Step1: Review Inputs

Design Purpose:

The CMS is designed by considering it a Greenfield system for a mature domain. The purpose of the design is the produce a detailed architecture for the implementation of the CMS.

Primary Functional Requirements:

The following use-cases as defined in the “Use Cases.pdf” are used as primary functional requirements:

UC-1: Display course information
UC-3: Subscribe/Unsubscribe courses
UC-7: Manage courses
UC-8: Manage Users

Quality attribute Scenarios:

Scenario ID	Importance to customer	Difficulty to implement according to architecture
QA-1	HIGH	HIGH
QA-2	LOW	MEDIUM
QA-3	HIGH	LOW
QA-4	MEDIUM	MEDIUM
QA-5	MEDIUM	LOW
QA-6	LOW	MEDIUM
QA-7	HIGH	HIGH

Constraints:

The “Quality attributes and constraints.pdf” file contains the relevant constraints for the CMS.

Step2: Establish iteration goal by selecting drivers

Drivers	Input
QA-1	Security
QA-4	Interoperability
QA-7	Performance
All constraints from the “Quality attributes and constraints.pdf” file	Defined by each constraint

Step 3: Choose one or more Elements to refine

The only element to refine is the entire CMS System since this is a Greenfield development effort.

Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

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

Design Decision and Location	Rationale
Logically Implement the client of the system using the Web Application reference architecture	Development of an applications initiated on a web browser, communication with a server through the HTTP protocol, are supported by the Web Application Reference structure. Web applications are simple to develop, they are easily accessible on multiple platforms, and they are scalable and maintainable.
Logically structure the server part of the system using service application reference structure	The service application would fetch or send data, it would also keep a backup. It is not a presentation layer rather a service used by other applications.
Physically structure the application using the three-tier deployment pattern	The CMS needs to be access through a browser, while data being fetched and saved, for this task the three-tier deployment pattern is the most suitable approach.

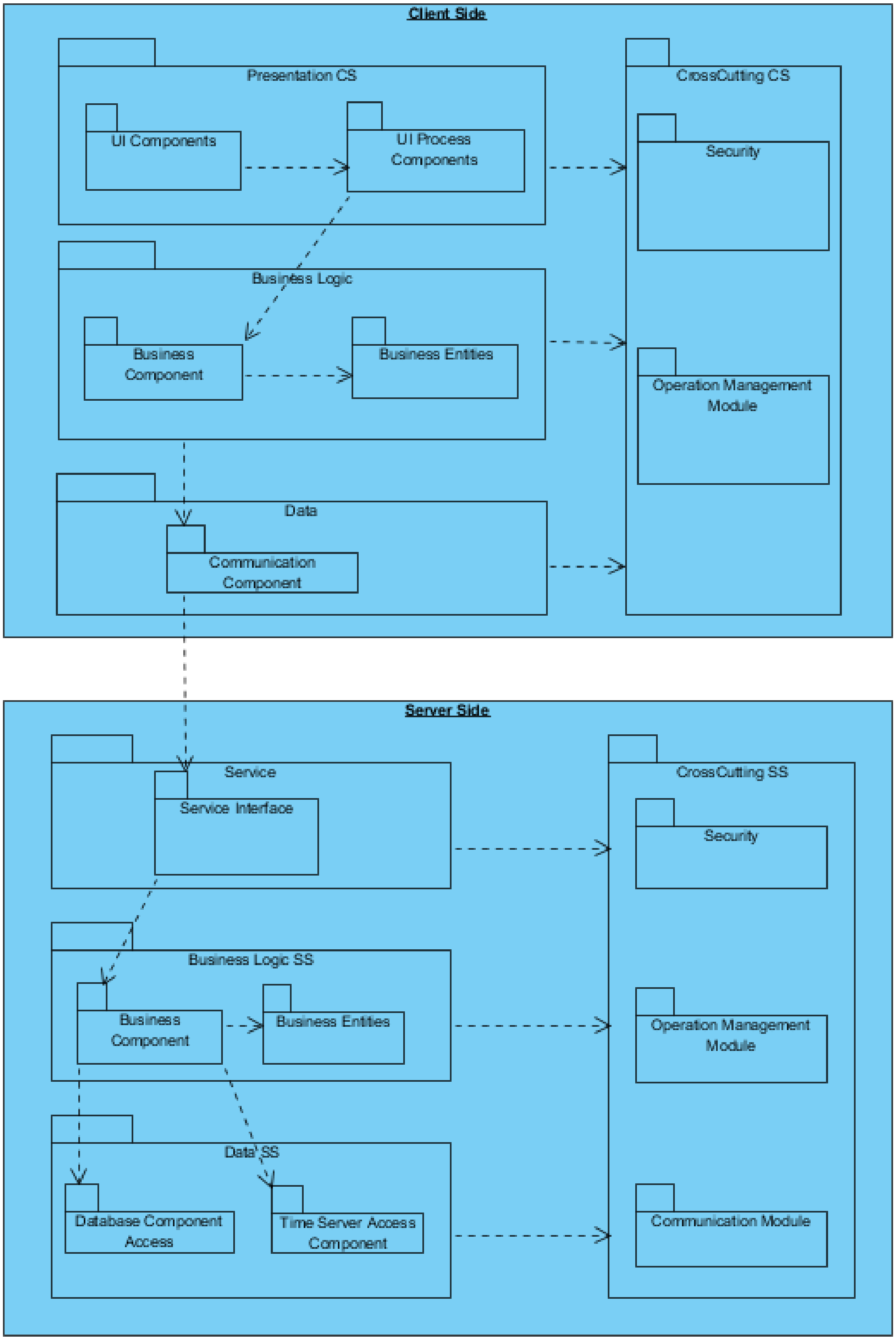
Alternative	Reason for discarding
Pipe architecture system	The CMS system is accessed by multiple actors which then interact with a centralized database to perform relevant tasks. A pipe architecture system does not allow an efficient approach for a client-server based system.
Layered Application system with client and server as the primary layers	A layered application system will introduce unnecessary overhead and complexity affecting the non-functional requirements of the system. We will still use a layered based approach for the client and business tiers, but the overall system will not be entirely layer based.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities

Design Decision and Location	Rationale
Create a module dedicated to accessing the secondary university system	This will help achieve QA-4
Create a UI	This will help achieve QA-7

Step 6: Sketch Views and Record Design Decisions

The diagram below shows the two reference architectures that were selected for client and server applications.


Client Side
Presentation CS
UI Components
UI Process Components
CrossCutting CS
Security
Business Logic
Business Component
Business Entities
Operation Management Module
Data
Communication Component
Server Side
Service
Service Interface
CrossCutting SS
Security
Business Logic SS
Business Component
Business Entities
Operation Management Module
Data SS
Database Component Access
Time Server Access Component
Communication Module

Element	Responsibility
Presentation Client Side (CS)	Components that control user interactions and use case control flow are contained in this layer
Business Logic CS	Components that perform business logic operations that can be executed logically on the client side are contained in this layer.
Data CS	This layer contains components that are responsible for communication with the server
Crosscutting CS	This layer contains components with functionality that goes across different layers
UI Component	The UI is rendered in this component and it also receives user inputs.
UI Process Component	This component is responsible for control flow of all the system use cases
Business Modules CS	This component either implements business operations that can performed locally or from the server side
Business Entities CS	The entries from the business domain and their associated business logic are represented in this component
Communication Component CS	Handles communication across layers and tiers
Server Side (SS)	Contains components that expose services that are consumed by clients
Business Logic SS	Contains components that perform business logic operations that require processing on the server side
Data SS	This component contains a responsible data persistence and provide common operations used to retrieve and store information.
Crosscutting SS	This layer contains components with functionality that goes across different layers
Service Interface SS	Exposes services that are consumed by clients
Business Modules SS	Implement business operations
Business Entities SS	These entities make up the domain model
Database Access Component	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.

Step 7: Perform analysis of current design and review iteration goal and design objectives

Not addressed	Partially Addressed	Completely Addressed
		QA-1
	QA-4	
		QA-7
CON-1		
		CON-2
CON-3		
		CON-4
	CON-5	
	CON-6	
		CRN-1
	CRN-2	
CRN-3		