Software Design & Architectures

SOFE 3650 Case Study: Course Management System (CMS)

CRN: 43963

Prepared By:

100617713 Samantha Husack
100620049 Alexander Hurst

Due: December 5, 2018
Instructor: Professor Ramiro Liscano
Possible Marks: 60

Case Study: Course Management System (CMS)

Table of contents

1: Business Case
2: System Requirements
    2.1: Use Case Model
    2.2: Quality Attribute Scenarios
    2.3: Constraints
    2.4: Architectural Concerns
3: Design Process
    3.1: ADD Step 1: Review Inputs
    3.2: Iteration 1
        3.2.1: Step 2: Iteration Goal By Selecting Drivers
        3.2.2: Step 3: Choose Elements of the System to Refine
        3.2.3: Step 4: Choose Design Concepts that Satisfy Selected Drivers
        3.2.4: Step 5: Instantiate Architectural Elements, Allocate Responsibilities, Define Interfaces
        3.2.5: Step 6: Sketch Views and Record Design Decisions
        3.2.6: Step 7: Perform Analysis of Current Design and Review Iteration Goal
    3.3: Iteration 2
        3.3.1: Step 2: Iteration Goal By Selecting Drivers
        3.3.2: Step 3: Choose Elements of the System to Refine
        3.3.3: Step 4: Choose Design Concepts that Satisfy Selected Drivers
        3.3.4: Step 5: Instantiate Architectural Elements, Allocate Responsibilities, Define Interfaces
        3.3.5: Step 6: Sketch Views and Record Design Decisions
        3.3.6: Step 7: Perform Analysis of Current Design and Review Iteration Goal
    3.4: Iteration 3
        3.4.1: Step 2: Iteration Goal By Selecting Drivers
        3.4.2: Step 3: Choose Elements of the System to Refine
        3.4.3: Step 4: Choose Design Concepts that Satisfy Selected Drivers
        3.4.4: Step 5: Instantiate Architectural Elements, Allocate Responsibilities, Define Interfaces
        3.4.5: Step 6: Sketch Views and Record Design Decisions
        3.4.6: Step 7: Perform Analysis of Current Design and Review Iteration Goal
4: Summary
5: References

1: Business Case

2: System Requirements

Requirement elicitation activities had previously been performed, and the following is a summary of the most relevant requirements collected. To view the entirety of the requirements for this business case, please refer to the Requirements Document here.

2.1 Use Case Model

The following use case model presents the most relevant use cases that support the CMS Model in the system. Other use cases are not shown.

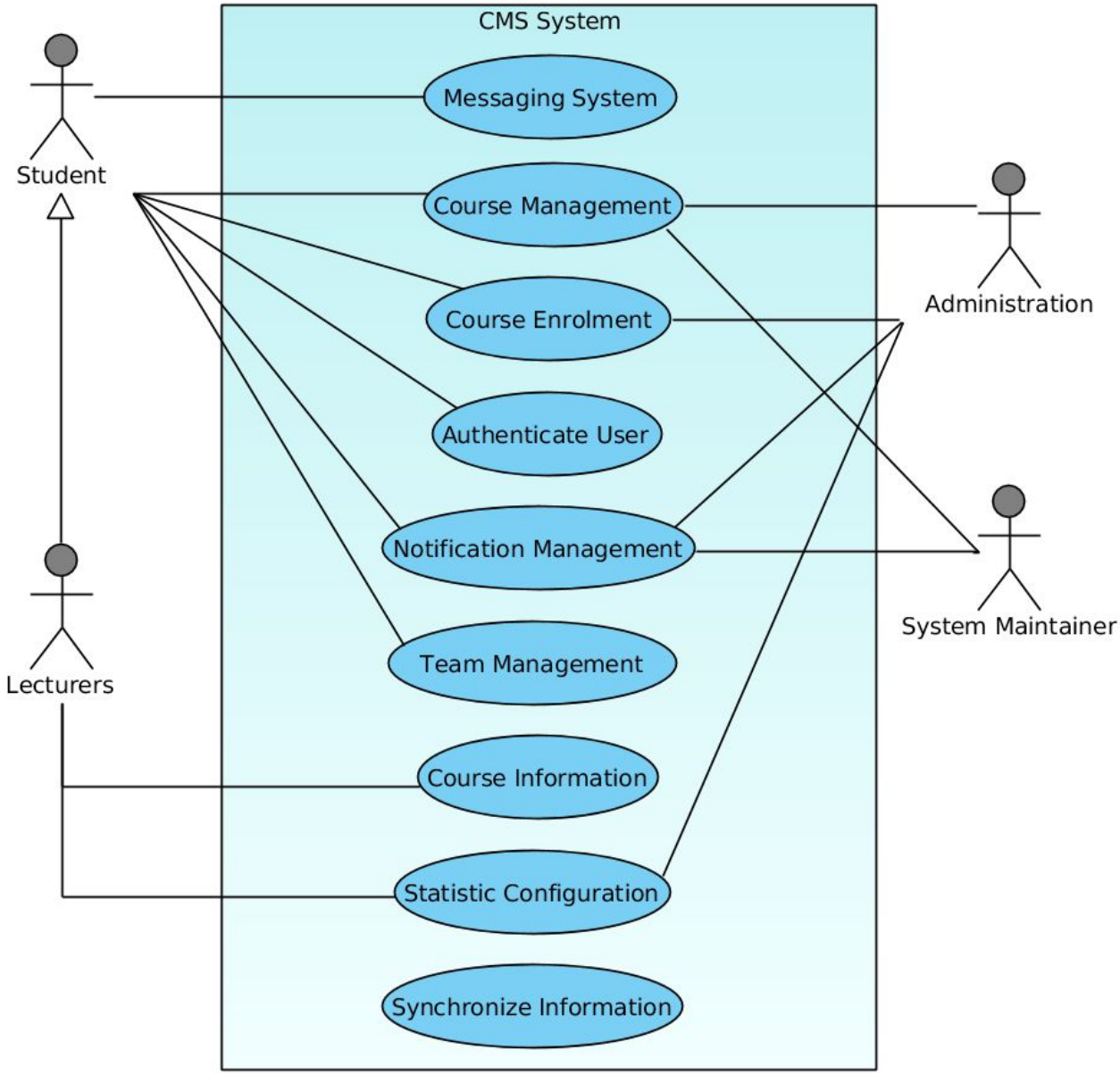

Figure 2.1 Use Case Model for CM System

Each of these use cases is described in the following table:

Use Case	Description
UC-1: Messaging System	The system can provide a messaging system for all users where: Students may send and receive messages to and from individuals, teams and all course participants including lecturer(s). Lecturers may send and receive messages to and from individuals, teams, other lecturers, and all course participants as well as post announcements.
UC-2: Course Management	Lecturer: Duplicate and import courses and materials (for their own courses only), Manage static course info, Prevent students from subscribing from non-qualified courses. Create courses, new courses, recreate the course, Register assistant lecturer, Prep roster, lecture schedule, Upload and manage course material Students: Search static and dynamic information and files. Students may only manipulate their own profile and personal information as well as retrieve course information. Maintainer: The system shall allow maintainers to limit the size of uploads for lectures and students, Limit total available space for specific courses Admin to manage courses, create new courses, delete courses, update static course info, appoint principal lecturers to courses, Admin to specify min num of students for course -- too little = cancelled course for that semester, No max limit for num course students ever. **NEEDS TO CONDENSE**
UC-3: Course Enrolment	The system shall allow course subscription by students, administration and enrolment policies shall be assigned by lecturers.
UC-4: Authenticate User	When a student, lecturer, administrator or maintainer would like to use the system, he/she needs an established session. If one is not in session, they are prompted to enter a user ID and password. Not all pages are available to all users.
UC-5: Notification Management	The system provides a notification manager for announcements, course updates and messages.
UC-6: Team Management	The system supports teams and teams management where lecturers can create and manage teams via the means of sending information, viewing, inserting and removing students as well assigning assistant lecturers. Students may join, create, leave, share to, upload, download from and message through teams as well.

UC-7: Course Information	The system shall provide, store, and represent static and dynamic information.
UC-8: Statistical Configuration	The system will allow statistical data to be drawn from the application records.
UC-9: Synchronize Information	The system can synchronize with secondary university systems.

[Add the diagram to a separate folder?]

2.2: Quality Attribute Scenarios

In addition to these use cases, a number of quality attribute scenarios were elicited and documented. The eight most relevant ones are presented in the following table. For each scenario, we also identify the associated use case(s) [AUC].

ID	Quality Attribute	Scenario	AUC
QA1	Security	A user can only make specified changes to the system based on their status and allowances. It is possible to know who performed the operations and when it was performed 100% of the time.	All
QA2	Availability	A failure occurs in the CMS during normal operation. The downtime is regulated to a maximum of 4 hours per month. All expected downtime has been announced a minimum of 48 hours in advance. Expected downtime occurs only during low-intensity hours.	All
QA3	Performance	Peak load performance even when usage is high.	All
QA4	Accuracy	A student requests to see their grades. The grade given should be accurate to all of the data that the system has been provided	UC-2
QA5	Compatibility, Usability	If a user wants to access information provided by a secondary university system, it will synchronise the content and display correctly.	UC-9
QA6	Durability	If the system crashes while a student enrols in a course when the system recovers the student should still be enrolled	UC-3
QA7	Simplicity	Lecturers should have the ability to send messages to all individuals, teams, etc with one click functionality	UC-1
QA8	Efficiency, Usability, Simplicity	The UI is consistent, descriptive and intuitive for all users, including the disabled. When a user wants to access any content it is less than 3 clicks away.	UC-7

2.3: Constraints

A set of constraints on the system and its implementation were collected. These are presented in the following table.

ID	Constraint
CON-1	A minimum of 1000 simultaneous users must be supported.
CON-2	The system must be accessed through a web browser (Chrome 70, Safari 12, Firefox 63, Internet Explorer 11, Opera 56) on different platforms: Windows, OSX, Linux.
CON-3	An existing relational database server must be used.
CON-4	The system shall provide an export to commonly used calendar formats
CON-5	The system shall be able to import BOZ roster information into the course Roster.
CON-6	The system shall have an intuitive, consistent and descriptive UI. The system shall be accessible by disabled (blind) users, who should be able to navigate the system and have access to all content and functionality.
CON-7	The system shall not allow users to change information which is contained and maintained by secondary university systems. The system shall automatically synchronize with secondary university systems.
CON-8	The system shall be easily testable, scalable, extensible, evolvable and maintainable.

2.4: Architectural Concerns

As this is a content management system under a greenfield development, only a few architectural concerns are identified initially, as shown in the following table.

ID	Concern
CRN-1	Establishing an overall initial system structure.
CRN-2	Leverage of the team's knowledge of various programming languages.
CRN-3	Allocate work to members of the development team.

3: Design Process

The following section is divided into four parts. First, we review the inputs. This is essential to identify the requirements chosen to be drivers in design decisions through the design process. Next, we iterate through three iterations of the ADD Process steps 2 through 7.

3.1: ADD Step 1: Review Inputs

In the first step of the ADD method, we review the inputs and identify which requirements will be considered as drivers. These inputs are summarised in the following table.

Category	Details

<table>
<tr><td>Design purpose</td><td>The purpose is to produce a sufficiently detailed design to support the construction of the system.</td></tr>
<tr><td>Primary functional requirements</td><td>From the uses cases presented in Assignment 2, the primary ones were determined to be:
UC-1: Because it directly linked to the core features of the system.
UC-2: Because it directly linked to the core features of the system.
UC-5: Because of the technical issues associated with it.
UC-6: Because it directly linked to the core features of the system.
UC-9: Because of the technical aspects, as well as core functionality.</td></tr>
<tr><td>Quality attribute scenarios</td><td>The scenarios described in Assignment 2 have now been prioritized as follows:
<table>
<tr><th>ID</th><th>Importance to CMS Stakeholders</th><th>Implementation Difficulty</th></tr>
<tr><td>QA-1</td><td>HIGH</td><td>MEDIUM</td></tr>
<tr><td>QA-2</td><td>MEDIUM</td><td>MEDIUM</td></tr>
<tr><td>QA-3</td><td>HIGH</td><td>HIGH</td></tr>
<tr><td>QA-4</td><td>MEDIUM</td><td>LOW</td></tr>
<tr><td>QA-5</td><td>HIGH</td><td>HIGH</td></tr>
<tr><td>QA-6</td><td>LOW</td><td>MEDIUM</td></tr>
<tr><td>QA-7</td><td>HIGH</td><td>HIGH</td></tr>
<tr><td>QA-8</td><td>HIGH</td><td>HIGH</td></tr>
</table>
From this list, only QA-1, QA-3, QA-5, QA-7 and QA-9 are selected as drivers</td></tr>
<tr><td>Constraints</td><td>All of the constraints discussed in Assignment 2 are included as drivers.</td></tr>
<tr><td>Architectural concerns</td><td>All of the architectural concerns discussed in the above section 2.4 are included as drivers.</td></tr>
</table>

3.2: Iteration 1

The first iteration of the ADD design process aims to establish an overall system structure.

3.2.1: Step 2: Iteration Goal By Selecting Drivers

The first iteration in the design of a content management system, although driven by a general architectural concern, all the drivers that may influence the general structure must be kept in mind. In the Course Management System, it is important to keep in mind the following drivers throughout this iteration:

- UC-9: Connect to a secondary university system
- QA-1: Security
- QA-3: Performance
- QA-7: Simplicity
- CON-2: System must be a web browser application

- CON-3: An existing relational database must be used
- CON-7: The system cannot allow changes to be made to a secondary university system

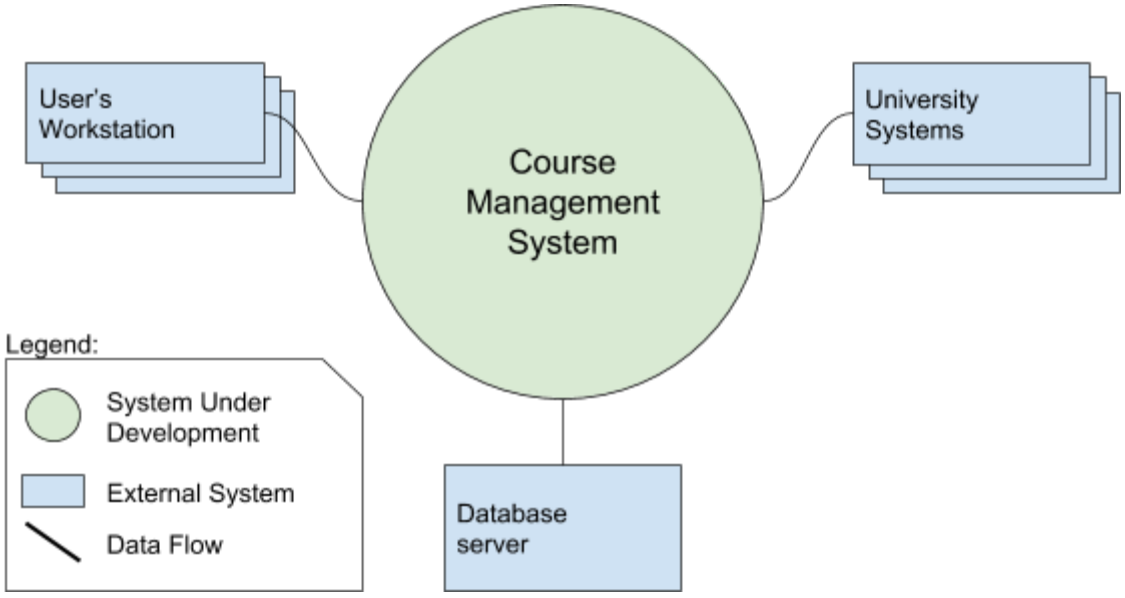

Figure 3.1 Context Diagram for the Course Management System

3.2.2: Step 3: Choose Elements of the System to Refine

"This is a greenfield development effort, so in this case, the element to refine is the entire Course Management System shown in figure 3.1. In this case, refinement is performed through decomposition" (Cervantes, 2016).

3.2.3: Step 4: Choose Design Concepts that Satisfy Selected Drivers

In this initial iteration, given the goal of structuring the entire system, the design concepts chosen reflect this decision. The following table summarizes the selection of design decisions.

Design Decision and Location	**Rationale**
Logically implement the client of the system using the Web Application reference architecture	The Web Application reference structure supports the development of applications initiated on a web browser, communicating with a server via the HTTP protocol. As we want the application to be able to be accessed through a web browser (CON-2) the client side of the system is best implemented using the Web Application reference structure where the web browser runs on the client machine. DISCARDED ALTERNATIVES: - Rich Internet Application: This reference architecture is oriented towards the development of applications with rich user interfaces running inside the browser as a plugin. This option was

	discarded as no plugin was wanted to be able to run the application itself, rather plugins could be used to interact with various options within the application itself. - Mobile Application: This reference structure is executed on a handheld device and usually works in collaboration with a support infrastructure that resides remotely. This means that the application would work only on handheld devices as an installed application In order to maintain multi-platform compatibility, this option was discarded.
Logically structure the server part of the system using a service application reference structure	The service application would provide an interface for the web application to fetch any of the data that it would need as well as to push any changes such as course registration. ADVANTAGES: - Clients and Service can be loosely coupled - No user interface is needed, and none provided No other alternatives were considered
Physically structure the application using Four tier deployment	A Four tier Deployment allows for separation between the Web server and the database. In this place a Business tier can be utilized to improve the security of the system by preventing public access, to decrease the coupling between the web tier and the database tier and to allow easier synchronization with secondary school systems. ADVANTAGES: - Separation of the Business logic tier from database and web tier - Addition of the Business logic tier improves security options especially for integration with secondary university systems - Decreases coupling between web server, and database DISCARDED ALTERNATIVES: - > four tier deployments: Having the Coupling of the web tier and the database tier would lead to greater difficulty implementing synchronization with secondary university systems
Build the user interface of the client application using HTTP and the angular framework	HTTP + react is a standard method for developing user-friendly scalable web applications (QA-8) Advantages: - Quick development - Good maintainability DISCARDED ALTERNATIVES: - HTML + PHP: PHP language is hard to maintain, inconsistent and insecure
Deploy the application servers	Custom deployment solution consisting of a Linux server dedicated to a web server and business logic tier, a secondary database server. Web Server running Apache to serve content, with Golang as the backend.

	Database server running Postgresql database. Both servers with redundancy/ load balancing servers. ADVANTAGES: - Requests have built-in asynchronous support, thanks to golang - Load balancing provides high availability + performance - Open source standard technologies provide free stable software - Controlling servers allows for backups + server requirements to be regulated by the school including opting for geo-redundant options DISCARDED ALTERNATIVES - Enterprise (Microsoft) using WebDeploy: fewer customization options lead to a more difficult four-tier deployment with less control

3.2.4: Step 5: Instantiate Architectural Elements, Allocate Responsibilities, Define Interfaces

The instantiation design decisions considered and made are summarized in the following table. No interface definitions can be made at this time.

Design Decision and Location	**Rationale**
Create a module dedicated to accessing the secondary university systems	The Service Interfaces component in the Services Layer is modified to abstract the access to the secondary university systems. This will facilitate the achievement of QA-2 and QA-5 and play a critical role in the achievement of UC-9.

The results of these instantiation decisions are recorded in the next step.

3.2.5: Step 6: Sketch Views and Record Design Decisions

The diagram shown in Figure 3.2 shows the sketch of a module view of the two reference architectures that were selected for the client and server applications. These have been adapted according to the design decisions we made.

[MODULE VIEW - CLIENT AND SERVER APPLICATION VIEW]

Figure 3.2 Module View

The following table outlines the elements as well as a short description of its responsibilities. In this first iteration, only the major functional responsibilities are detailed.

Element	Responsibility

[TO BE FILLED OUT WHEN DIAGRAM IS COMPLETE]

[INITIAL DEPLOYMENT DIAGRAM - 4-tier]

Figure 3.3 Initial Deployment diagram for the CMS

The deployment diagram in Figure 3.3 demonstrates the 4-tier deployment and the associated components. The responsibilities of these components and the observed relationships are outlined below:

Element	Responsibility
Relationship	**Description**

3.2.6: Step 7: Perform Analysis of Current Design and Review Iteration Goal

The following table summarizes the design progress

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions made during the iteration
UC-1			No relevant decisions were made, as it is necessary to identify the elements that participate in this use case that is associated with the scenario.
UC-3			No relevant decisions were made, as it is necessary to identify the elements that participate in this use case that is associated with the scenario.
UC-5			Specific details for notification management have not been decided as of now
UC-7			Selected reference architecture establishes the modules that will support this functionality.
	UC-9		The chosen server-side Service Application reference

			structure and the module dedicated to accessing the secondary university systems in the Services Layer allows the system to access the secondary university systems.
	CON-1		Selected reference architecture establishes the modules that will support this functionality.
		CON-2	The selected Client-Side Web Application reference structure addresses this constraint.
	CON-3		The selected Service Application will allow this constraint to be addressed fully at a later stage.
CON-4			No relevant decisions were made, as it is necessary to identify the elements that participate in this constraint.
	CON-5		The selected Service Application will allow this constraint to be addressed fully at a later stage.
CON-6			Implementation details of the user interface are currently undecided
	CON-7		Business Logic layer will provide an interface for synchronizing with secondary university systems for access only
	CON-8		Frameworks and deployment tiers have been carefully selected to decrease coupling between components for easier scaling and maintenance
	QA-1		Business logic layer will allow for efficient handling of security with secondary schools as well as some security abstraction from the web server
	QA-2		The use of redundant servers will improve the availability of the service, as will the decision to use stable, standard and open source frameworks
	QA-5		Business logic layer will provide an interface for secondary universities to access the information in the school's database in a synchronized manner
QA-7			The interface specifics will be designed in future iterations
QA-8			The interface specifics will be designed in future iterations
		CRN-1	Frameworks, deployment, application, etc patterns have been chosen and are awaiting further specific design
	CRN-2		Frameworks and languages chosen are familiar to the development team
CRN-3			(non-existent) Dev team has not been allocated jobs

3.3: Iteration 2

The second iteration of the ADD design process aims to identify structures to support primary functionality.

3.3.1: Step 2: Iteration Goal By Selecting Drivers

3.3.2: Step 3: Choose Elements of the System to Refine

3.3.3: Step 4: Choose Design Concepts that Satisfy Selected Drivers

3.3.4: Step 5: Instantiate Architectural Elements, Allocate Responsibilities, Define Interfaces

3.3.5: Step 6: Sketch Views and Record Design Decisions

3.3.6: Step 7: Perform Analysis of Current Design and Review Iteration Goal

3.4: Iteration 3

The third iteration of the ADD process aims to

3.4.1: Step 2: Iteration Goal By Selecting Drivers

3.4.2: Step 3: Choose Elements of the System to Refine

3.4.3: Step 4: Choose Design Concepts that Satisfy Selected Drivers

3.4.4: Step 5: Instantiate Architectural Elements, Allocate Responsibilities, Define Interfaces

3.4.5: Step 6: Sketch Views and Record Design Decisions

3.4.6: Step 7: Perform Analysis of Current Design and Review Iteration Goal

4: Summary

5: References

H. Cervantes and R. Kazman, *Designing software architectures: a practical approach*. Toronto: Addison-Wesley, 2016.