Final Project – SOFE3650

Shreyans Rishi 100585817, Randell Roopsingh 100470075,
11/18/2018

Table of Contents

1.0 System Requirements

Requirement elicitation activities had been previously performed, and the following is a summary of the most relevant requirements collected.

1.1 Use Case Model

The use case model in Figure 1.1 presents the most relevant use cases that support the CMS model in the system. Other use cases are not shown.

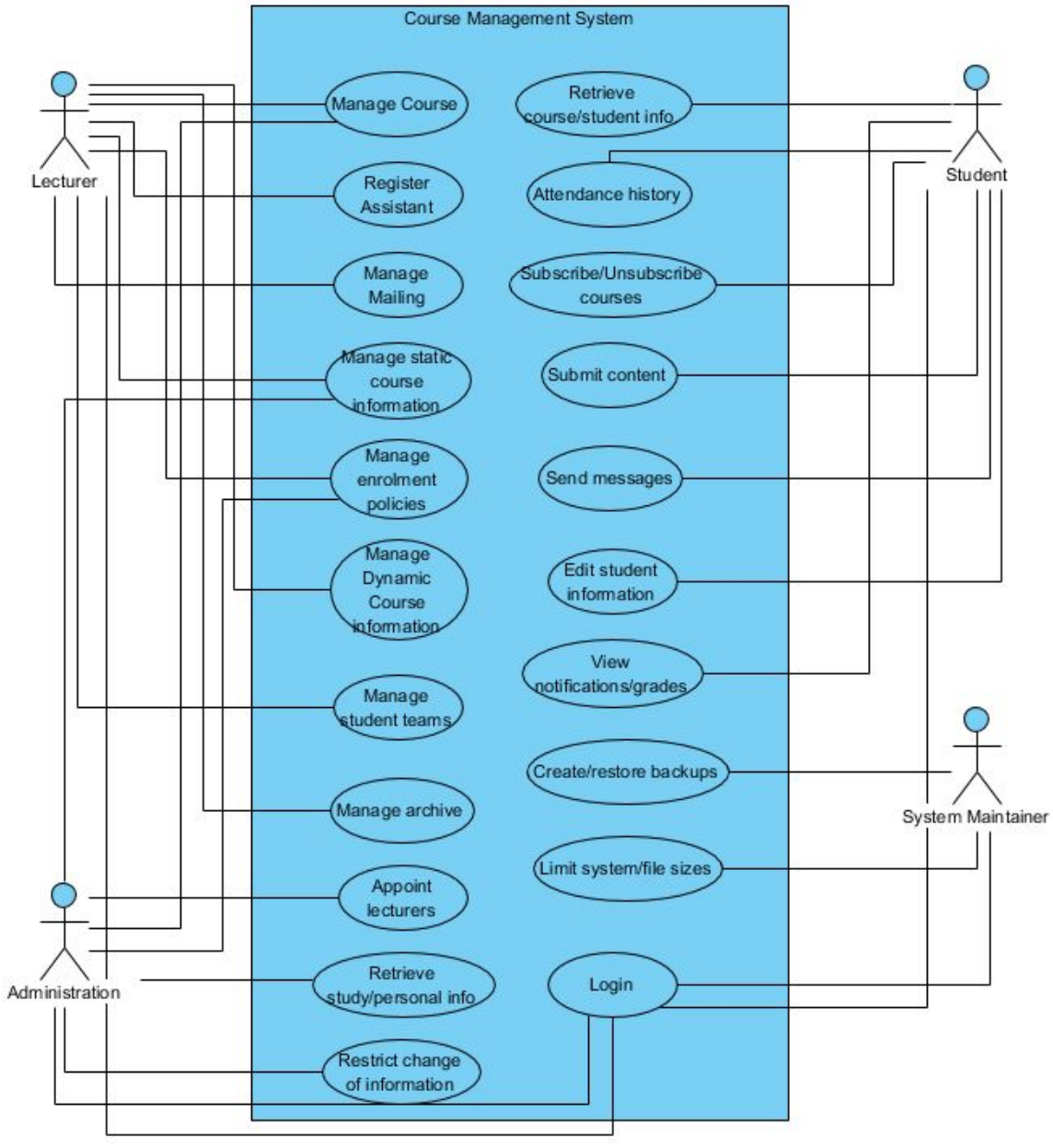


FIGURE 1.1 Use case model for the CMS System

Each of the use cases are described in the following table:

Use Case ID	Use Case	Description
UC-1	Manage Courses	The system should allow the user to view, add, upload and duplicate content to a course. User restrictions to do so should also be allowed.
UC-2	Manage Grading	Appropriate users should be allowed to view, upload, delete or change the grades of a student.
UC-3	Manage Mailing	Appropriate users should be allowed to access and use the mailing system.
UC-4	Manage Static Course Information	The system shall allow the appropriate users to view, add, or edit static course information.
UC-5	Manage Enrolment Policies	The system shall allow the appropriate users to view, add, or edit the enrolment policies.
UC-6	Manage Dynamic Course Information	The system shall allow the appropriate users to view, add, or edit static course information.
UC-7	Manage Student Teams	The system shall allow the appropriate users to view, add, or edit student teams.
UC-8	Manage Archives	Users should be able to view, use, duplicate a course/ system archive.
UC-9	Retrieve Course/Student Information	Appropriate users should be able to view course information or the

		information of a student in that course.
UC-10	Retrieve Attendance History	The system should allow the user to check the attendance history of a student.
UC-11	Subscribe/Unsubscribe Courses	The system should allow students to subscribe/unsubscribe to a course.
UC-12	Submit Content	The system should allow a user to submit course content online.
UC-13	Send Messages	The system should allow a user to send messages to another user.
UC-14	Edit Student Information	The system should allow appropriate users to edit the student information for a course.
UC-15	View Notifications	The system should allow all users to view course/system notifications.
UC-16	Create/Restore Backups	The system should allow the maintainer to create backups and restore system if needed using previous backups.
UC-17	Limit System/File Sizes	The system should allow the maintainer to limit file sizes being uplaoded by the lecturers or students.
UC-18	Appoint Lecturers/Assistant	The system should allow the appropriate users to appoint lecturers/assistant lecturers for a course

UC-19	Retrieve Educational/Personal Information	The system should allow the retrieval of educational or personal information of any user.
UC-20	Restrict change of Information	The system should allow the appropriate users to restrict the information that can be changed by any other uses.
UC-21	Login	The system should allow all users to securely login in to the system and be able to use all the features they are allowed.

1.2 Quality Attribute Scenarios

In addition to these use cases, a number of quality attribute scenarios were elicited and documented in the table below. The associated use case for each scenario is also included.

ID	Quality Attribute	Scenario	Associated Use Cases
QA-1	Performance	Sometimes huge file loads are sent to the system at peak load. The system processes the file size and limits the uploading size and time.	UC-17
QA-2	Modifiability	A lecturer can upload courses and content for it in real time or scheduled.	UC-1
QA-3	Availability	The system should always be up for all users, restricting access depending on user type. If the system goes down it should come back up in a few hours. $hours/month downtime allowed.	UC-16
QA-4	Usability	A user logs in to the system and can pick from a list of options to view, edit, update or upload information.	UC - 9 - UC - 15, UC -18, UC-21
QA-5	Security	The system keeps tracks of changes done by every user and only allows user to change the information they have access to.	UC-20

1.3 Constraints

Finally, a set of constraints on the system and its implementation were collected. They are shown in the below table.

ID	Constraint
CON-1	Limit Downtime to 4h/month. Announce downtime at least 48 hours in advance. Downtime only during low-intensity hours
CON-2	Must be user friendly. Support Dutch and English. Max 3 clicks to reach any content. Single login to access all content. Consistent UI
CON-3	All content must be accessible by disabled users
CON-4	Allow to import calendar and other data
CON-5	Must work and synchronize with secondary universities

1.4 Architectural Concerns

As being a greenfield development and as we are new developers and new to the ADD process the team started off with no architectural concerns.

2.0 The Design Process

We now delve deeper from our requirements to our design.

2.1 ADD Step 1: Review Inputs

The first step of the ADD method involves reviewing the inputs and identifying which requirements will be considered as drivers. The inputs are listed in the below table with details.

Category	Details
Design Purpose	The purpose is to produce an efficient and sufficiently detailed design to support the construction of the Course Management System.
Primary Functional Requirements	The primary use cases were determined to be: UC-1: Because it directly supports the primary requirements of the system UC-4: Because it directly supports the primary requirements of the system UC-6: Because it directly supports the primary

	requirements of the system UC-13: Because it directly supports the primary requirements of the system UC-16: Because of the technical issues associated UC-20: Because of the technical issues associated UC-21: Because of the technical issues associated

Quality Attribute Scenarios: The scenarios were described in Section 1.2. They have now been prioritized as follows:

Scenario ID	Importance to the Customer	Difficulty of Implementation According to the Architect
QA-1	High	High
QA-2	High	Low
QA-3	High	Medium
QA-4	Medium	Medium
QA-5	High	High

Constraints: All of the constraints discussed in Section 1.3 are included as drivers.

Architectural concerns: All of the architectural concerns discussed in Section 1.4 are included as drivers.

2.2 Iteration 1

This section presents the results of the activities that are performed in each of the steps of ADD in the first iteration of the design process.

2.2.1 Selecting Drivers

This is the first iteration of the ADD process on the CMS system. The iteration goal is to establish an overall system structure. The following influence that structure:

QA-1: Performance
QA-2: Modifiability
QA-3: Availability
CON-2: Must be user friendly. Support Dutch and English. Max 3 clicks to reach any content. Single login to access all content. Consistent UI
CON-3: All content must be accessible by disabled users

CON-5: Must work and synchronize with secondary universities

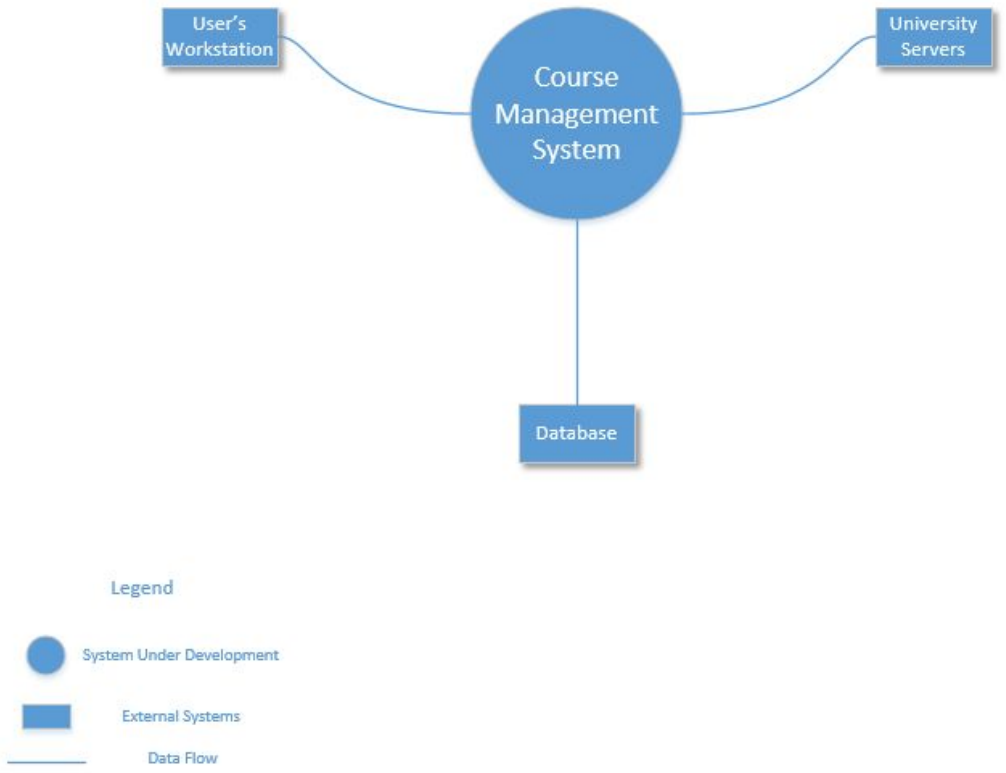


FIGURE 2.1 Context Diagram for the CMS System

2.2.2 Elements of the System to Refine

This is a greenfield development effort (Cervantes, 2016). So the element to refine is the CMS system as a whole. See Figure 2.1 for details on this system. Using the textbook, refinement is done through decomposition.

2.2.3 Design Concepts that Satisfy the Selected Drivers

Design concepts are selected according to Appendix A in FCAPS. In order to establish an overall structure the following design decisions were made:

Design Decisions and Location	Rationale
Logically Structure the client and server part of the system using the Web based application architecture.	The rationale behind selecting this is we do not want to deploy the application on every students machine (CON-2), and students should be able to access this system where ever they want (QA-3 and QA-2). It needs to be accessible over the

<table>
<tr>
<td>.</td>
<td>internet and we want to minimize client-side resources.

<table>
<tr><th>Alternative</th><th>Reason for Discarding</th></tr>
<tr><td>Rich Client Application</td><td>Has to be deployed on users machine and we want portability</td></tr>
<tr><td>Rich Internet Application</td><td>Restricted Access to local resources and out performed when compared to Web based</td></tr>
<tr><td>Mobile application</td><td>The limitations that make this option not viable is the screen size, and resources available.</td></tr>
<tr><td>Service Applications</td><td>Application is used by humans so needs user interface</td></tr>
</table>
</td>
</tr>
<tr>
<td>Physically structure the application using the Four-Tier Deployment</td>
<td>This deployment has the best security as due to accessibility needs, it allows the web server to reside in a publicly accessible network.

<table>
<tr><th>Alternative</th><th>Reason for Discarding</th></tr>
<tr><td>>4 Tier Deployment</td><td>Not necessary for our application. Will increase complexity for little increased performance results</td></tr>
<tr><td>< 3 Tier Deployment</td><td>Not complex enough, need at least three tiers for the web based application as complex</td></tr>
</table>
</td>
</tr>
</table>

	<table><tr><td></td><td>as ours.</td></tr><tr><td>Distributed Deployment</td><td>The downside is that making modifications like adding tiers is expensive</td></tr></table>
Code the User Interface using HTML, PHP, and Javascript	HTML and PHP is easy to code and Javascript allows easy access to server. HTML and Javascript is also very easy to code and make changes.
Creating relational database using MySQL	MySQL is easy to work with HTML and PHP to make connections directly to the database.

2.2.4 Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces
The instantiation design decisions made in this iteration are summarized in the following table:

Design Decisions and Location	**Rationale**
Modify the Business layer of the web application to handle the university logic.	Even though the logic may be similar the school usually needs more fine tuning due to strict course requirements.
Modify the Business logic tier of the four tier deployment to handle the university logic.	Even though the logic may be similar the school usually needs more fine tuning due to strict course requirements.

2.2.5 Sketch Views and Record Design Decisions

The diagram in Figure 2.2 shows the sketch of the module view of the two reference architectures that were selected for the client and server applications. These have now been adapted according to the design decisions we have made.

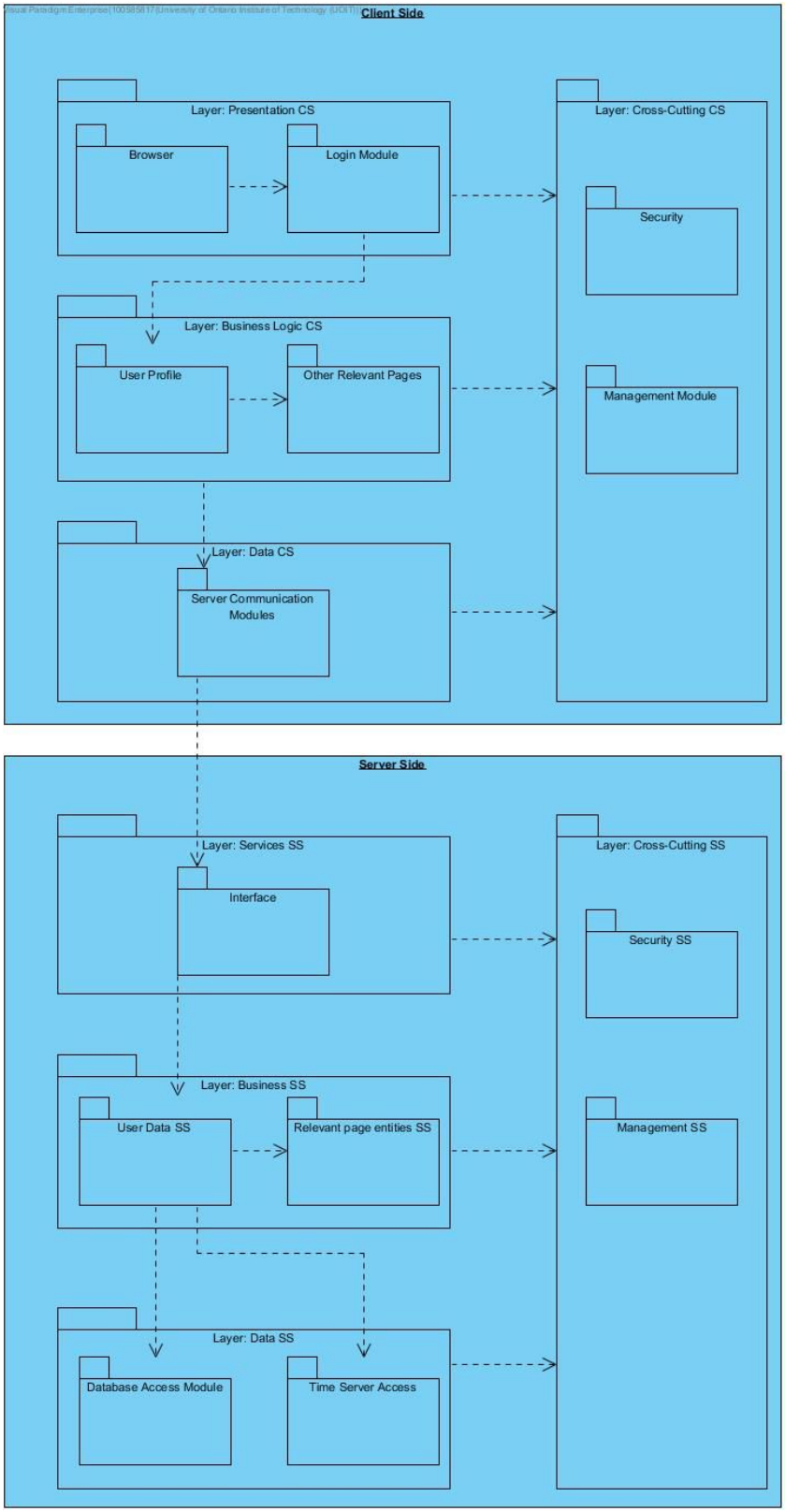


FIGURE 2.2 Sketch of a module view of client side and server side architecture

The following table summarizes the information that is captured in Figure 2.2:

Element	**Responsibility**
Presentation CS	This layer contains all the modules that are related to the user interactions with the system.
Business Logic CS	This layer contains all the modules that are related to the business logic operations of the system and can be executed locally on the client side.
Data CS	This layer contains all the modules that are responsible to initiate communication with the server.
Cross-Cutting CS	This layer ensures the security and manages all the different operations happening in the other layers.
Browser CS	This layer allows the user to be able to access the user interface created to easily access the system from anywhere.
Login Module CS	This is the user interface that receives user login information as inputs.
User Profile CS	This layer has modules that perform business operations on the user side based on the input received by the database/server.
Other Relevant Pages CS	This layer has other relevant modules that the user need to perform various other tasks and requires some kind of interaction directly from the database/server.
Server Communication Modules	This layer has modules that are used to connect the client side to the server side for data transfer.
Services SS	This layer has all the modules that are being consumed by the user on the client side.
Business Logic SS	This layer processes all the modules in the Business Logic CS layer, on the server side.
Data SS	This layer processes all the data that is

	required by the client side after the communication has been initiated by the time servers.
Cross-Cutting SS	This layer ensures the security and manages all the different operations happening in the other layers of the server side.
Interface SS	This layer contains all the modules that are required to expose services that are consumed by the users on the client side.
User Data SS	This layer contains all the modules that are required to get and return the user information to the client side.
Relevant Page Entities	This layer contains all the modules that are required to get and return all other entities for every other services being used by the user on the client side.
Database Access Module	This module is responsible to make sure that all the information that is being sent to or received by the database is persistent.
Time Server Access	This module handles all the communication with the servers made by the client.

The deployment diagram in figure 2.3 shows how the individual components associated with the previous diagram are deployed. The responsibilities of the elements in the diagram are summarized as follows:

Element	**Responsibility**
User's Workstation	The user's devices, that can host the client side logic of the system.
Application Server	The server that can host the server side logic of the web application.
Database Server	The server that contains and hosts the relational database of the system
Time Server	All External Time Servers

Also information about relationships between some elements in the diagram that is worth recording is summarized in the following table:

Relationship	Description
Between web/app server and database server	Communication with the database will be done using MySQL protocol
Between web/app server and time server	The SNMP protocol is used (at least initially)

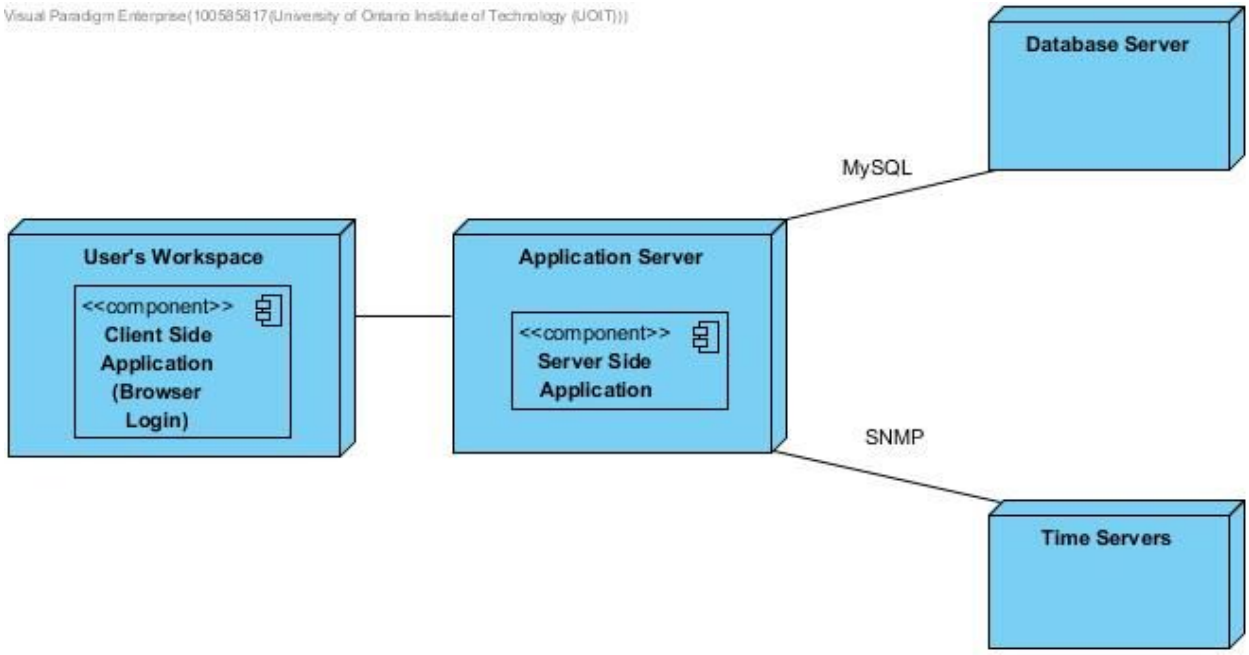


FIGURE 2.3 Deployment Diagram for the CMS system

2.2.6 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

The following table summarizes the design progress using the Kanban board technique.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
	UC-1		User restriction is now possible with SQL database. Other user interface features not yet implemented.
	UC-4		User restriction is now possible with SQL database. Other user interface features not yet implemented.

	UC-6		User restriction is now possible with SQL database. Other user interface features not yet implemented.
UC-13			No structure has been implemented that allows for this feature yet.
	UC-16		SQL database allows for system backups as needed.
	UC-20		Authorized users will have access to the SQL table containing users and their access. This means authorized users can make changes to other users access.
	UC-21		By using an sql database it allows for a table to store user ids and the restrictions they have. This allows us to meet this use case.
	QA-1		Because we are using a web based application with four tiers and sql database large files will be no issue.If it need to be restricted due to load then it is easy to do so. .
	QA-2		Because we are using a web based application with four tiers and sql database multiple users can interact with the system at once and change files.
	QA-3		Because we are using a web based application and sql database the only downtime would be for system maintenance and significant changes.
	CON-2		Web based user interface checks off all items in this constraint
	CON-3		By using a web based application we have taken the first step in this process. We now have to add the correct packages to allow complete

			addressing of this constraint
		CON-5	Web based application is universal the only thing is access to the secondary universities database. so easily can sync with other universities who support SQL databases

3.0 Iteration 2: Identifying Structures to Support Primary Functionality

This section presents the results of the activities that are performed in each of the steps of the ADD in the second iteration of the CMS system.

3.0.1 Selecting Drivers

The goal of this iteration is to address the general architectural concern of identifying structures to support primary functionality.

In this second iteration the architect considers the system's primary use cases:

UC-1: Manage Courses
UC-4: Manage Static Course Information
UC-6: Manage Dynamic Course Information
UC-13: Send Messages
UC-21: Login

3.0.2 Elements of the System to Refine

The elements that will be refined in this iteration are the Web based application architecture, the User Interface and the MySQL relational database. In order to fully support the functionality, there will be collaboration of all the layers within these architectures

3.0.3 Design Concepts that Satisfy the Selected Drivers

The following table summarizes the design decisions.

Design Decisions and Location	**Rationale and Assumptions**
Create a Domain Model for the application	Before starting a functional decomposition we must create an initial domain model for the system. We must identify major entities in the domain and their relationships. No other alternatives were considered as this

	is the foundation.
Identify Domain Objects that map to functional requirements.	Each distinct functional element of the application needs to be encapsulated in a self-contained building block -- a domain object. You could skip the consideration of domain objects but you risk requirement considerations.
Decompose Domain Objects into general and specialized components.	Domain objects represent complete sets of functionality. “Components” are referred to as modules. There are no good alternatives.
Use Spring Framework and Hibernate	Spring is a widely used framework to support enterprise application and Hibernate integrates well with Spring. Other alternatives were considered in JEE and other ORM frameworks but due to FCAPS and the fact that there was no real push to select a specific one so Spring and Hibernate were chosen.

3.0.4 Instantiate Architectural Elements, Allocate Responsibilities, and define Interfaces

The following table contains the design decisions made:

Design Decisions and Location	**Rationale**
Create only an initial domain model	Primary use case entities need to be identified and modeled but only an initial domain model is created to accelerate this phase of the design
Map the system to use cases to domain objects	Initial identification of domain objects can be made with the system use cases.
Decompose the domain objects across the layers to identify layer specific modules with an explicit interface	This technique ensures that modules support all functionalities. This does arise a architectural concern CRN-1: A majority of modules shall be unit tested.

Connect components associated with modules using Spring	This allows different aspects to be supported and modules to be unit tested (CRN-1).
Associate frameworks with a module in data layer	ORM mapping is encapsulated in the modules that are contained in the data layer.

3.0.5 Sketch Views and Record Design Decisions

As a result of the decisions made in step 5, several diagrams are created.

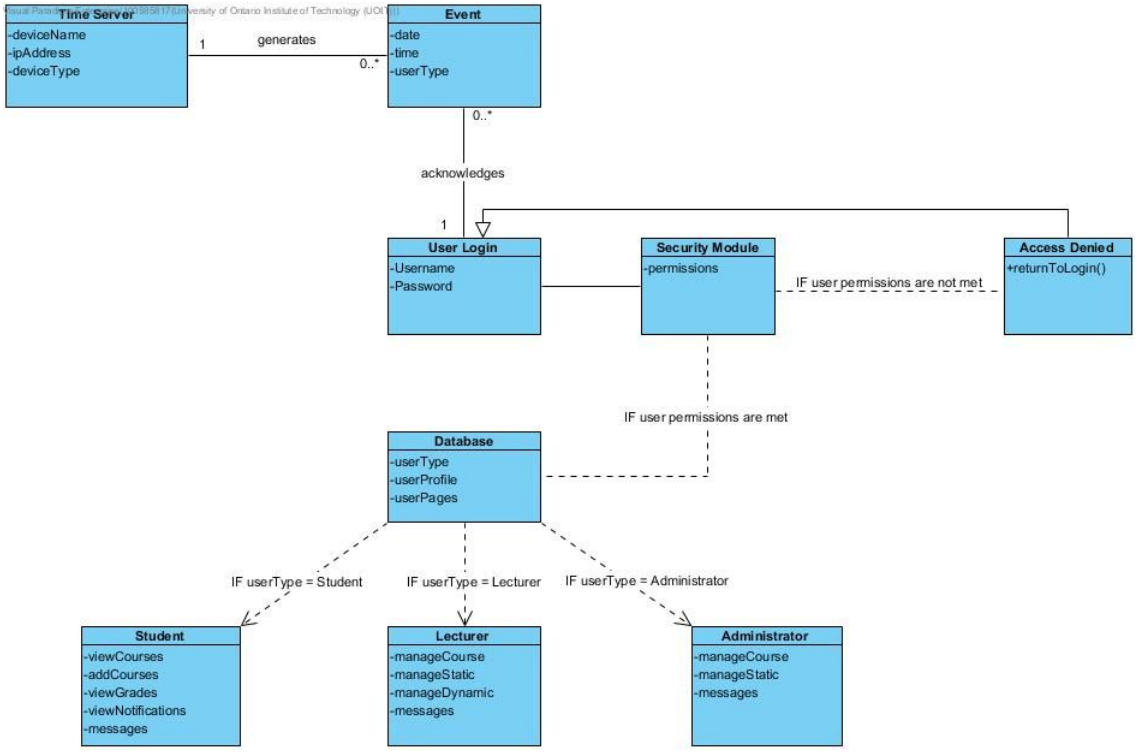


FIGURE 3.1 Initial Domain Model

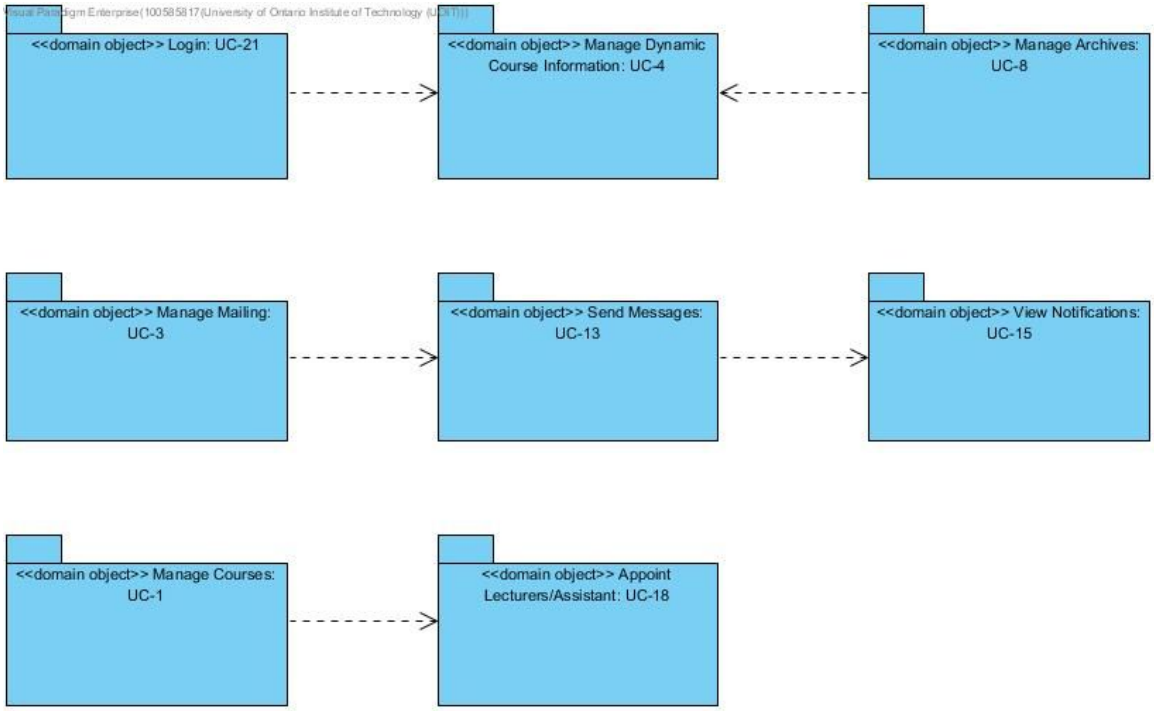


FIGURE 3.2 Domain Objects associated with the use cases

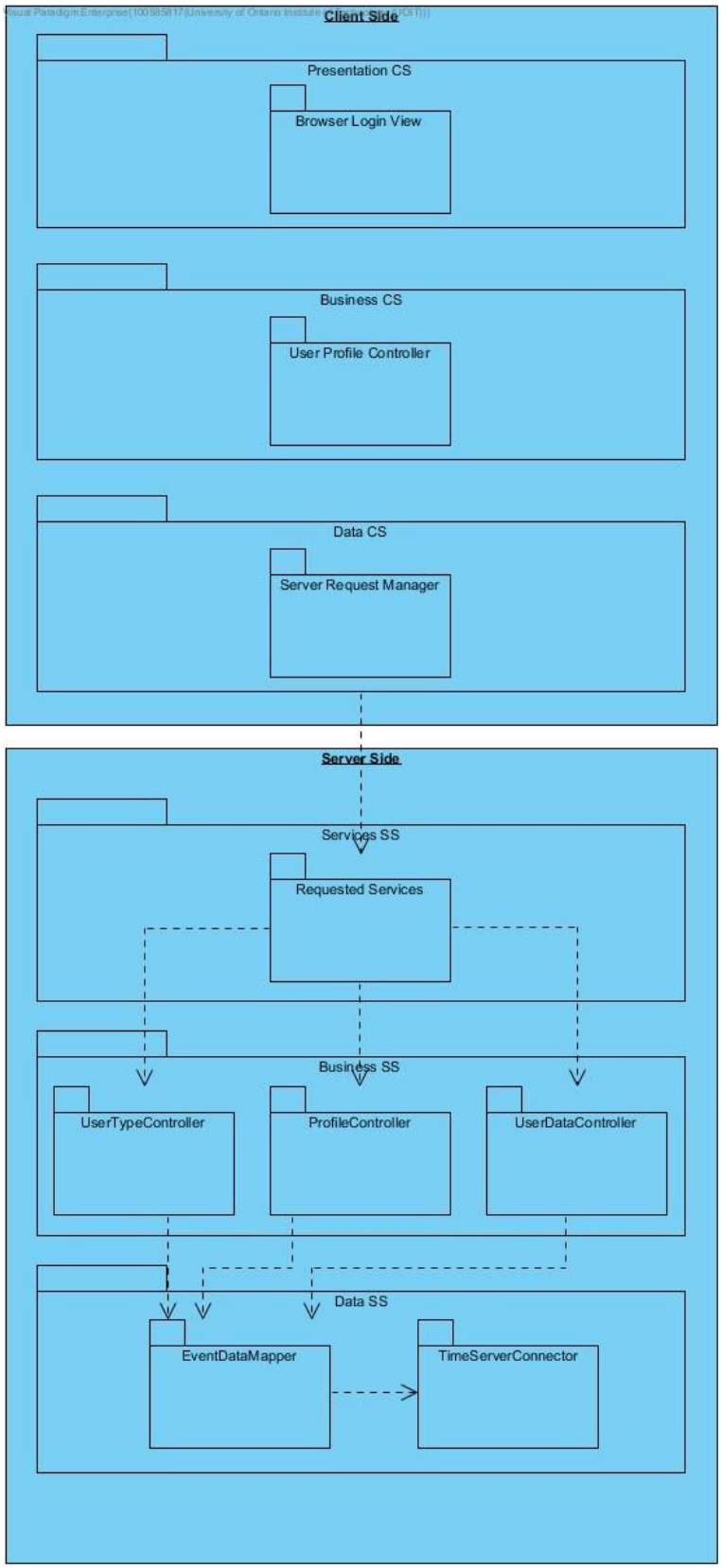


FIGURE 3.3 Modules that support primary use cases

Element	Responsibility
Browser Login View	Displays the login view that asks user for username and password inputs.
User Profile Controller	Displays the user profile information after login is successful.
Server Request Manager	Responsible for communication with the server-side logic
Requested Services	Provides a facade that receives the requested services from the clients.
UserTypeController	Contains business logic that contains the user type information.
ProfileController	Contains business logic that contains user profile information.
UserDataController	Contains business logic that contains user data for the other web pages like manage course, etc.
EventDataMapper	Responsible for the persistence of operations related to the events requested.
TimeServerConnector	Responsible for the communication of the system with the time servers.

Figure 3.4 shows an initial sequence diagram for UC-21 (user logs into system). It shows how a lecturer would start up and login to the system as well as the response from the system.

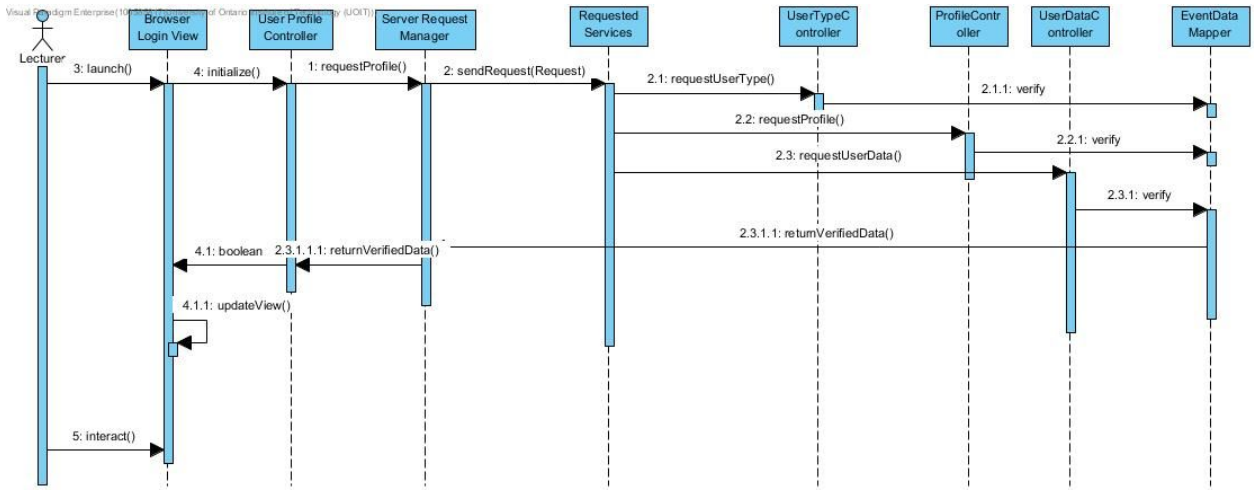


FIGURE 3.4 Sequence Diagram for UC-21

From the interactions identified in the sequence diagram, initial methods for the interfaces of the interacting elements can be identified:

Element: User Profile Controller
bool initialize(): Opens the network representation of the user profile so the user can interact with it.
updateView(): The profile view is update after receiving verified data from the server.

Element: Server Request Manager
requestProfile(): All profile related information is requested to the server.

Element: Requested Services
sendRequest(Request): All the requested services needed are received by the server

Element: UserType, Profile and UserData Controller
requestUserType(), requestProfile(), requestData(): All of the necessary data is sent to the next element.

Element: EventDataMapper
Verify: All the requested data from the database is verified.
returnVerifiedData(): All verified data above is returned back to the client.

3.0.6 Perform Analysis of Current Design and Review Iteration

The decisions made in this iteration provided an initial understanding of how functionality is supported in the system.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		UC-1	Modules across the layers and preliminary interfaces to support this use case have been identified
		UC-4	Modules across the layers and preliminary interfaces to support this use case have been identified
		UC-6	Modules across the layers and preliminary interfaces to support this use case have been identified

	UC-13		Modules across the layers and preliminary interfaces to support this use case have been identified
		UC-16	Modules across the layers and preliminary interfaces to support this use case have been identified
		UC-20	Modules across the layers and preliminary interfaces to support this use case have been identified
		UC-21	Modules across the layers and preliminary interfaces to support this use case have been identified
	QA-1		The elements that support the associated use case (UC-17) have been identified.
		QA-2	The elements that support the associated use case(UC-1) have been identified.
	QA-3		The elements that support the associated use case(UC-16) have been identified.
		CON-2	User interface modules have been identified
	CON-3		User interface modules have been identified
		CON-5	No relevant decisions were made
	CRN-1		New architectural concern was introduced in this iteration. It was also partially handled by Connect components associated with modules using Spring

4.0 Iteration 3: Addressing Quality Attribute Scenario

This section presents the results of the activities that are performed in each of the steps of ADD in the third iteration of the design process.

4.0.1 Selecting Drivers

For this iteration, the architect focuses on QA-3 quality attribute scenario: Availability- The system should always be up for all users, restricting access depending on user type. If the system goes down it should come back up in a few hours. 4 hours/month downtime allowed.

4.0.2 Elements of the System to Refine

For this scenario, the elements that will be refined are the physical nodes that were identified during the first iteration.

- Application Server
- Database Server

4.0.3 Design Concepts that Satisfy the Selected Drivers

The design concepts used in this iteration are the following:

Design Decisions and location	**Rationale and Assumptions**
Introduce the active redundancy tactic by replicating the application server and other critical components such as the database	By replicating elements if there is any down time the back ups can be used to then a comparison between databases can be done to ensure changes during down time are not lost

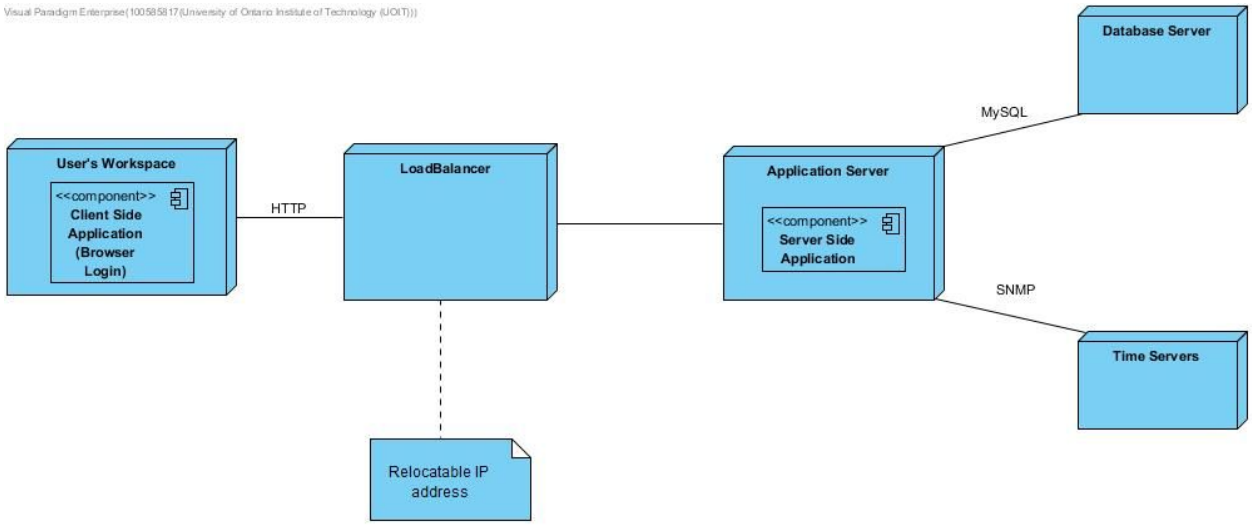
4.0.4 Instantiate Architectural Elements, Allocate Responsibilities, and define Interfaces

The instantiation design decisions are summarized in the following table:

Design Decisions and location	**Rationale**
Use active redundancy and load balancing in the application server	Because two replicas of the application server are active at any time, it makes sense to distribute and balance the load among the replicas. This tactic can be achieved through the use of the Load-Balanced Cluster pattern. This introduces a new architectural concern CRN-2:Manage state in replicas.
Implement load balancing and redundancy using technology support	Many technological options for load balancing and redundancy can be implemented without having to develop an ad hoc solution that would be less mature and harder to support.

4.0.5 Sketch Views and Record Design Decisions

The following table describes responsibilities for elements that have not been previously listed:



Element	Responsibility
LoadBalancer	Dispatches requests coming from clients to the application servers. The load balancer also presents a unique IP address to the clients.

The UML sequence diagram shown in Figure 4.2 illustrates how the LoadBalancer that was introduced in this iteration dispatches requests to support QA-3.
As the purpose of this diagram is just as an illustration, the names of the methods are just temporary.

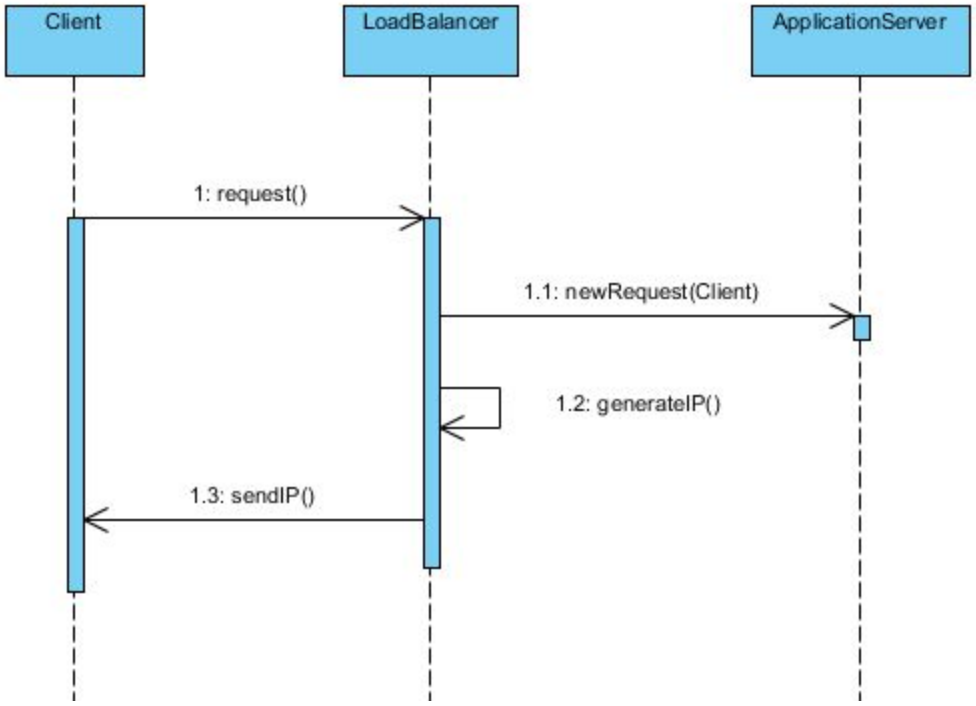


FIGURE 4.2 LoadBalancer

4.0.6 Perform Analysis of Current Design and Review Iteration

In this iteration important design decisions were made to address QA-3. The following table summarizes the status of the different drivers and the decisions made. Drivers that were completely addressed were removed.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
	UC-13		No relevant decisions were made.d
	QA-1		No relevant decisions were made.
		QA-3	With redundancy we significantly limit down time and reduce probability of failure.
	CON-3		No relevant decisions were made.
	CRN-1		No relevant decisions were made.
CRN-2			This is new and was introduced this iteration. At this point no relevant

			decisions were made.