Architecture Final Project

CMS ADD Iterations

Gabriel Gelgor 100614547
Ethan Wallace 100632495

Table of Contents:

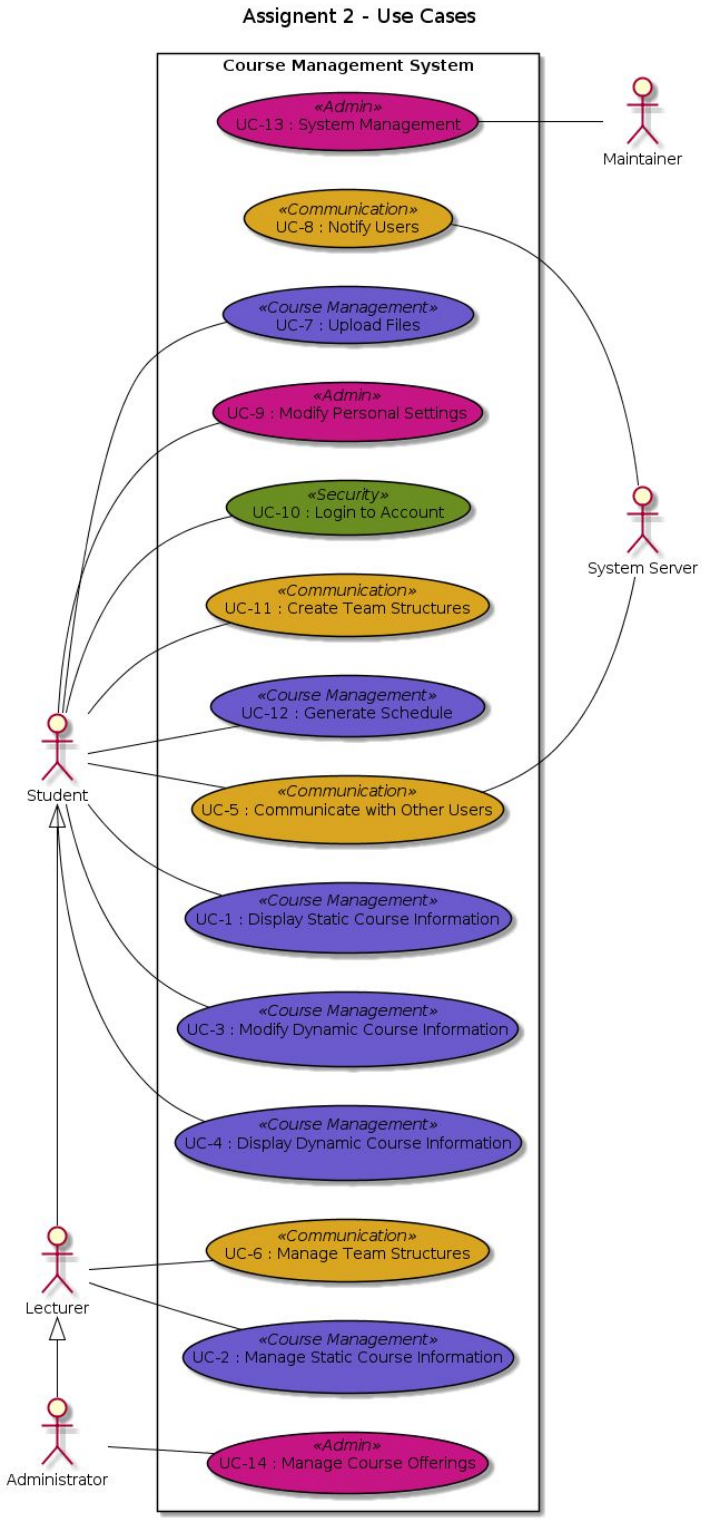
Contributions:

Ethan and Gabe contributed to the following project equally.

Ethan	50%
Gabe	50%

1.1: System Requirements

1.1.1: Use Case Model



Use Case Diagram

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

<table>
<tr><th>Category</th><th>Details</th></tr>
<tr><td>Design objectives</td><td>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.</td></tr>
<tr><td>Primary Functional Requirements</td><td>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</td></tr>
<tr><td>Quality Attribute Scenarios</td><td><table><tr><th>Scenario ID</th><th>Customer Importance</th><th>Difficulty to make</th></tr><tr><td>QA-1</td><td>High</td><td>Low</td></tr><tr><td>QA-2</td><td>High</td><td>Medium</td></tr><tr><td>QA-3</td><td>Medium</td><td>Medium</td></tr><tr><td>QA-4</td><td>Medium</td><td>High</td></tr><tr><td>QA-5</td><td>High</td><td>High</td></tr><tr><td>QA-6</td><td>Medium</td><td>Low</td></tr></table>From This list, QA-1, QA-2, and QA-5 have been selected as QA drivers.</td></tr>
<tr><td>Constraints</td><td>All constraints have been selected as drivers</td></tr>
<tr><td>Concerns</td><td>All concerns have been selected as drivers</td></tr>
</table>

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

<table>
<tr><th>Design Decisions and Location</th><th>Rationale</th></tr>
<tr><td rowspan="2">The client part of the system will be structured using the Rich Internet Application reference architecture.</td><td>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.</td></tr>
<tr><td>
<table>
<tr><th colspan="2">Discarded Alternatives</th></tr>
<tr><th>Alternative</th><th>Reason for Discarding</th></tr>
<tr><td>Rich Client Application</td><td>Although architecturally similar, this alternative flies in the face of CON-2:</td></tr>
</table>
</td></tr>
</table>

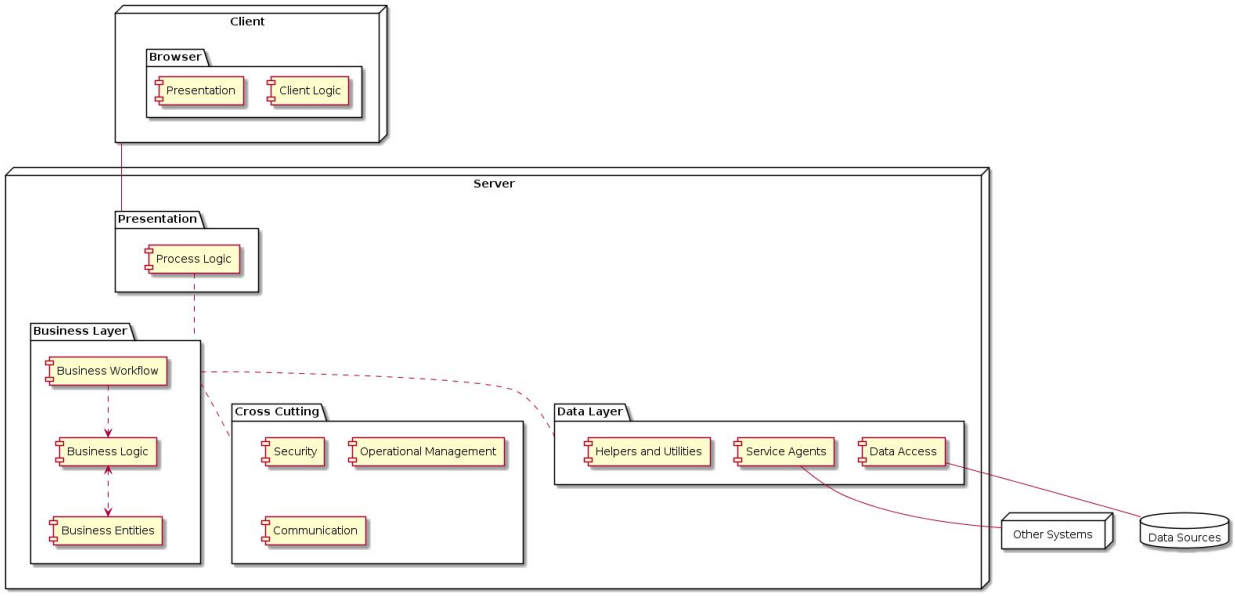
<table>
<tr><td></td><td>
<table>
<tr><td></td><td>the system must be accessible through a web browser. Additionally, it was deemed that storing information locally is unnecessary, and opens up more security and performance problems than it fixes.</td></tr>
</table>
</td></tr>
<tr><td rowspan="2">The server part of the system will be structured using the Web Application reference architecture.</td><td>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.</td></tr>
<tr><td>
<table>
<tr><td colspan="2">Discarded Alternatives</td></tr>
<tr><td>Alternative</td><td>Reason for Discarding</td></tr>
<tr><td>Mobile Application Reference Architecture</td><td>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.</td></tr>
</table>
</td></tr>
<tr><td rowspan="2">The deployment will be structured behind a load-balanced cluster deployment pattern.</td><td>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.</td></tr>
<tr><td>
<table>
<tr><td colspan="2">Discarded Alternatives</td></tr>
<tr><td>Alternative</td><td>Reason for Discarding</td></tr>
<tr><td>N-tier deployment</td><td>The N-tier deployment patterns were considered, and ultimately rejected.</td></tr>
</table>
</td></tr>
</table>

	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



Module View of the reference architectures (Key: UML)

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-5		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.

2.1: ADD Iteration 2 - Identifying Structures to Support Primary Functionality

---

2.1.1 Establish Iteration Goal by Selecting Drivers

The goal of this iteration is to confront CON-6, QA-1 and QA-5: modularity, privacy, and security. These three requirements heavily influence the underlying construction of a system. While not heavily affect overall architecture, they do heavily affect internal component structure. The drivers are as follows:

- QA-1: Privacy
- QA-5: Security
- CON-6: The system must be modular.

2.1.2 Choose One or More Elements of the System to Refine

In this iteration we will be refining the server architecture (Web Application reference architecture). Within this architecture, the modules most related to our identified drivers are inside the cross-cutting system; specifically the security and communication components.

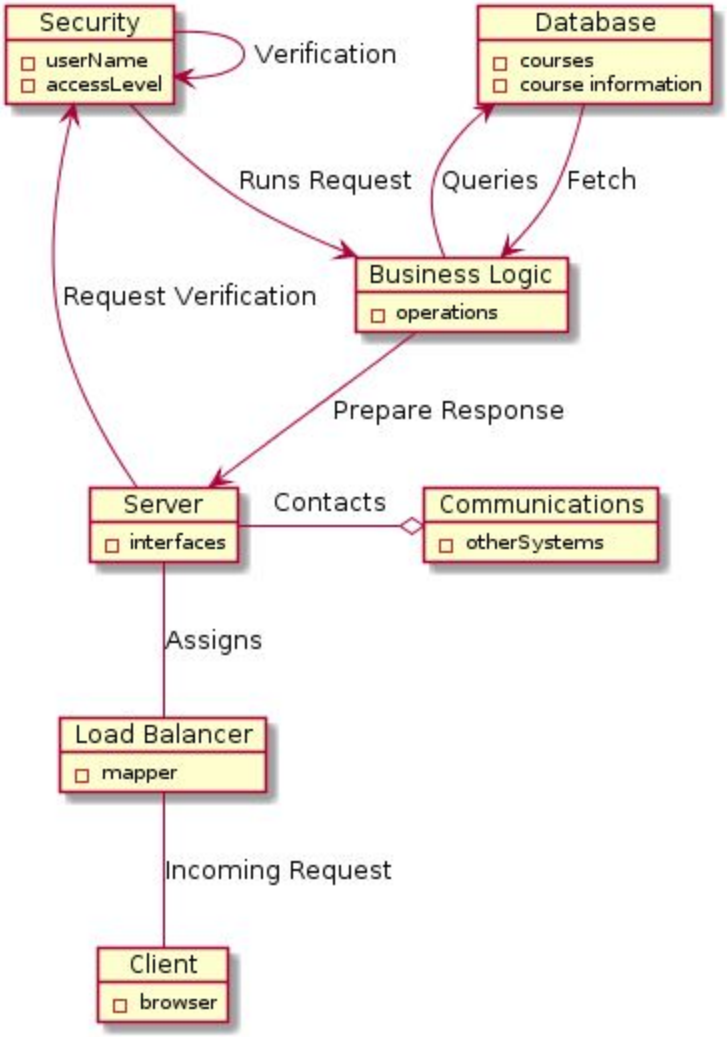
2.1.3 Choose One or More Design Concepts That Satisfy the Selected Drivers

Design Decisions and Location	**Rationale**
Use an explicit interface for the server architecture.	With an explicit interface, we can ensure that we have as much modularity in the back end server system as possible. We can easily create new, or improve old, functionality and point common standard interfaces to whatever we need. This will allow easy upgrades of the system.

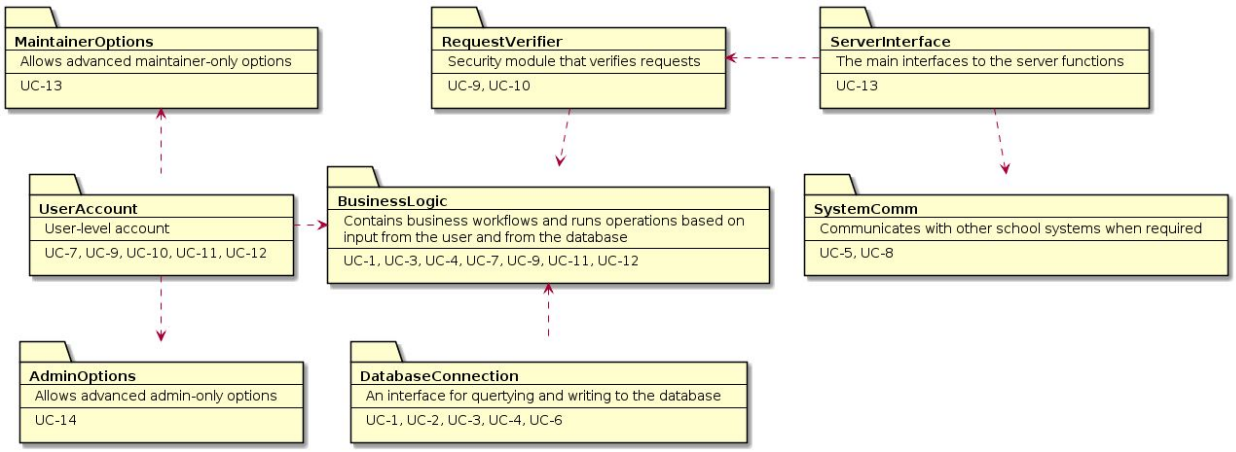
2.1.4 Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

Design Decisions and Location	**Rationale**
Abstract all functionality to an explicit interface	All server modules will have an explicit interface on the server side. This will increase modularity and allow easy upgrades or reconfiguration of the server's internal structure. This would support QA-2, CON-3, and CON-6.
Make generic functions that can be pointed to by any interface for increased code reusability	By ensuring a consistent interface, we can ensure that the application will always work correctly regardless of the browser frontend. As well, we can ensure that existing functionality can be replicated on different server types by just pointing the public interface to the correct function. This supports CON-4 and CON-6

2.1.5 Sketch Views and Record Design Decisions



Initial domain model (Key: UML)



Domain objects associated with the use case model (Key: UML)

2.1.6 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
		QA-1	Selected a reference architecture that will support fulfilling this functionality.
	QA-5		Selected a reference architecture that will support further development of this quality attribute.
		CON-6	Selected a reference architecture that will support fulfilling this design requirement.

3.1: ADD Iteration 3 - Assessing Quality Attribute Scenario (QA-5)

3.1.1 Establish Iteration Goal by Selecting Drivers

The goal of this iteration is to determine how well this system handles an attack where a user attempts to brute force their way into an account that is not theirs. From studying this scenario, a more robust system of handling such situations in the future will be developed. The drivers are as follows:

- QA-5: Security

3.1.2 Choose One or More Elements of the System to Refine

In this iteration we will be refining the the cross-cutting layer of the system architecture. Most attention within this later will be directed towards the security component of the layer, with an internal domain model being developed for it.

3.1.3 Choose One or More Design Concepts That Satisfy the Selected Drivers

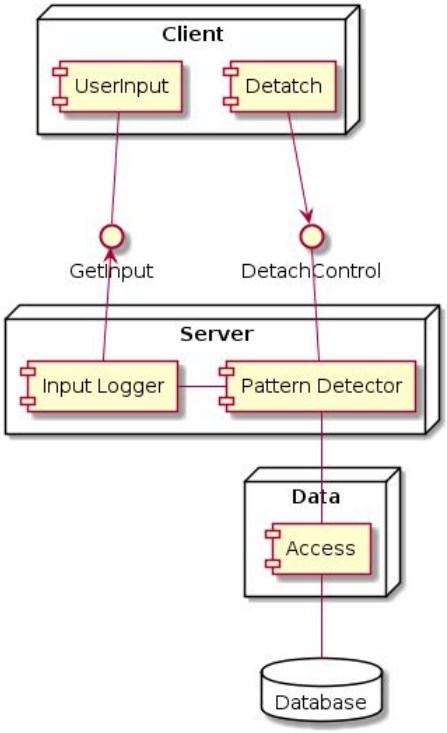
Design Decisions and Location	Rationale
Implement a domain model within the security component to	By enabling our security module to compare current activity with known malicious activity patterns, we can shut down any attack as early as possible. If such a thing occurs, the machines matching that

compare network traffic or service request patterns to a set of signatures or known patterns of malicious behaviour. This information will be accessed from a database.	pattern can be immediately shut out of system access points
Implement a ‘detachable UI’ that separates itself from the network in the event that a brute-force attack occurs.	This security method wastes the attacker’s time by allowing their scripts to continue, but removes all stress on the system network.
Give security access to data layer so that it can correspond suspicious activity with users.	By linking suspicious activity with users/IP addresses, we can create a priority list for our security system. Higher risk users are identified immediately, and more resources can be put towards finer pattern recognition.

3.1.4 Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

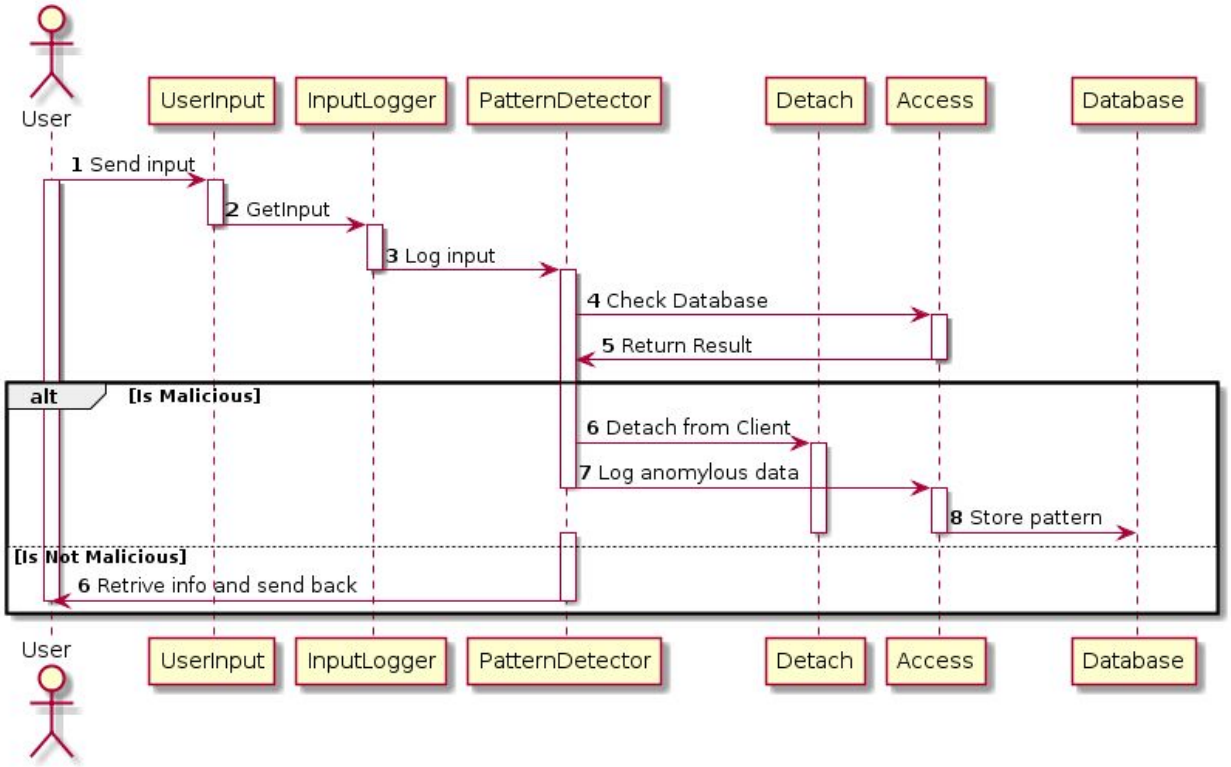
Design Decisions and Location	**Rationale**
Internally split security component into input logger and pattern detector.	Creates the separation of concerns required to facilitate the original design decision.
Add a ‘detach’ component split among both the client and the business layer.	The cross-cutting layer informs the business layer that suspicious activity is afoot on the client’s end, which then sends a signal to the client’s browser requesting it to detach from the network.
Interface created between security component and data layer.	Enables the storage/retrieval of patterns from a database, and the logging of suspicious users.

3.1.5 Sketch Views and Record Design Decisions



Security Deployment Diagram (Key: UML)

Element	**Responsibility**
Access	Communicates queries to server and fetches the result of those queries.
Input Logger	Formats and prepares user input data in a record-friendly format for storage in case of recognized user pattern.
Pattern Detector	Compares user with known suspicious users, and their input with a database of known attack patterns. Triggers user interface detachment module upon confirmation of attack.
Usernput	Controller that is imbedded into the user's view.
Detatch	Detaches the user interface from the network; malicious scripts will continue to run, but no extra strain will be placed on the architecture. This is a tactic to waste the time of attackers.



Security feature sequence diagram (Key: UML)

3.1.6 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
		QA-5	Developed a robust mechanism to protect the system in case of a detected attack.
	CON-4		Network reliability increased as strain from attacks will no longer affect the system.
		CON-2	No relevant decisions made.
		CON-3	No relevant decisions made.
	CON-6		No relevant decisions made.
		CRN-1	No relevant decisions made.
	QA-1		No relevant decisions made.
		QA-2	Availability increased, as less time will be spent

			processing brute force attacks thanks to detachment functionality.
UC-1			No relevant decisions made.
UC-2			No relevant decisions made.
UC-3			No relevant decisions made.