Iteration 2: Identifying Structures to Support Primary Functionality

The focus of this section is to present the results of the activities that are performed from each step of the ADD process in the second iteration. This iteration will dive deeper into more detailed decisions that will promote better implementation which will aid the development team. As we can not predict everything, we need to be more disciplined on which decisions we make and attack them from the most significant to least significant.

Step 2: Establish Iteration Goal by Selecting Drivers

The goal of this iteration is to address the general architectural concern of identifying structures to support primary functionality. Identifying these elements is important for understanding how functionality is supported and addressing CRN-3.

Besides CRN-3, the architect considers the system's primary use cases:

- UC-1
- UC-2
- UC-6
- UC-9

Step 3: Choose One or More Elements of the System to Refine

The elements that will be refined in this iteration are the modules located in the different layers defined by the three-tier reference architectures from the previous iteration. In general, the support of functionality in this system requires the collaboration of components associated with modules that are located in the different layers.

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

In this iteration, the book Pattern-Oriented Software Architecture is used to select several architectural design patterns. The following table summarizes the design decisions.

Design Decisions and Location	**Rationale**
Create a **Domain Model** for the application	Before starting a functional decomposition, it is necessary to create an initial domain model for the system, identifying major entities in the domain, along with their relationships.
Identify **Domain Objects** that map to the functional requirements	Each distinct functional element of the application needs to be encapsulated in a self-contained building block - a domain object.
Decompose **Domain Objects** into general and specialized **Components**	Domain objects represent complete sets of functionality, but this functionality is supported by finer-grained elements located within the layers. The "components" in this pattern are what we have referred to as modules.
Using JSON programming language, handlebars, and	By using JSON which is a lightweight data-interchange format where data is able to be transferred and used through a web application.

sockets	Alongside that JSON supports many frameworks and libraries which make scalability. By using handlebars which are functions that support GUI it makes it easier to display information from the server. Sockets.io is a library that is a real-time web application that allows for bidirectional communication between clients and servers.

Step 5: 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**
Create only an initial domain model	The entities that participate in the primary use cases need to be identified and modeled but only an initial domain model is created, to accelerate this phase of design.
Map the system use cases to domain objects	An initial identification of domain objects can be made by analyzing the system's use cases. To address CRN-3, domain objects are identified for all of the use cases.
Decompose the domain objects across the layers to identify layer-specific modules with an explicit interface	This technique ensures that modules that support all of the functionalities are identified. The architect will perform this task just for the primary use cases. The rest of the team members will work on other modules, thus dividing up the work. A new concern will be created in which is identified below: CRN-4: Some of the modules shall be tested to determine their functionality.

Step 6: Sketch Views and Record Design Decisions

As a result of the decisions made in step 5, the design decisions can be portrayed into several diagrams.

- Figure showcases an initial domain model for the system
- Figure shows the domain objects that are instantiated for the use case model in Section

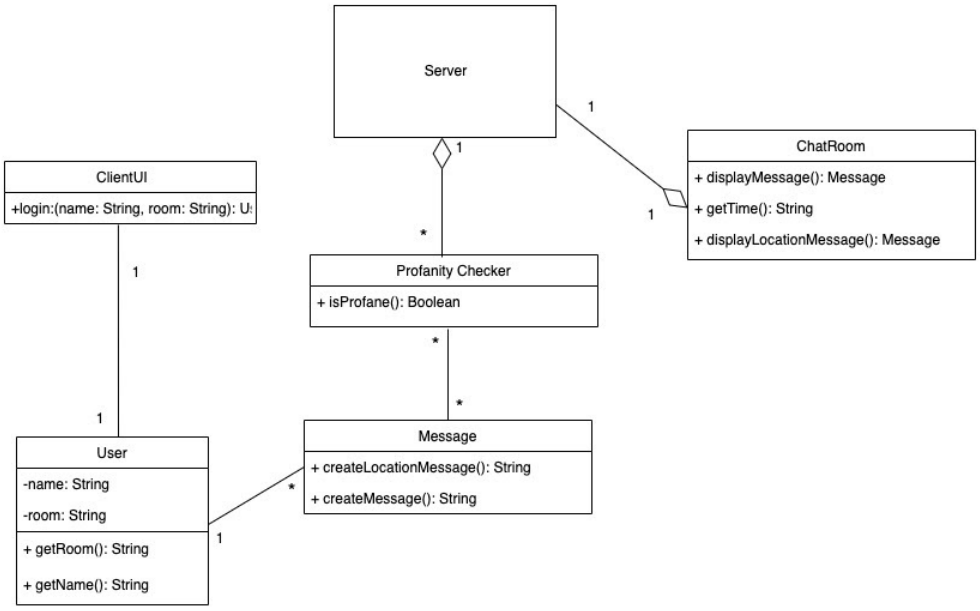

Figure 3.1 *Initial Domain Model*

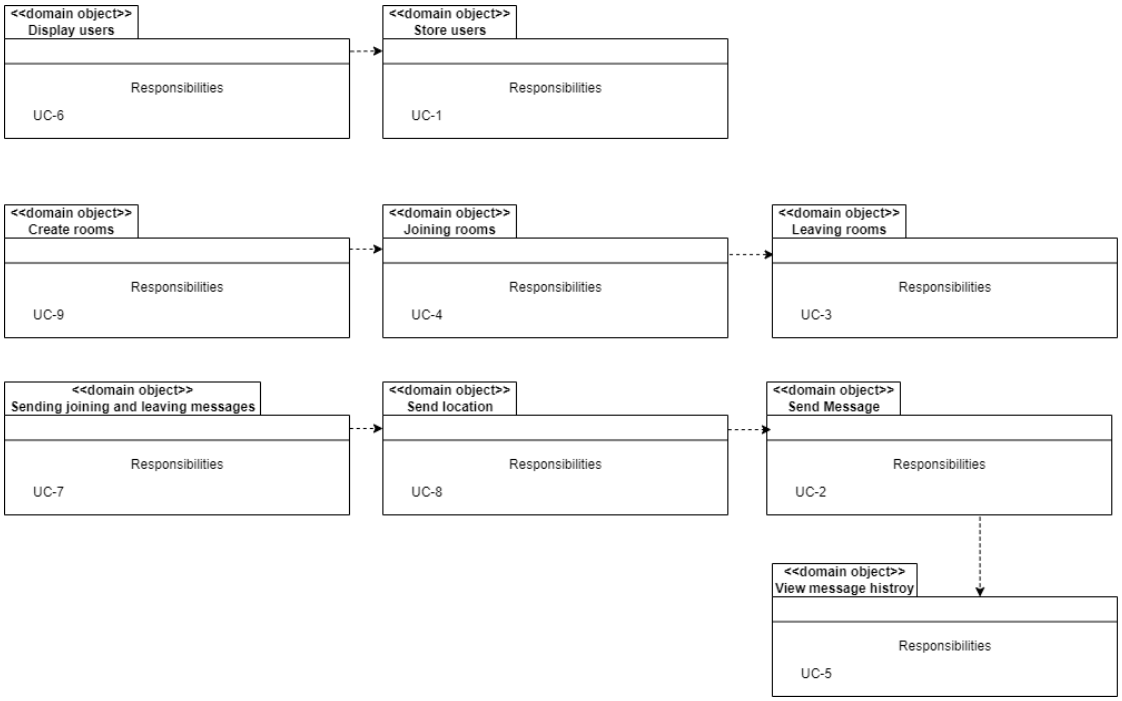

Figure 3.2 *Domain Objects associated with the use case model*

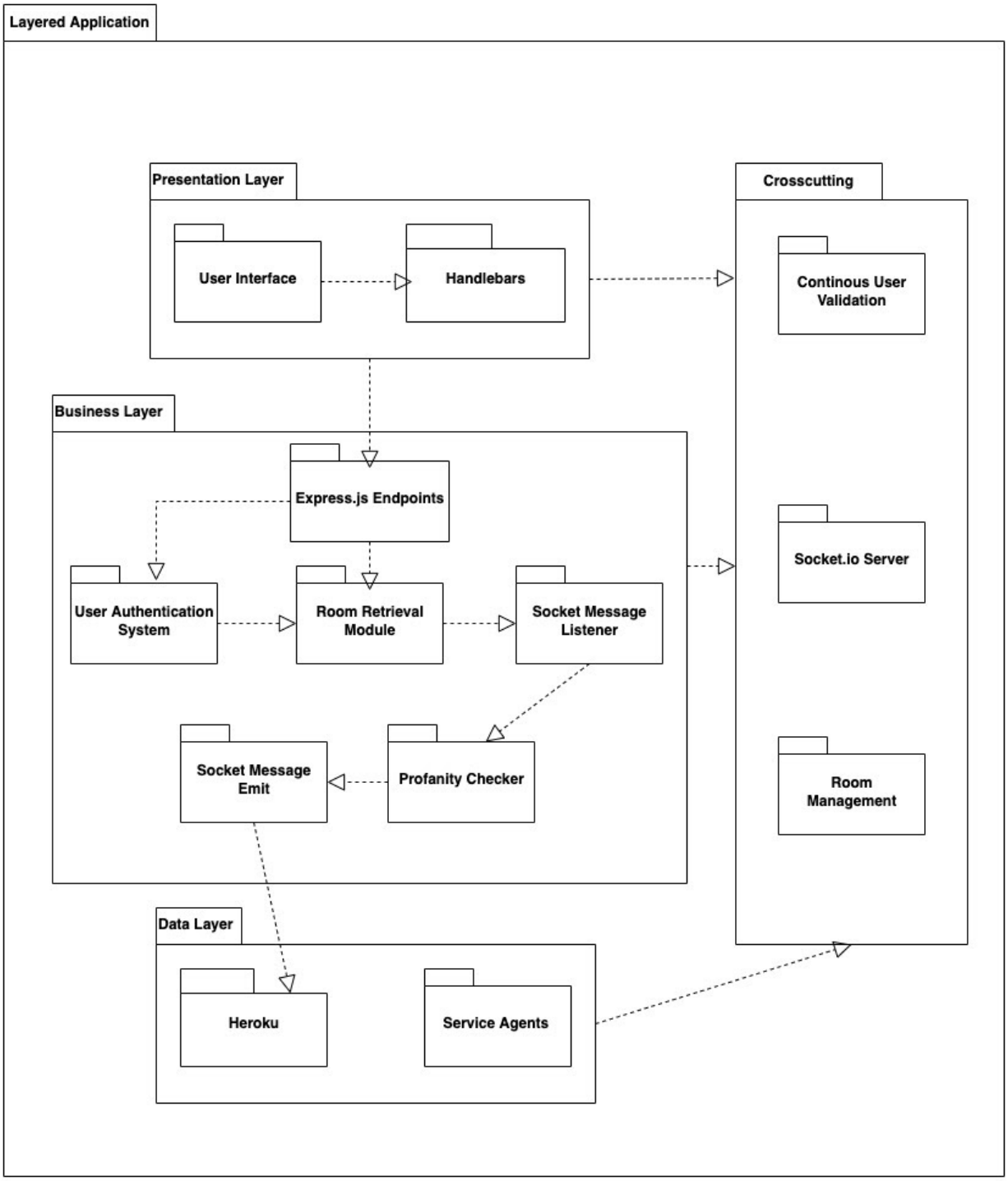

Figure 3.3 *Modules that support the primary use cases*

Element	Responsibility
Express.js Endpoints	Responsible for endpoints, allowing to build a REST based project, it allows to

	define GET and POST methods. Also for the program, to send messages between requests and to send responses back to the client.
User authentication Systems	Responsible for ensuring the user was created and registered within the database correctly.
Room Retrieval Module	Responsible for getting the room Id based on the user input
Socket Message Listener	Responsible for listening to the message front he emitted a message module.
Profanity Checker	Responsible for checking user input to the chat to see if there is a profan message.
Socket Message Emit	Responsible for sending/emitting for which the listener will take in.
Continuous User Validation	Responsible for checking if a user is still registered within the database.
Socket.io Server	Responsible for encapsulating the emitter and listener.

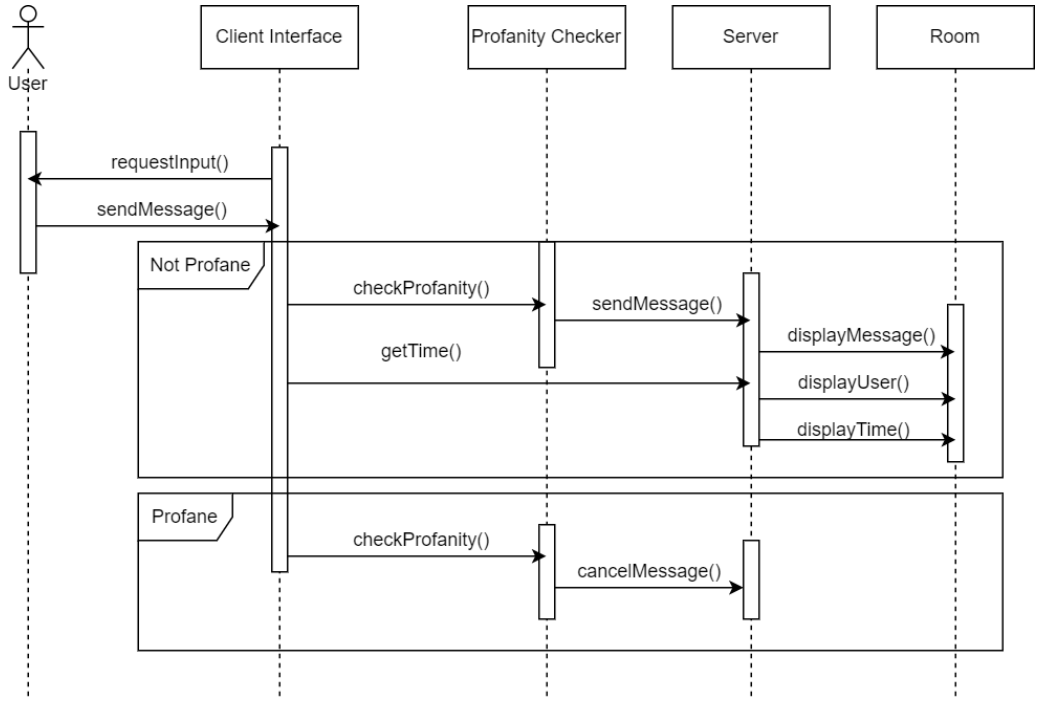

Figure 3.4 *Sequence diagram for UC-2*

Element	**Responsibility**
Client Interface	Display GUI for user to input/view text, view users and send location.

Profanity Checker	Checks if messages do not have offensive language before displaying it to the users.
Server	Sends the messages to the rooms and cancels messages which are offensive.
Room	Displays messages, users and the time messages.

Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

The decisions made in this iteration provided an initial understanding of how the system shall function. The modules associated with the primary use cases were identified by the architect, and the module associated with the rest of functionality were identified by the rest of the team

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During this Iteration
		UC-1	Modules across the layers and preliminary interface to support this use case have been identified.
		UC-2	Modules across the layers and preliminary interface to support this use case have been identified.
		UC-6	Modules across the layers and preliminary interface to support this use case have been identified.
		UC-9	Modules across the layers and preliminary interface to support this use case have been identified.
	QA-1		The elements that support the associated use case (UC-1) have been identified.
	QA-3		No relevant decisions made.
	QA-4		The elements that support the associated use cases (All UC-1) have been identified.
	QA-5		The elements that support the associated use case (UC-6) have been identified.
		CON-3	Modules responsible for authenticating have been identified.

CON-4		No relevant decision made.
CON-5		No relevant decision made.
	CRN-2	Modules incorporating handlebars have been identified.
	CRN-3	Modules associated with all of the use cases have been identified and a work assignment matrix has been created.
CRN-4		The architectural concern of unit-testing modules, which was introduced in this new interaction.