ITERATION #1

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 for this section: 15

Case Study: Course Management System (CMS)
ITERATION 1

Table of contents

Design Process

The following section reflects iteration 1 of the ADD process. 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 go through this first iteration of the ADD Process for steps 2 through 7.

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.

<table>
<tr><th>Category</th><th>Details</th></tr>
<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-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>
<tr><td>QA-9</td><td>HIGH</td><td>MEDIUM</td></tr>
</table>
Only QA-1, QA-3, QA-5, QA-7 and QA-8 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>

Iteration 1

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

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

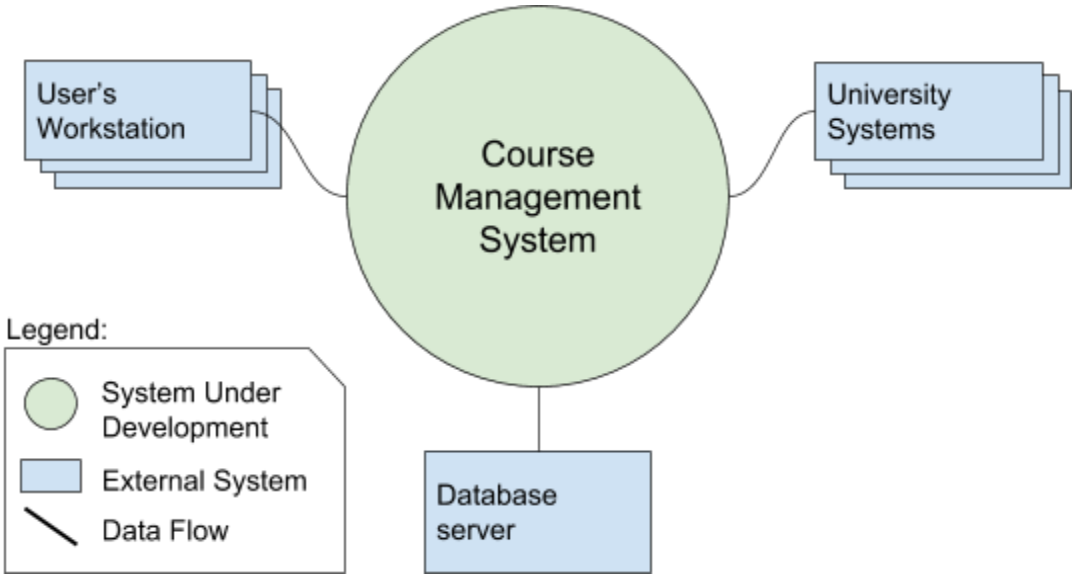

Figure 3.1 Context Diagram for the Course Management System

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: 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: - Less than 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

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	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.

university systems	

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

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.

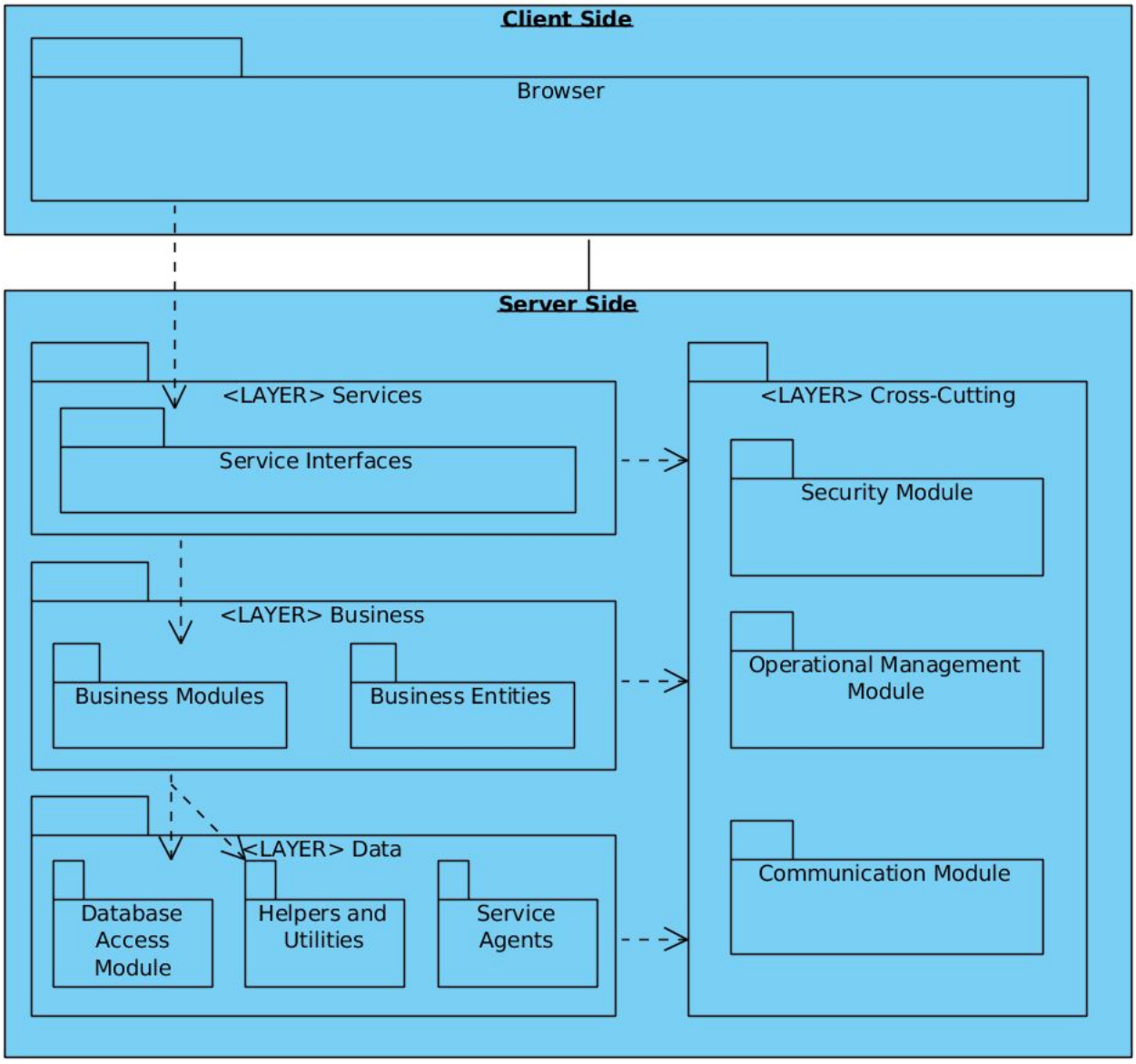

Figure 3.2 Module View for the Client and Server Application View

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**
Browser Client Side	The browser takes care of all client-side operations, sending requests back to the server.
Services Server Side (SS)	**This layer contains modules that expose services that are consumed by the clients**
Service Interfaces SS	These modules expose services that are consumed by the clients
Business SS	**This layer contains modules that perform business logic operations**
Business Modules SS	These modules implemented business operations
Business Entities SS	These entities make up the domain model
Data SS	**This layer contains modules that are responsible for data persistence and for communication with the time servers**
Database Access Module SS	This module is responsible for persistence of business entities into the relational database. It performs object-oriented to relational mapping and shields the rest of the application from persistence details.
Helpers and Utilities SS	These modules implement utilities of the data layer
Service Agents SS	These modules implement service agents for the server side data layer
Cross Cutting SS	**These modules have functionality that goes across the different layers, such as security, loggin and I/O**

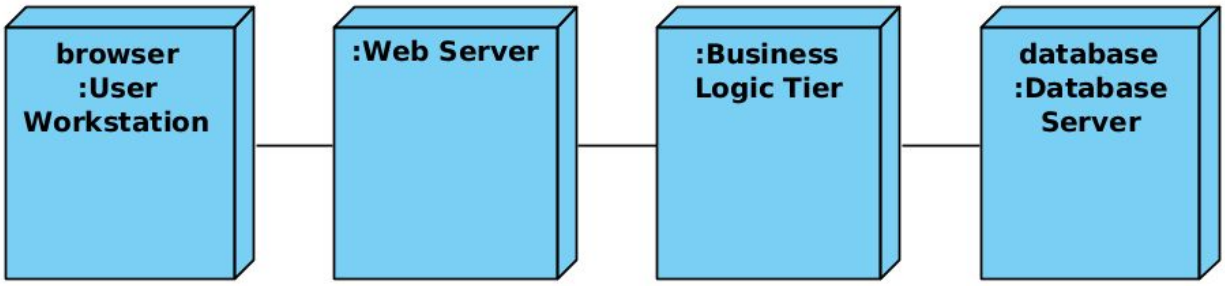

Figure 3.3 Initial Deployment diagram for the CMS

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
User Workstation	The user's browser that connects directly to the client side of the web application.
Web Server	The web server that hosts the server side logic of the application and also serves web pages and content
Business Logic Tier	
Database Server	The server that hosts the existing database.
Relationship	**Description**
Between the browser and the webserver	Communication between the Webclient and the Webserver will be handled by http get/ post requests and responses
Between the webserver and the Logic Tier	API using JSON objects to communicate will be used to interface the webserver with the logic tier
Between the logic tier and the Database	The Logic tier will communicate with the database through the use of sql commands

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-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