Software Design and Architecture:
Deliverable 2 - Iteration 1
CRN: 43509 - Group #33

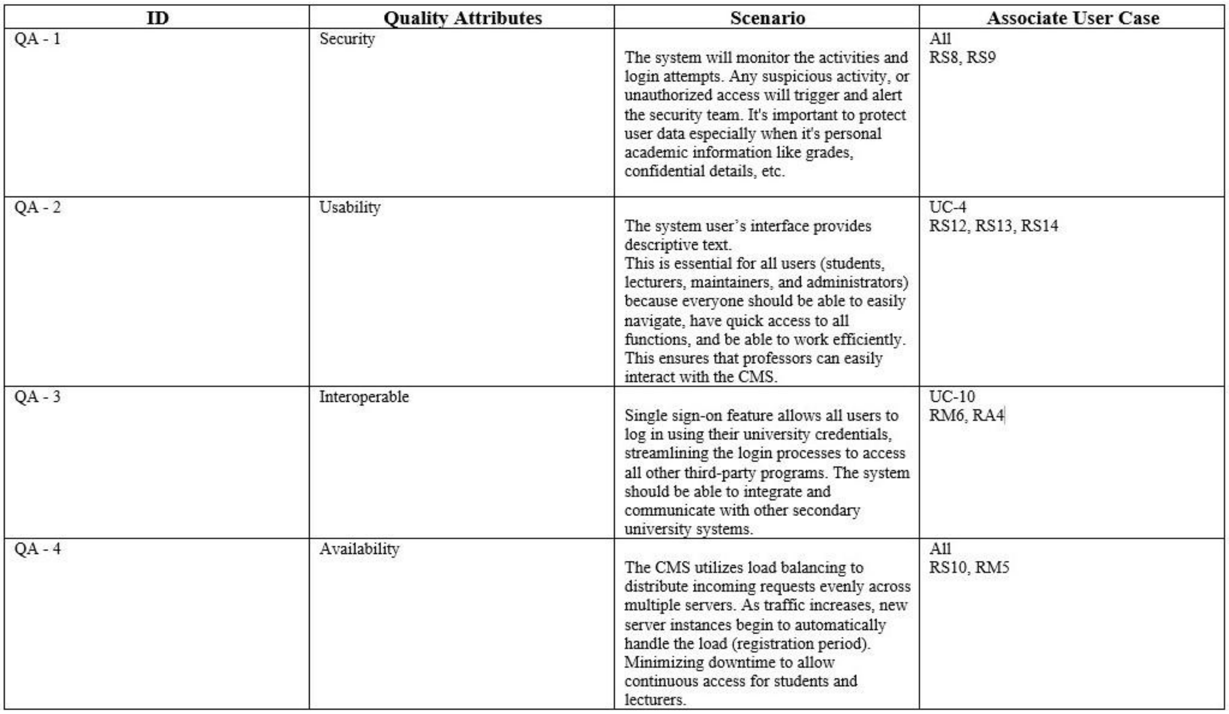
SOFE 3650U

Members:
Rida Siddiqi (100825212)
Navdeep Virdi (100827233)
Kevin Cao (100847959)

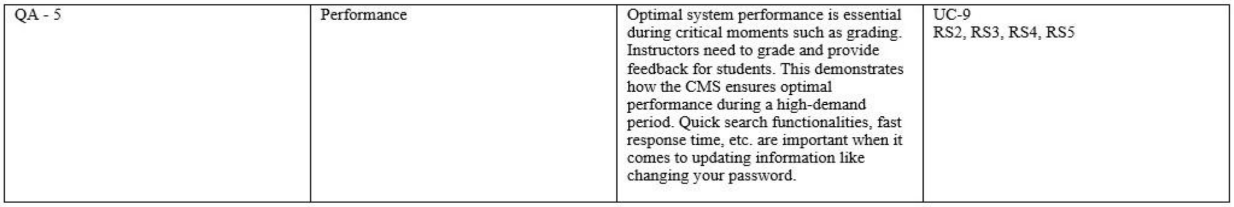
Step 1: Review Inputs

Use Case	Description
R1	The system will provide course information to the participants involved. In this case, the system will forward the data to the students, lecturers, and administration.
R2	System will store course information. Anything of importance regarding the student's education will be stored in a database.
R3	System will represent course information. This would allow the system to be interoperable.

Quality Attributes :



ID	Quality Attributes	Scenario	Associate User Case
QA - 1	Security	The system will monitor the activities and login attempts. Any suspicious activity, or unauthorized access will trigger and alert the security team. It's important to protect user data especially when it's personal academic information like grades, confidential details, etc.	All RS8, RS9
QA - 2	Usability	The system user's interface provides descriptive text. This is essential for all users (students, lecturers, maintainers, and administrators) because everyone should be able to easily navigate, have quick access to all functions, and be able to work efficiently. This ensures that professors can easily interact with the CMS.	UC-4 RS12, RS13, RS14
QA - 3	Interoperable	Single sign-on feature allows all users to log in using their university credentials, streamlining the login processes to access all other third-party programs. The system should be able to integrate and communicate with other secondary university systems.	UC-10 RM6, RA4
QA - 4	Availability	The CMS utilizes load balancing to distribute incoming requests evenly across multiple servers. As traffic increases, new server instances begin to automatically handle the load (registration period). Minimizing downtime to allow continuous access for students and lecturers.	All RS10, RM5



QA - 5	Performance	Optimal system performance is essential during critical moments such as grading. Instructors need to grade and provide feedback for students. This demonstrates how the CMS ensures optimal performance during a high-demand period. Quick search functionalities, fast response time, etc. are important when it comes to updating information like changing your password.	UC-9 RS2, RS3, RS4, RS5

<table>
<tr><th>Category</th><th>Details</th></tr>
<tr><td>Purpose of Design</td><td>Since this is a greenfield system, the primary purpose is to create a detailed design that will support the construction of the system.</td></tr>
<tr><td>Functional Requirements</td><td>Primary Functions:
UC-1: Signing up for the course.
UC-2: Creating the team.
UC-7: Backup/Restoration.</td></tr>
<tr><td><b>Quality Attribute Scenarios</b></td><td>
<table>
<tr><td>Scenario ID</td><td>Importance</td><td>Difficulty to Implement</td></tr>
<tr><td>QA-1</td><td>High</td><td>Medium</td></tr>
<tr><td>QA-2</td><td>Medium</td><td>High</td></tr>
<tr><td>QA-3</td><td>Medium</td><td>Medium</td></tr>
<tr><td>QA-4</td><td>High</td><td>Low</td></tr>
<tr><td>QA-5</td><td>Medium</td><td>Medium</td></tr>
</table>
</td></tr>
</table>

Step 2 - Select inputs that will be in the iteration

It’s important to note that the first architectural concern must be achieved.
CRN-1: Establishing an overall system structure.

QA-1: Availability
QA-2: Security
QA-3: Usability
QA-4: Functionality
CON-1: Leverage the efficiency and retrieval of the system
CON-3: User access and permissions
CON-5: Core usability attribute
CRN-2: Attach team's knowledge and experience in framework and other technologies

Step 3 - Choose elements of the system to decompose

The first iteration consists of a developed greenfield system. Since there are no previous designs, the whole system will be decomposed.

Step 4 - Choose design concepts that satisfy the input criteria

Design decision	Rationale
Logically implement the system's client using Web Application reference architecture	The Web Application reference structure facilitates the creation of applications that are launched in a web browser and use the HTTP protocol to communicate with a server. Since we want users to be able to access the web application through a web browser, the Web Application references structure is the best way to implement it, with the browser running on the client computer.
Logically structure the system's server using a service application reference structure	The service application will allow the web application to have access to retrieve any necessary data and to push updates.
Build the client's interface using HTTP and the angular framework	HTTP + react is a standard method for developing user-friendly scalable web applications
Deploy the application servers	This custom deployment solution consists of a secondary database server and a Linux server dedicated to the web server and business logic tier. A web server that uses Apache for load balancing, redundancy servers, and content delivery uses Golang as its backend.

Step 5 - Interface and Responsibilities

Interface	Responsible
Single responsibility principle	The class or module must have only one reason to change. Each of the classes or modules should remain the same until one reason is given to change and a part of the systems' functionality.
High cohesion, low coupling	Responsibilities are closely grouped and the dependencies are between separate parts of the system. In this case, the course manager class has responsibilities that are very similar and work together but each of the dependencies are on separate parts of the course manager class.
Coherence	The responsibilities are related logically and make a coherent and comprehensible design. The course manager has multiple roles (student, lecturer, admin, and system maintainers) that hold responsibilities to create the course manager class design.
Standardization	This creates a standardized way to communicate and interact between the different parts of the system.
Encapsulation	Accomplishing encapsulation will separate the statements of methods and allow the methods to be implemented.
Multiple inheritance	Components in the classes can be implemented to different interfaces because that would allow different interactions and behaviors from the course manager class.

Step 6 - Record Design Decisions

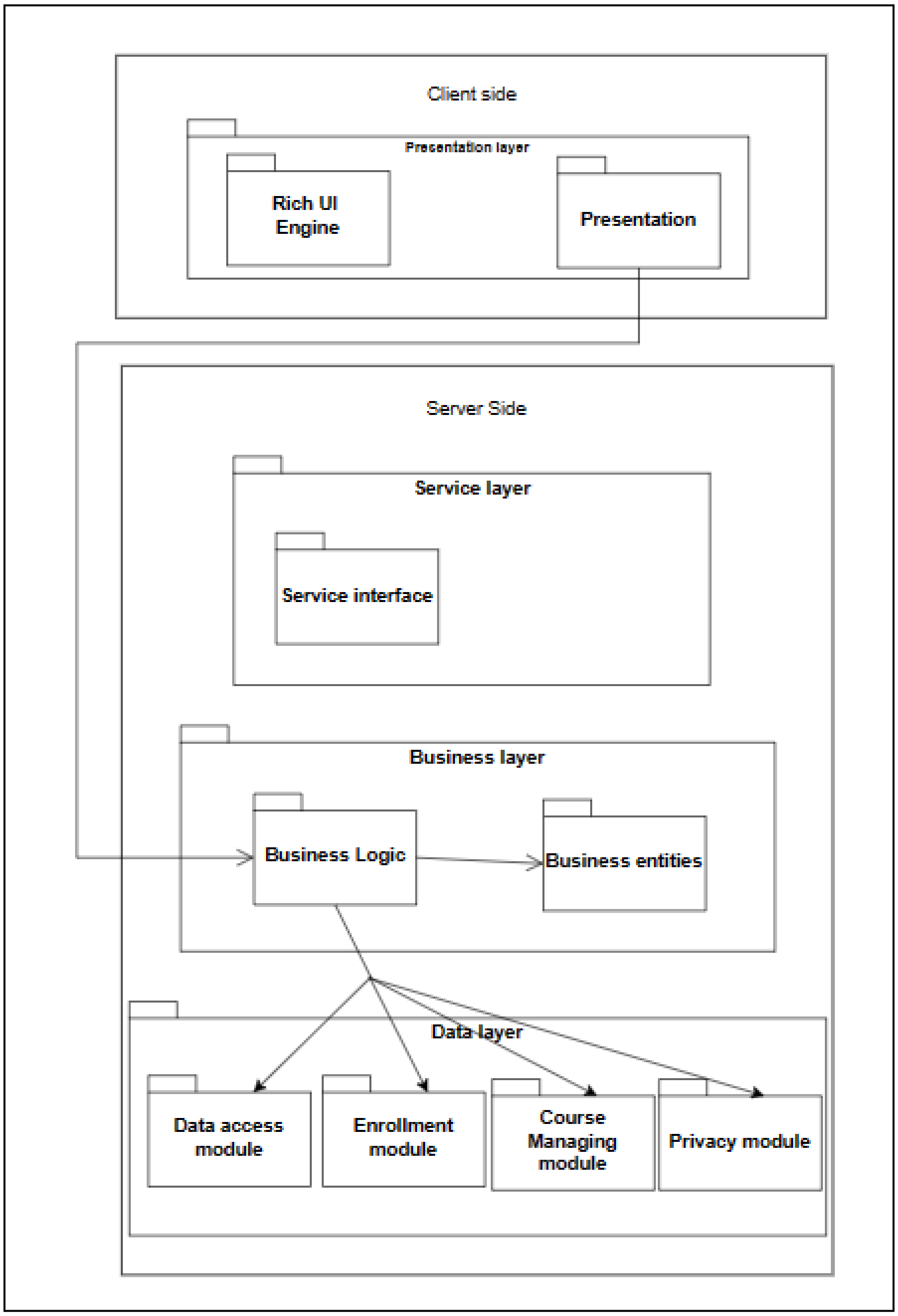


Figure 1: module view for client and server application view

Element	Responsibility
Presentation layer	The presentation takes care of all client-side operations.
Services Server	This layer contains modules that are consumed by the clients

side	
Service interface	These modules contain services that are consumed b the client
Business layer	This layer content modules that perform business logic operations
Business modules	These modules implemented business operations
Business Entities	These entities make up the domain model
Data layer	This layer contains modules that are responsible for data persistence and for communication with the time servers
Data Access Modules	This module is responsible for the persistence of the business entities into the relational database. It performs object-oriented to relational mapping and shield the rest of the application from persistence detail
Enrollment modules	These module is responsible for the enrollment of client when registering into course
Course managing modules	This module contain all the information of the available course for client to enroll
Privacy modules	These modules protect the privacy of the system information.

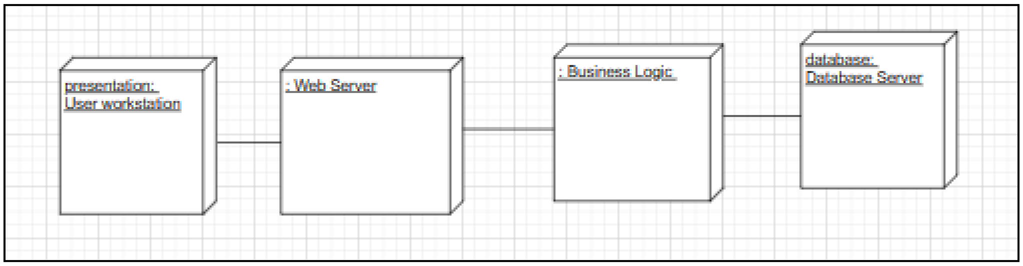
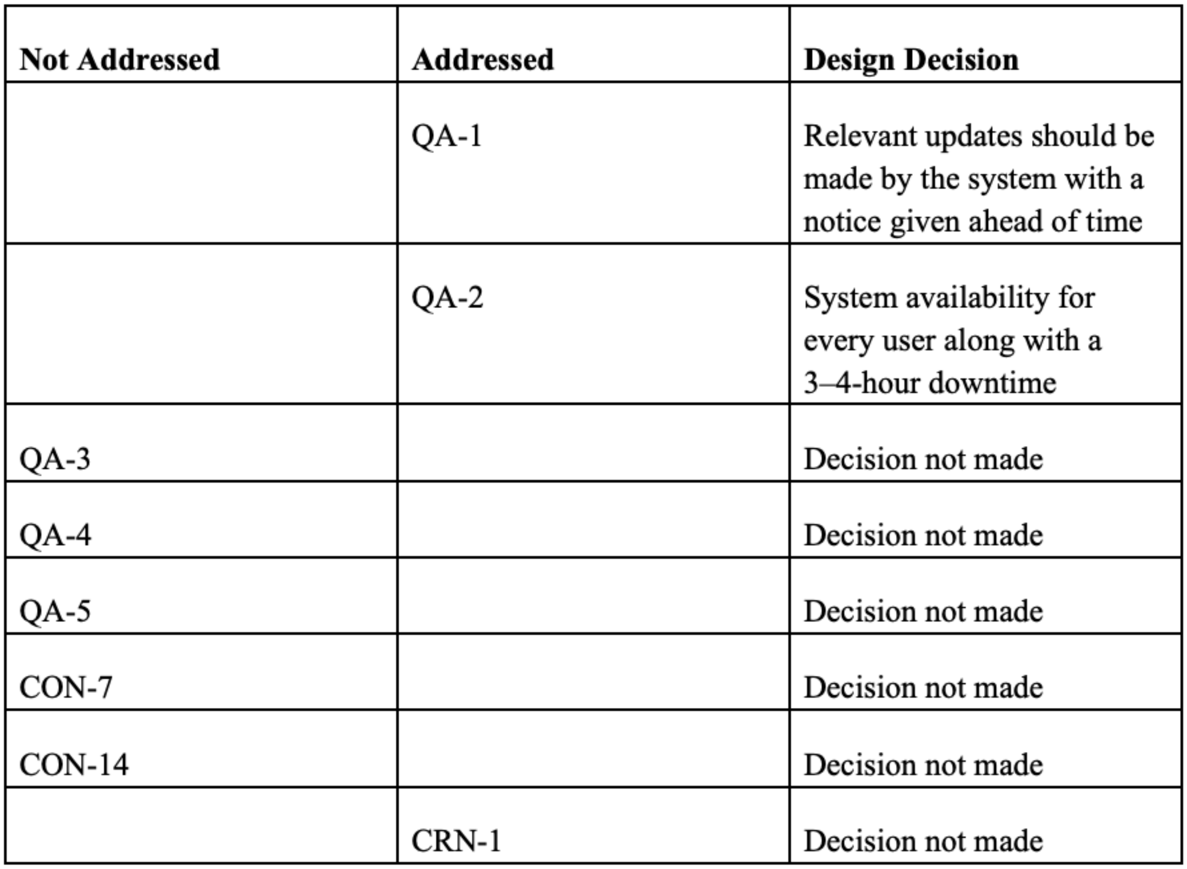


Figure 2: Initial Deployment diagram for the CMS

Step 7 - Analysis:



Not Addressed	Addressed	Design Decision
	QA-1	Relevant updates should be made by the system with a notice given ahead of time
	QA-2	System availability for every user along with a 3–4-hour downtime
QA-3		Decision not made
QA-4		Decision not made
QA-5		Decision not made
CON-7		Decision not made
CON-14		Decision not made
	CRN-1	Decision not made