Architecture Final Project

*CMS ADD Iterations*

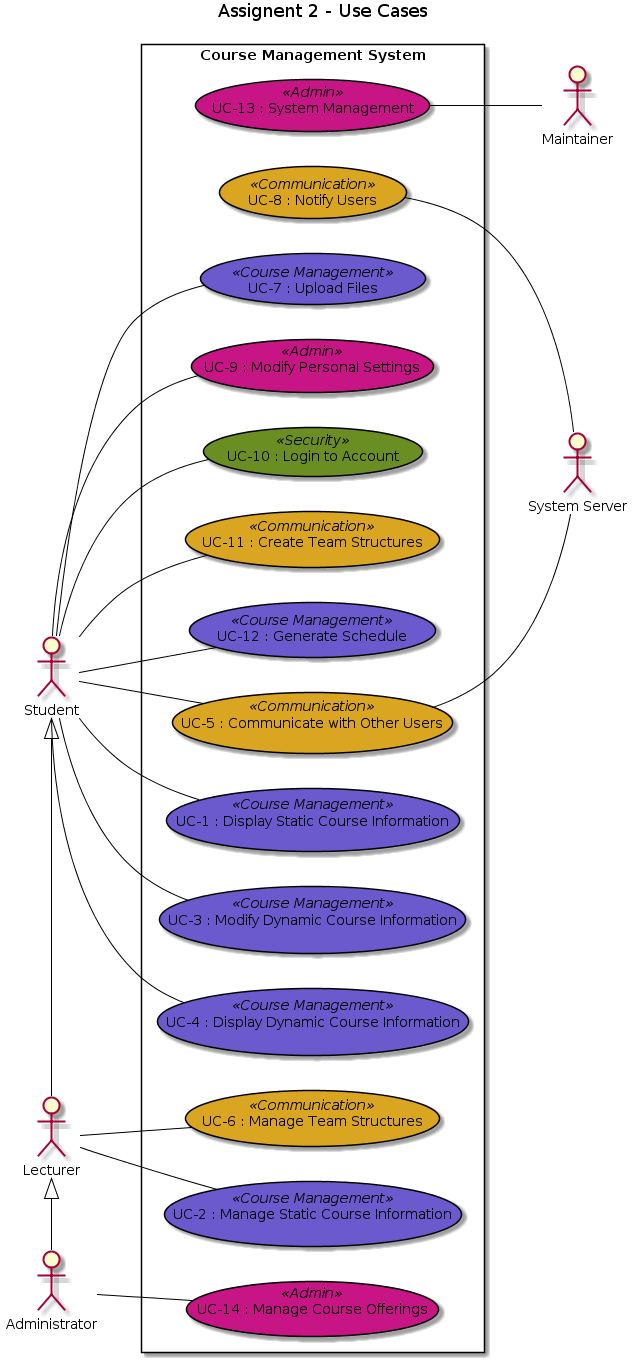
Gabriel Gelgor 100614547

Ethan Wallace 100632495

**Table of Contents:**

1.1: System Requirements

**1.1.1: Use Case Model**



**1.1.2: Quality Attribute Scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Quality Attribute** | **Scenario** | **Associated Use Case** |
| QA-1 | Privacy | Privacy of all stakeholders is respected by having a permissions and access system via a university login. That way students can only see relevant information to their own results and those who do not need to use advanced functions (like students and lecturers) cannot affect the functioning of the system | UC-9 |
| QA-2 | Availability | The system will strive to have high availability. The maintainers have access to many system management functions to allow this, including system images and backup/restore procedures. | UC-13 |
| QA-3 | User Friendliness | All system components should be as user friendly as possible. Regardless of the permissions of the user, all information should be within 3 clicks for example. | ALL |
| QA-4 | Accessibility | All system components should be accessible to students/staff with disabilities. Screen-reader support for the blind should be integrated into every function. | ALL |
| QA-5 | Security | A login system will ensure security. Only the person associated with the account should have access to that account. | UC-10 |
| QA-6 | Interoperability | Administrators will have access to a wide variety of special functions and coroutines to integrate the CMS with other technological assets belonging to the university. | UC-14 |

**1.1.3: Constraints**

|  |  |
| --- | --- |
| **ID** | **Constraint** |
| CON-1 | To prevent unexpected downtime, this system requires the use of multiple seperate servers in different locations ready to take on the load should any other server fail. |
| CON-2 | The system must be accessible through a web browser. |
| CON-3 | The system requires two separate server types, one for messaging and one for data storage. Sharing the functionality on a single server runs the risk of greatly slowing down the entire system, and in the worst case can cause the crashing of one service to cascade to the other. |
| CON-4 | The network connection of the servers to the system must be reliable. |
| CON-5 | All website programming must be screen-reader friendly. |
| CON-6 | The system must be built in a modular design. |

**1.1.4: Architectural Concerns**

|  |  |
| --- | --- |
| **ID** | **Concern** |
| CRN-1 | Generating a prototype software structure to facilitate a robust and reliable piece of software. |
| CRN-2 | Ensure that conflicting requirements can be solved in a sensible and acceptable way |
| CRN-3 | Properly distribute work among other engineers |

1.2: ADD Input Review

|  |  |
| --- | --- |
| **Category** | **Details** |
| Design objectives | This is a Greenfield system in a mature domain. Very little new technical ground will be tread over development if at all. The purpose is to design a robust scaffolding that is able to support the final product. |
| Primary Functional Requirements | From the use cases in section 1.1.1, the primary functionality of this system has been determined to be:   * UC-1: As it provides core functionality * UC-2: As it support core functionality * UC-3: As it provides core functionality * UC-4: As it provides core functionality |
| Quality Attribute Scenarios | |  |  |  | | --- | --- | --- | | **Scenario ID** | **Customer Importance** | **Difficulty to make** | | QA-1 | High | Low | | QA-2 | High | Medium | | QA-3 | Medium | Medium | | QA-4 | Medium | High | | QA-5 | High | High | | QA-6 | Medium | Low |   From This list, QA-1, QA-2, and QA-5 have been selected as QA drivers. |
| Constraints | All constraints have been selected as drivers |
| Concerns | All concerns have been selected as drivers |

1.2: ADD Iteration 1 - Generate Software System Structure

**1.2.1 Establish Iteration Goal by Selecting Drivers**

The goal of this iteration is to confront CRN-1, generating a prototype software structure to facilitate a robust and reliable piece of software. Addressing this concern is necessary to being able to proceed with the second iteration. The drivers are as follows:

* QA-1: Privacy
* QA-2: Availability
* QA-5: Security
* CON-1: To prevent unexpected downtime, this system requires the use of multiple seperate servers in different locations ready to take on the load should any other server fail.
* CON-2: The system must be accessible through a web browser.
* CON-3: The system requires two separate server types, one for messaging and one for data storage.
* CON-6: The system must be built in a modular design.
* CRN-1: Generating a prototype software structure to facilitate a robust and reliable piece of software.

**1.2.2 Choose One or More Elements of the System to Refine**

The element to be refined in this iteration is the entire system. In this scenario, refinement is achieved via component decomposition.

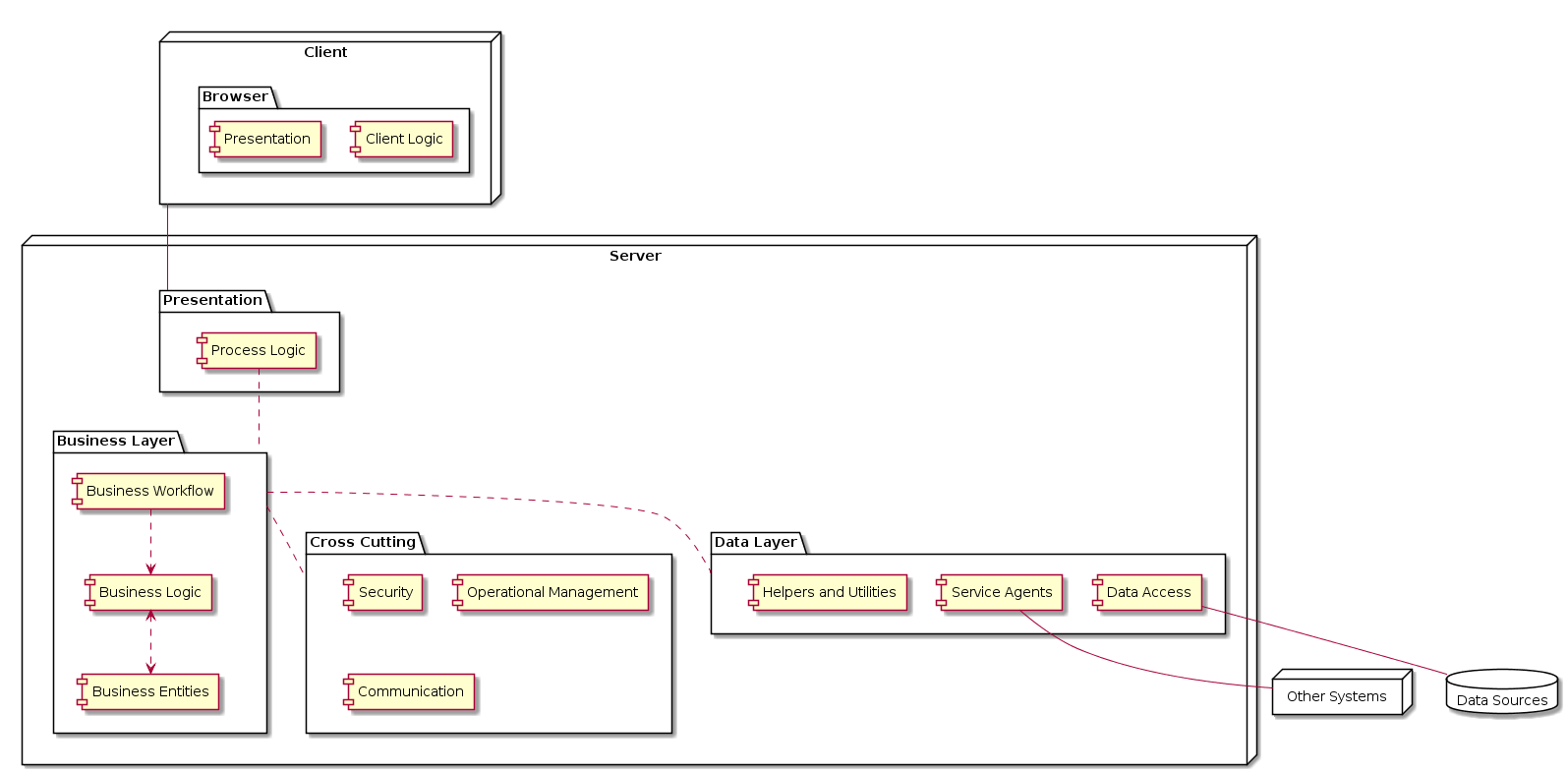
**1.2.3 Choose One or More Design Concepts That Satisfy the Selected Drivers**

|  |  |
| --- | --- |
| **Design Decisions and Location** | **Rationale** |
| The client part of the system will be structured using the Rich Internet Application reference architecture. | As a rich internet application in the browser, CON-2 is fulfilled using this architecture. As every student has a browser, QA-2 is achieved using this system (as is CON-1). The client application would allow for sign-ins (verified by the backend server) thus achieving QA-5. |
| |  |  | | --- | --- | | **Discarded Alternatives** | | | ***Alternative*** | ***Reason for Discarding*** | | Rich Client Application | Although architecturally similar, this alternative flies in the face of CON-2: the system must be accessible through a web browser. Additionally, it was deemed that storing information locally is unnecessary, and opens up more security and performance problems than it fixes. | |
| The server part of the system will be structured using the Web Application reference architecture. | This architecture would see a separate application server that takes requests and provides outputs (mediated by load-balancers - see below). We can have multiple application servers, we could create separate messaging and data storage servers (CON-3). These servers would provide authentication services for the client (QA-5). The modular approach to web application architectures achieves CON-6. |
| |  |  | | --- | --- | | **Discarded Alternatives** | | | ***Alternative*** | ***Reason for Discarding*** | | Mobile Application Reference Architecture | While able to achieve a very similar goal, it was decided that designing this architecture around mobile access is unnecessary. The same product can be achieved with greater sophistication without the added requirement to worry about mobile restrictions. Additionally, there was no specific requirement to make this application available on mobile platforms. | |
| The deployment will be structured behind a load-balanced cluster deployment pattern. | An important constraint is CON-1, which requires multiple servers to minimize downtime. CON-3 and CON-6 can also be fulfilled using this system. Because of the good performance associated with this deployment pattern, QA-1 is fulfilled. |
| |  |  | | --- | --- | | **Discarded Alternatives** | | | ***Alternative*** | ***Reason for Discarding*** | | N-tier deployment | The N-tier deployment patterns were considered, and ultimately rejected. They did not provide the appropriate flexibility and performance that is inherit in load-balanced clusters. In an N-tier deployment, we could run into issues of overloading the server with requests leading to downtime (in opposition to CON-1 and QA-5). | |

**1.2.4 Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces**

|  |  |
| --- | --- |
| **Design Decisions and Location** | **Rationale** |
| Remove plug-in execution container in the rich internet application | The rich application will be constructed using native browser protocols (i.e., though normal JavaScript). As we will use JavaScript (more specifically, ReactJS) to engineer the front-facing client application, we do not need a plug-in execution container (e.g., an applet) to provide the required functionality. As JavaScript is contained by default in every browser, this ensures ease of use and portability. |
| The introduction of a ServerManagement component to act as an enhanced server load balancer | In order to ensure the correct information is received by the correct server type, the server load balancer will have additional business logic attached to it. This logic will allow for the dynamic routing of requests to the correct server (whether it be a messaging or data server). |

**1.2.5 Sketch Views and Record Design Decisions**

****

|  |  |
| --- | --- |
| **Element** | **Responsibility** |
| Presentation | Module that controls the user view and what controllers they have access to. |
| Client Logic | Component that handles all use cases that does not require the accessing of a server. |
| Process Logic | Component to ensure that all information coming from the client side is directed to the appropriate destination for handling. |
| Business Logic | This module is responsible for the checking of business rules that require the accessing of a server, whether it be for messaging or data storage/retrieval. |
| Business entities | This module represents the domain model. |
| Business workflow | The set of connectors used to generate a fully operational business process. |
| Service Agents | The module that handles interoperability with external systems. |
| Data Access | Component to connect with API endpoints on the database. |
| Helpers and Utilities | Component that handles connections with a database (connection strings, query sending and fetching, etc.). |
| Security | Ensures that all permission levels are enforced, maintains encryption methods, watches for suspicious access. |
| Communication | Communication between all other components residing server-side. |
| Operational Management | Exception management and logging. |
| Browser Layer | Contains all components that are directly interfaced with by the client and their browser. |
| Presentation Layer | Contains components that are required to parse/handle incoming user input. |
| Business Layer | Contains all logical components central to the functional requirements. |
| Data Layer | Contains all components required to properly interact with data and their respective databases. |
| Cross Cutting Layer | Contains all components that have an impact on multiple layers of the server at once. |

**1.2.6 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose**

|  |  |  |  |
| --- | --- | --- | --- |
| **Not Addressed** | **Partially Addressed** | **Completely Addressed** | **Design Decision made during the iteration** |
|  |  | CON-1 | Selected a reference architecture that will support fulfilling this functionality. |
|  |  | CON-2 | Selected a reference architecture that will support fulfilling this functionality. |
|  |  | CON-3 | Selected a reference architecture that will support fulfilling this functionality. |
|  | CON-6 |  | While using a load-balanced cluster deployment pattern lays the groundwork for this functionality, actual modularity on a component level has yet to be defined. |
|  |  | CRN-1 | A full prototype architecture for this software system has been developed. |
|  | QA-1 |  | While the groundwork for this has been created due to our choice of architecture, we did not address this specifically in this iteration. |
|  |  | QA-2 | Selected a reference architecture that will support fulfilling this functionality. |
|  | QA-3 |  | While the groundwork for this has been created due to our choice of architecture, we did not address this specifically in this iteration. |
| UC-1 |  |  | No relevant decisions made. |
| UC-2 |  |  | No relevant decisions made. |
| UC-3 |  |  | No relevant decisions made. |
| UC-4 |  |  | No relevant decisions made. |