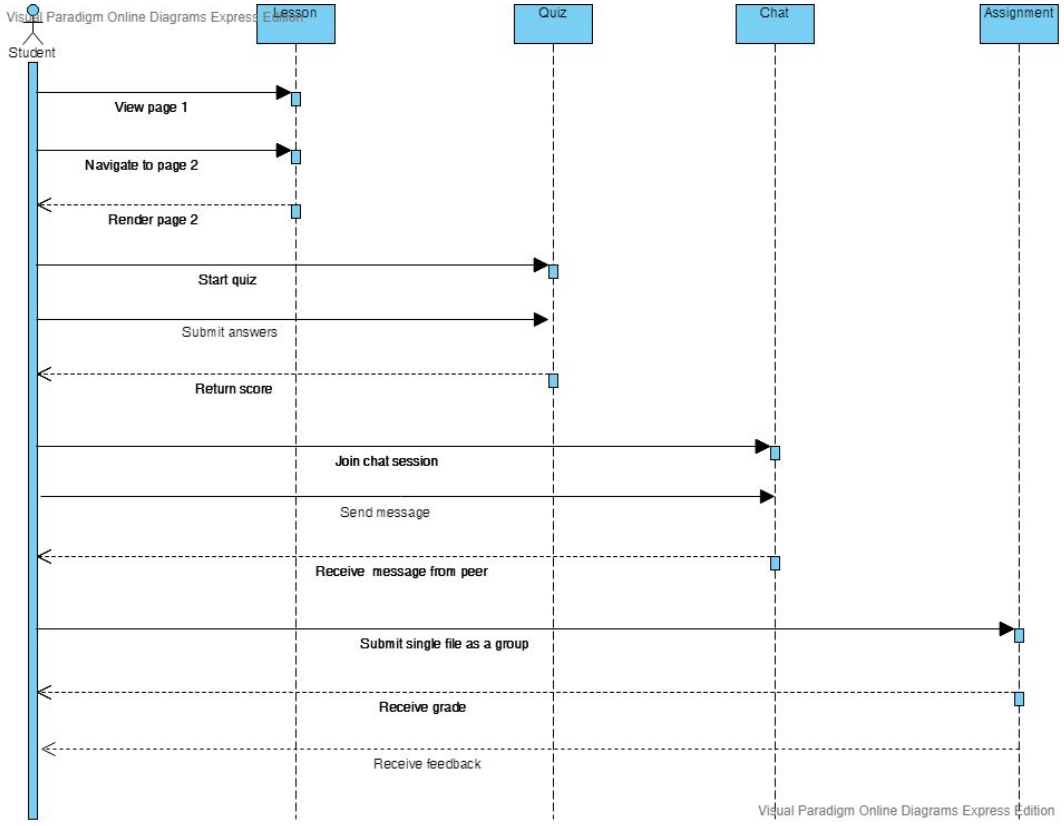
Moodle has been very successful in implementing an architecture capable of supporting millions of users performing complex interactions. “Moodle’s design (with clear separation of application layers) allows for strongly scalable setups.” [T19] Due to its flexibility, Moodle web servers and databases can be put on separate servers, allowing load-balancing and immense scalability. The Moodle architecture is also designed to take advantage of multithreading and gigabit ethernet to improve latency and throughput. Moodle’s organization as a modular and plugin-based system also allows the fine-tuning of performance, and the ability to adapt to requirements changes during scaling.

ii. Security

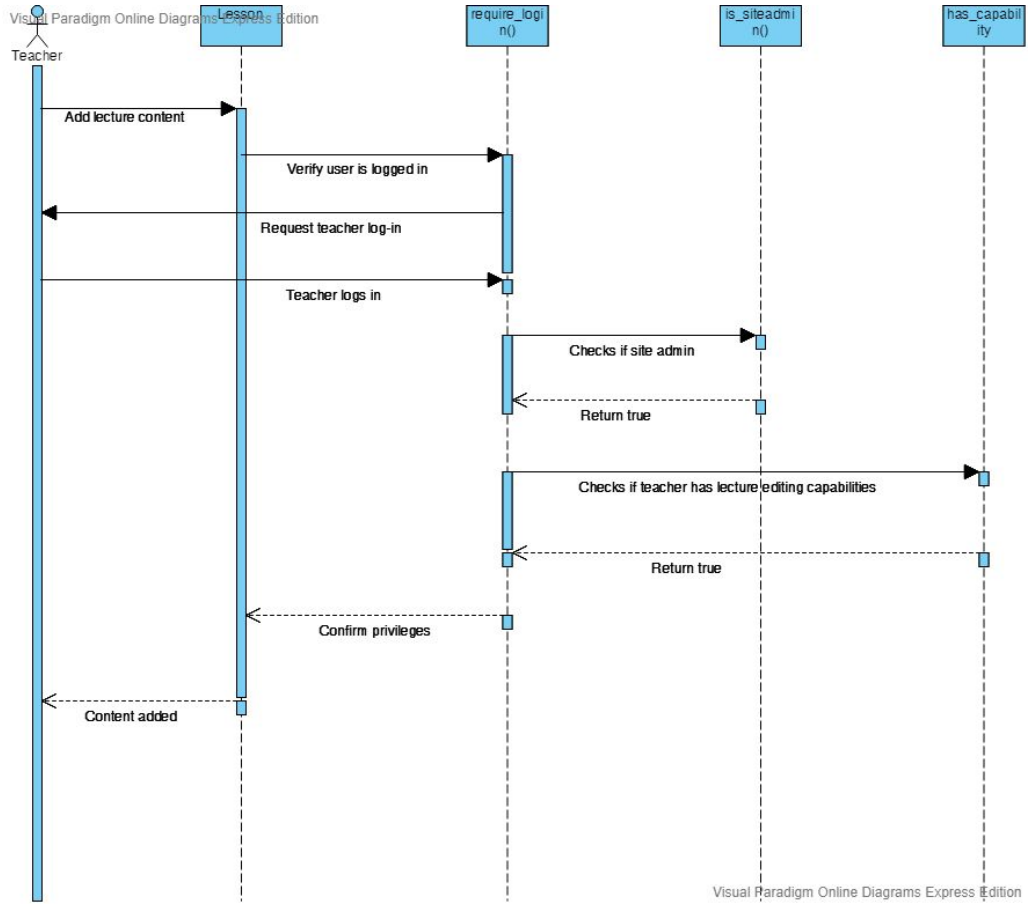
Moodle does a fitting job in their security measures by reliably controlling and monitoring who can perform what actions. This is exemplified by the functional elements in

the Access API such as has_capability(), require_login(), isguestuser() and is_siteadmin(). These functions are responsible for authenticating and verifying the user of Moodle and grant them access to content and actions accordingly. For instance, this would prevent situations where students were incorrectly given the ability to access the quiz activity module. In the case of forbidden access, Moodle is able to both detect and recover by throwing control exceptions. [TC9]

Top-Level System Sequence Diagram



Access API Sequence Diagram



Proposed Extension of the Architecture

a. Description of extension

The extension we have decided to add to Moodle is to allow students to set their visibility on the class list, keeping Moodle compliant with the EU GDPR. Our inspiration for this enhancement comes from the Moodle Tracker [5.a.1]. Administrators will have the ability to enable or disable the student's ability to set visibility. Once the administrator enables this option students are automatically set to "Hidden" on the class list. If a student wants to be visible to others they can go and update their visibility settings globally, or for a specific course.

There are a few benefits of having such enhancements. The first one being protection of identity against access leaks. This extension would successfully protect a students' identity from those who are collecting names through the list of students. Nowadays, even with only the name, an incredible amount of personal information could be revealed.

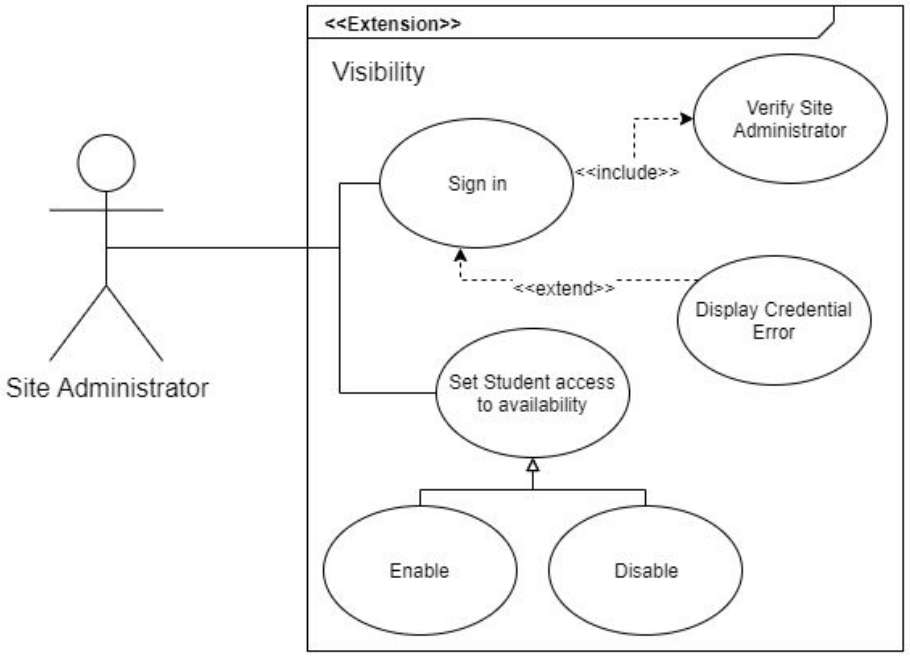
Second, such applications would allow a user to hide themselves from others to accommodate their own needs and conditions. A student might want to be hidden from former groupmates if they've had unpleasant dealings, or to hide from those who overwhelmingly ask for help on assignments and quizzes.

This extension would spare the students from some unwanted trouble and give them confidence their privacy was well taken care of.

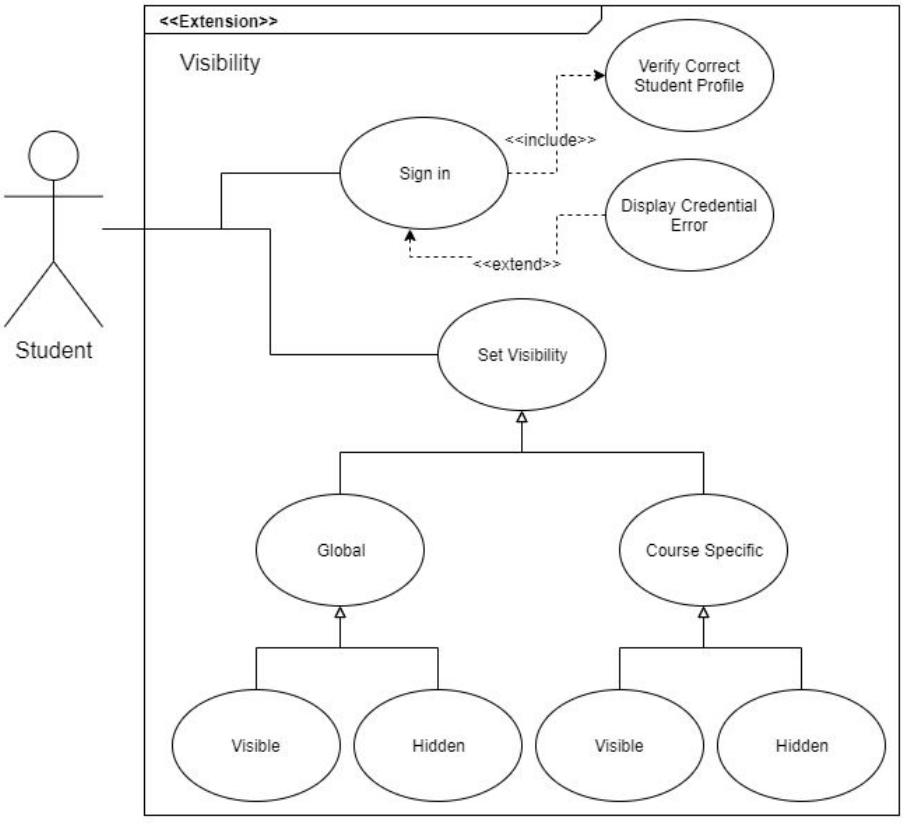
b. Functional and non-functional requirements

i. Functional Requirements

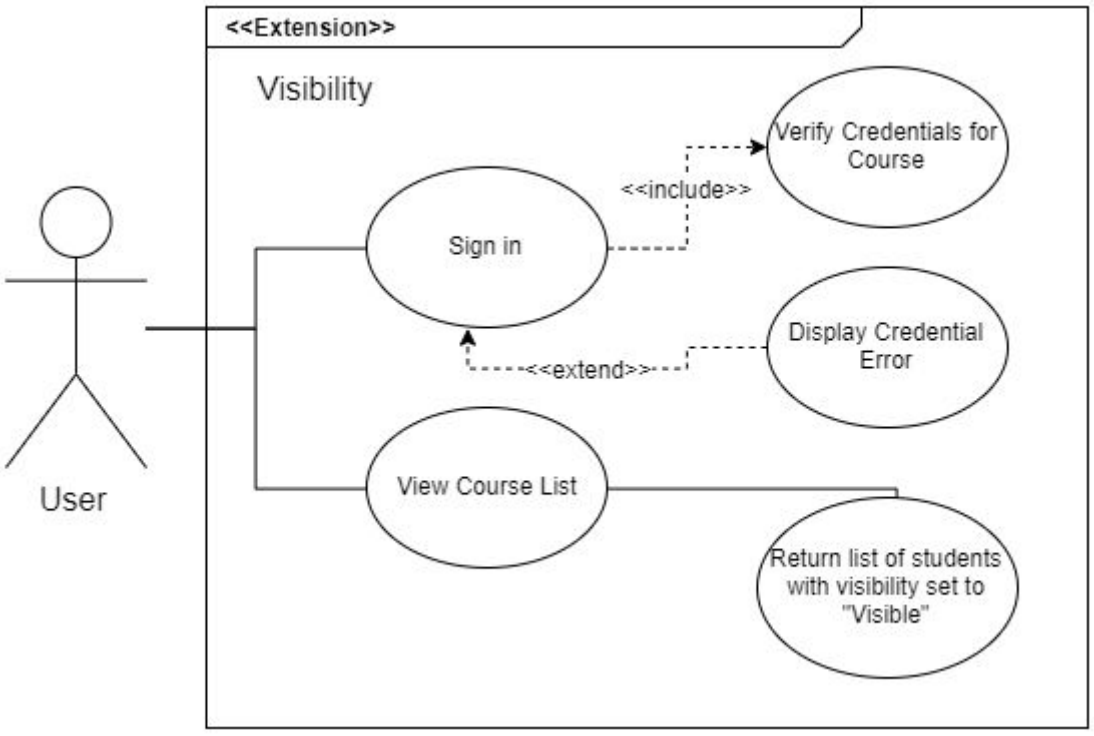
1. The extension shall allow site administrators to enable or disable student access to visibility settings.



2. The extension shall allow students to set global and course-specific visibility preferences.



3. The extension shall exclude students who opt to be “Hidden” from appearing on the course participant list.



4. The extension shall be an opt-in system, by default having students be “Hidden”, in order to comply with the European Union General Data Protection Regulation.

ii. Non-Functional (Quality) Requirements

1. Usability - Site administrators should be able to easily and quickly enable student access to visibility settings.
2. Interoperability - Users opting to be “Hidden” or “Visible” globally should have these decisions reflected in all other Moodle modules.
3. Security - No one, other than site administrators, will be able to view users who choose to be “Hidden” in a course.
4. Safety - If the database containing user visibility preferences is corrupted, the software must treat each user as if they had decided to be “Hidden”.

c. **How the current architecture can support the extension**

Moodle has a well-defined architecture. It has helped millions of students acquire knowledge conveniently. This part will be used to explain the correlation between the existing Moodle and our new feature extension.

Firstly, there are several APIs that would be influenced, including but not limited to: Access API, Data Manipulation API, Persistent API, Page API.

Since Access API takes care of what is seen and what certain users are allowed to do, it needs to enable the system to “hide” users who want to be hidden from the classlist. Also, it needs to enable administrators to enable or disable all requests to “hide” students. As described in

the Moodle docs, this API is also in charge of extending new capabilities to Moodle, which is essentially what we are doing now.

For Data Manipulation API, there needs to be a method to tag specific users who would be hidden from the classlist. This is why Manipulation API has to be adjusted to implement this extension. For example, a user with user ID 123456 submits a request of hiding himself from the classlist to the administrator who has enabled it for the class, then this user should be tagged "hidden for course XXX" in the database. This way, Moodle can handle users just like before, no complicated process needs to be constructed to implement this new feature.

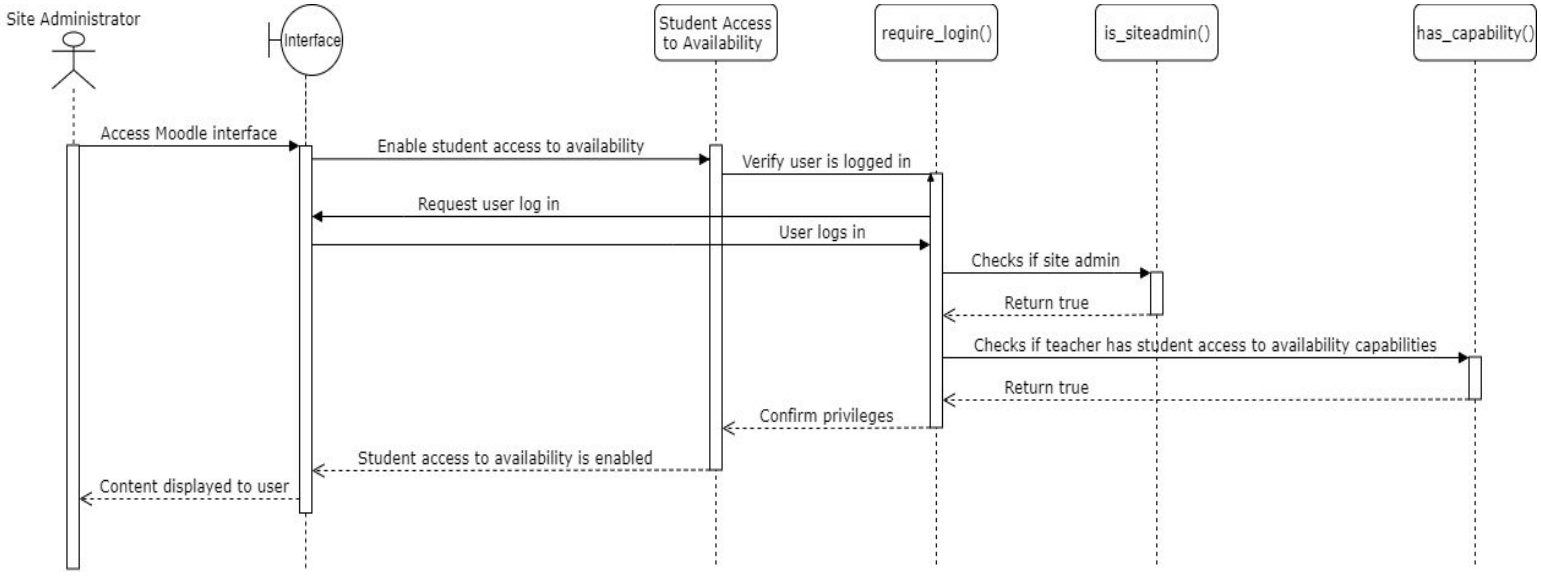
The reason that Persistent API is influenced as well is because when the administrator approves it, the user and the class needs to be updated. This is done by Persistent API as described in Moodle docs. Persistent API "represent[s] an object stored in the database and provide[s] the methods to create, read, update, and delete those objects." Updating this information for users and classes is crucial for this feature.

Lastly, page API needs to be taken into consideration too. Our new feature's core performance occurs on the page showing the members in the class. Page API needs to be coordinating with other APIs to decide which members to show and which should be hidden. Otherwise, even if we did everything described above, we wouldn't be seeing the results we desire.

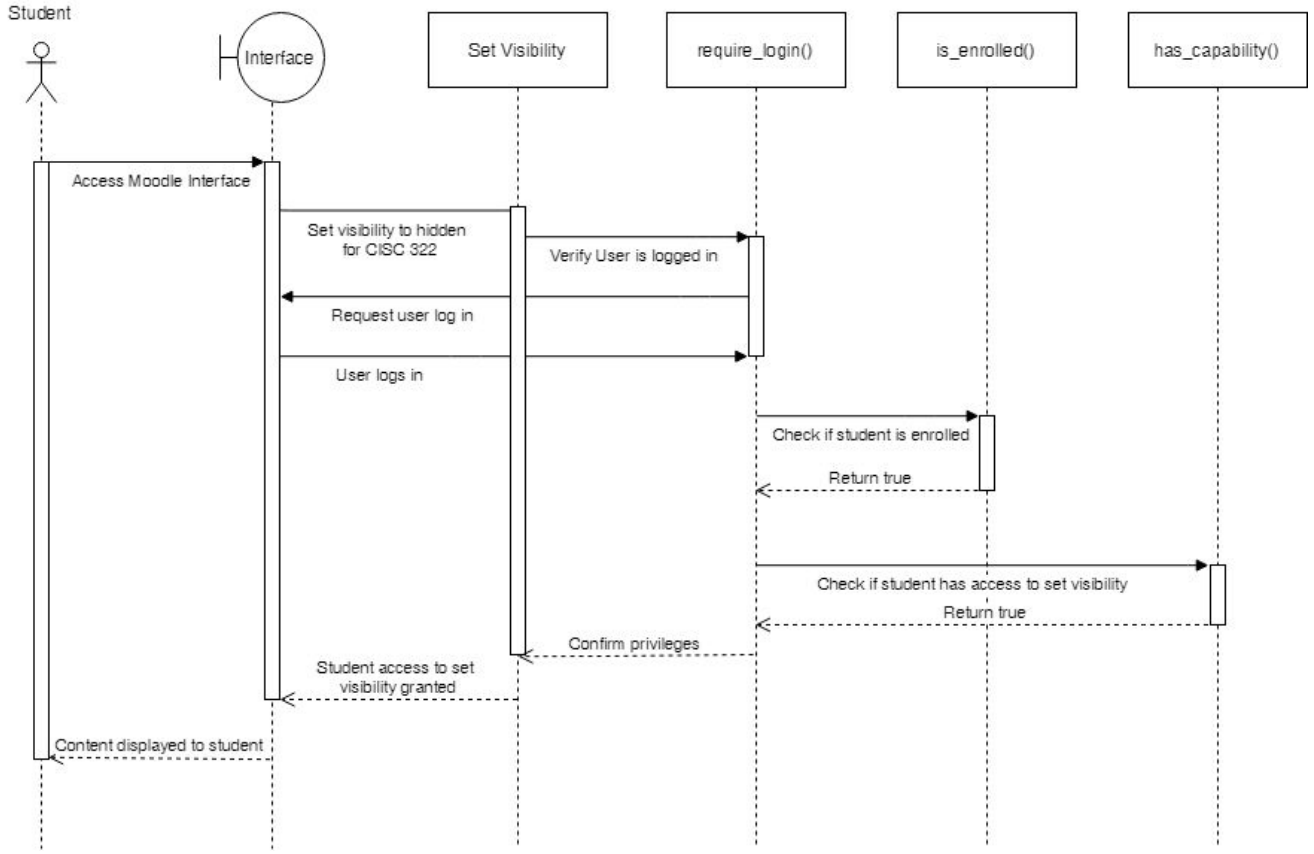
For interfaces, there are two main interfaces that should be updated in order to implement this new feature. First one is the page where the teacher is creating the course, the teacher needs to decide whether this course is allowed to have "hidden" students, or not. This simple clickbox should be added on that page. Secondly, the students should be able to submit a request of hiding himself in his home page of certain course, or globally. That request can be delivered as a new notification to the instructor.

d. **Sequence diagram to explain the flow of use case scenario**

The sequence diagram below illustrates the flow of the first functional requirement above, in this case a site administrator is enabling the student access to visibility settings.



This second sequence diagram illustrates the flow of the second functional requirement above, that of a student setting the visibility settings of a course they are enrolled in.



Conclusions

We investigated the conceptual architecture of the Moodle system, and have organized them into several parts as different features of software architecture.

We analyzed Moodle's purpose and scope as it is an open source LMS which expands to 231 countries, and currently has almost 200 000 000 total users. The target audience of Moodle is everyone that has an interest in joining Moodle. Since it's open source, it means anyone can use Moodle to build their own projects, or be part of other projects that someone else created. We found Moodle architecturally interesting as that it's highly modular, its scalability and multilingualism, it's foundation on open-source development, its common theme of accessibility and using assistive technologies, and it's easy integration. For software quality attributes, the fast performance it provides, the secure environment and protection of information it has, the ability to detect and avoid system failures, the multi-platform service, the ability to evolve to user needs, its ease of use, and its interaction with other systems are analyzed. All of these specifications were gracefully balanced to provide a useful system that has earned its success and popularity.

We gathered that the most important stakeholders for Moodle were acquirers, accessors, communicators, competitors, developers, investors, maintainers, suppliers, support staff, system administrators, testers and users. A chart is built to explain their individual roles to the system. We established an analysis on the functional requirements as assignments, chat, choice, database feedback, forum, glossary, external tools and quiz. We also built an use-case diagram for the functional requirements. For the non-functional requirements, they are analyzed as availability, capacity, interface, modifiability, performance, reliability, safety, security, supportability, testability and usability. Both the functional and non-functional requirements are expanded to explanations in shall statements.

We collected the architectural views and have presented them as the six viewpoints and analyzed their individual importance to the system as perspectives of security, performance and scalability, reliability and resilience, then evolution. We then went deeper into each of the perspectives, and explained why the levels of importance are ranked that way. Then we analyzed the context view of the system as account management, course content, enrollment and access control, students, teachers, externals and organizations with educational needs. Then we went more in depth to each of the views and created a context diagram to express their relationships in the system. We also analyzed the concrete architecture itself to discover the ways viewpoints were utilized to build the functional view of the system. From there we used perspectives to reason about how the system satisfies its quality attributes. Finally we included sequence diagrams to cover the most important use cases faced by Moodle.

We proposed an enhancement to the architecture, that would provide the choice for students to be non-visible to other students in the same class with the permission of an administrator. We identified the functional requirements and corresponding non-functional requirements, and used UML use case diagrams to visualize the functional requirements for our enhancement. Then we justified how the current architecture could support our enhancement, explaining which components and interfaces can be used in order to implement it. Lastly we provided sequence diagrams to explain the flow of the use case scenarios established.

In conclusion, we have analyzed Moodle in terms of its purpose of scope, its audience, the reason it's architecturally interesting, its software quality attributes, its stakeholders, its functional and non-functional requirements,its architectural and context views, its concrete architecture and we have proposed an enhancement for the current architecture.

References

[1.a.1, 1.a.5 & 1.b.2] "Moodle." *Wikipedia*, Wikimedia Foundation, 16 Oct. 2019, en.wikipedia.org/wiki/Moodle.
[1.a.2-3] "Mission." *Mission - MoodleDocs*, 20 July 2017, docs.moodle.org/dev/Mission.
[1.a.4] "Statistics." *Moodle.org*, 18 Oct. 2019, stats.moodle.org/.
[1.b.1&3] "About Moodle." About Moodle - MoodleDocs, 4 Dec. 2018, docs.moodle.org/37/en/About_Moodle.

[2.d.1]https://onq.queensu.ca/d2l/le/content/345189/viewContent/1992808/View
[2.d.1]https://onq.queensu.ca/d2l/le/content/345189/viewContent/2046394/View
[2.d.2]https://moodle.org/plugins/
[2.e.1]https://docs.moodle.org/dev/Main_Page
[2.e.2]https://scitools.com/

[3.a.1] https://delftswa.gitbooks.io/desosa2018/elasticsearch/chapter.html#1-stakeholders
[3.a.2] https://docs.moodle.org/dev/Overview
[3.a.3] https://docs.moodle.org/19/en/GNU_General_Public_License
[3.a.4] https://moodle.com/partners/#
[3.a.5] https://docs.moodle.org/dev/New_feature_ideas
[3.a.6] https://docs.moodle.org/dev/Moodle_3.7_release_notes
[3.a.7] https://docs.moodle.org/dev/Testing
[3.b.1] "Activities." *Activities - MoodleDocs*, https://docs.moodle.org/37/en/Activities.
[3.b.2] Imam, F 2019, Module VIII: Architecture Definition Process: Scope, Concerns, and Principles, lecture notes, Software Architecture CISC 322, Queen's University, delivered 30 Sep 2019

[4.1] "4.4 Quality Attribute Scenarios in Practice." *4.4 Quality Attribute Scenarios in Practice :: Chapter 4. Understanding Quality Attributes :: Part Two: Creating an Architecture :: Software Architecture in Practice :: Programming :: ETutorials.org*.

[4.2] "Architectural Perspectives." *Software Systems Architecture: Identifying and Using Scenarios*, by Nick Rozanski and Eoin Woods, Addison-Wesley, 2012, pp. 121-134.

[5.a.1] Anabitarte, Amaia. "Enable a User to Be Hidden in the Participation List of a Course." *[MDL-67279] Enable a User to Be Hidden in the Participation List of a Course - Moodle Tracker*, 16 Nov. 2019, 12:29AM, tracker.moodle.org/browse/MDL-67279?fbclid=IwAR2tLuJ7l-KUrcXFMzWtFfJ_K7I3VrWfn1E0CApm-UU-jToPh1luAXEDIQ4.

[6.1-4] “Architectural Perspectives.” *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*, by Nick Rozanski and Eoin Woods, Addison-Wesley, 2012, pp. 39–53.

[T1] https://docs.moodle.org/37/en/Chat_activity
[T2] https://docs.moodle.org/37/en/Using_Chat
[T3] https://docs.moodle.org/37/en/Database_activity
[T4] https://docs.moodle.org/37/en/Using_Database
[T5] https://docs.moodle.org/37/en/Forum_activity
[T6] https://docs.moodle.org/37/en/Forum_settings
[T7] https://docs.moodle.org/37/en/Lesson_activity
[T8] https://docs.moodle.org/37/en/Building_Lesson
[T9] https://docs.moodle.org/37/en/Using_Lesson
[T10] https://docs.moodle.org/37/en/Quiz_activity
[T11] https://docs.moodle.org/37/en/Quiz_settings
[T12] https://docs.moodle.org/37/en/Survey_activity
[T13] https://docs.moodle.org/37/en/Using_Survey
[T14] https://docs.moodle.org/37/en/Workshop_activity
[T15] https://docs.moodle.org/dev/Access_API
[T16] stackoverflow.com/questions/4083878/how-to-check-whether-user-are-login-in-moodle
[T17] www.cheatography.com/hitteshahuja/cheat-sheets/moodle-developer-s-cheat-sheet/
[T18]www.viewpoints-and-perspectives.info/home/perspectives/performance-and-scalability/
[T19] https://docs.moodle.org/37/en/Performance_recommendations#Scalability
[TC1] https://docs.moodle.org/37/en/Assignment_activity
[TC2] https://docs.moodle.org/37/en/Choice_activity
[TC3] https://docs.moodle.org/37/en/Feedback_activity
[TC4] https://docs.moodle.org/37/en/Glossary_activity
[TC5] https://docs.moodle.org/37/en/External_tool
[TC6] https://docs.moodle.org/37/en/SCORM_activity
[TC7] https://docs.moodle.org/37/en/Wiki_activity
[TC8] https://docs.moodle.org/dev/Access_API
[TC9] https://www.viewpoints-and-perspectives.info/home/perspectives/security/