Project Deliverable 3

CRN: 43509

Group 7

Kalapan Kannathasan - 100759041

Yash Patel - 100746810

Sujeev Uthayakumar - 100744194

Zirak Mughal - 100749132

Date: December 6, 2021

Table of Contents

1.1 System Requirements

Use Case Model:

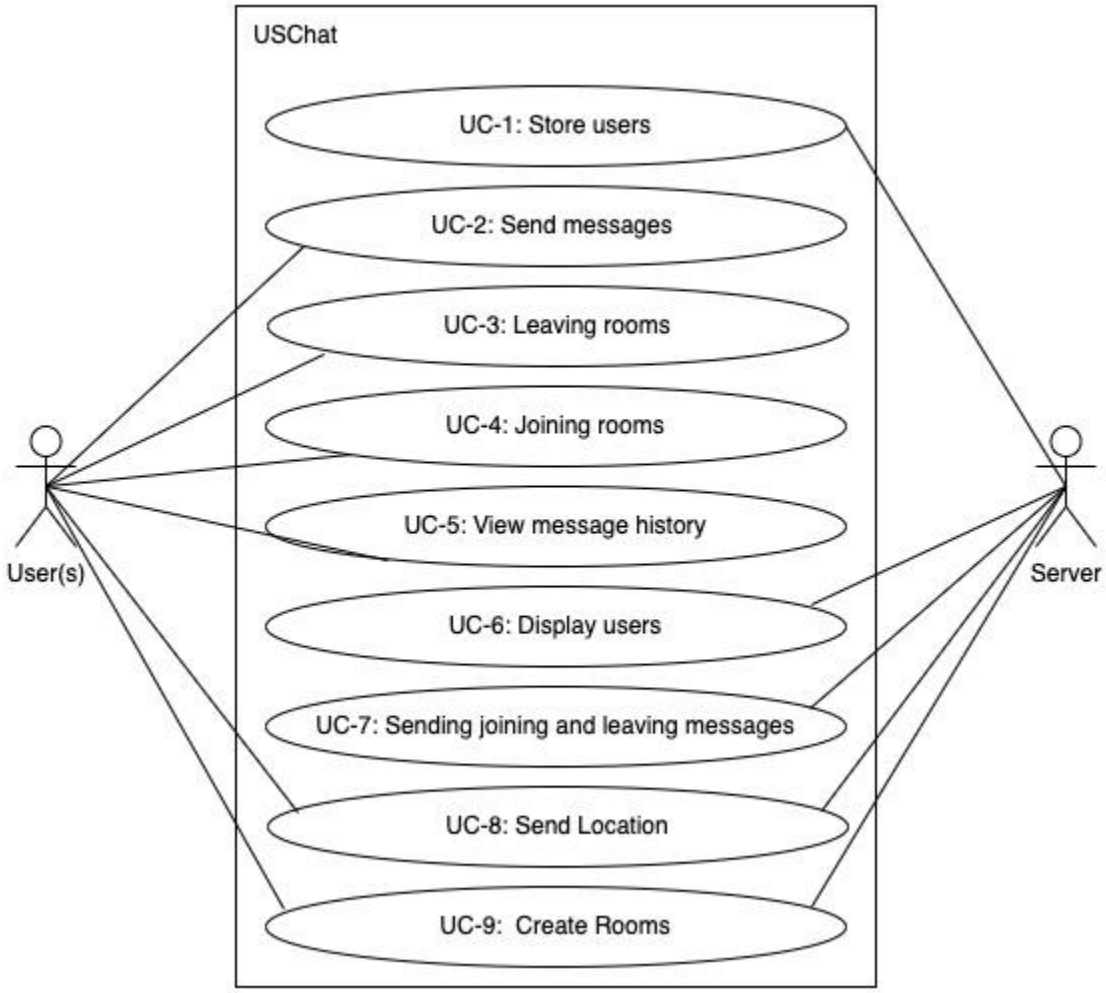


Figure 1.1 *Use case model for the FCAPS system*

Quality Attributes Scenarios:

ID	Quality Attribute	Scenario	Associated
QA-1	Scalability	Create a user and send it to be stored within the server, with an ever-expanding user base.	UC-1, UC-9
QA-2	Usability	Users are able to send messages and the timestamps are recorded.	UC-2
QA-3	Security	Send location coordinates, however, the coordinate will not be exact as it will be moved to protect privacy.	UC-8

QA-4	Performance	The ability to perform all tasks without comprising real-time functionality. Where messages are seen within a second of being sent.	All
QA-5	Availability	Users are able to see other users as well as the admin.	UC-6

Constraints:

ID	Constraints
CON-1	A minimum of 10 users must be supported by the system.
CON-2	Messages should be sent and displayed in less than 1 second on either end.
CON-3	Users should be authenticated before joining the room, and after through continuous checking.
CON-4	Network connection between user and server must have low bandwidth and be reliable.
CON-5	The system must be accessed through a web browser such as Chrome, Firefox Safari, etc.
CON-6	Large amount of messages will have to be stored in a single session.
CON-7	An existing relation database server must be used. This server must hold the user and room information that will be used to fetch information.

Architectural Concerns:

ID	Concern
CRN-1	Establishing an overall initial system structure.
CRN-2	Leverage the team's knowledge about JSON and dependencies, including handlebars, and socket.io.
CRN-3	Allocate work to members of the development team.

1.1 The Design Process

This is the first part of the design process, we must translate the requirements and quality attributes that were previously gathered into design decisions. These design decisions will be derived from figure 1.2, which is a list of our quality attributes. This is the beginning of making design decisions that will allow for the development of the entire design of USChat.

1.2 ADD Step 1: Review Inputs

The first step of the ADD method involves reviewing the inputs and identifying which requirements will be considered as drivers. The inputs for USChat are summarized in the table below.

Category	**Details**
Design Purpose	The purpose is to build a sufficiently detailed design to support the construction of the USChat program.
Primary functional requirements	UC-1: Because it directly supports the core of the business UC-2: Because it directly supports the core of the business UC-6: Because it directly supports the core of the business UC-9: Because it directly supports the core of the business
Constraints	All the constraints discussed in system requirements will be included as drivers
Architectural Concerns	All the architectural concerns discussed in system requirements will be included as drivers

ID	**Importance to the customer**	**Difficulty of Implementation According to the Architect**
QA-1	Low	High
QA-2	High	Low
QA-3	High	Medium
QA-4	Medium	Medium
QA-5	Medium	High

2.1 Iteration 1: Establishing an Overall System Structure

The focus of this section is to present the results of the activities that are performed in each of the steps in iteration 1 of the ADD design process. Step 2 till the end of step 7 will be focusing on iteration 1, where we will refine our requirements and quality attributes.

2.2 Step 2: Establish Iteration Goal by Selecting Drivers

This is the first iteration of the design of a USChat system, so the iteration goal is to establish the initial design of the system, where we will establish the overall system structure. This iteration will be driven by general architectural concerns, and there will be a focus on all the drivers that may further influence the general structure of the system. In particular, we will focus on items within:

- QA-1: Scalability
- QA-3: Security
- QA-4: Performance
- QA-5: Availability
- CON-3: Users should be authenticated before joining the room
- CON-4: Network connection between user and server must have low bandwidth and be reliable
- CON-5: The system must be accessed through a web browser such as Chrome, Firefox Safari, etc.

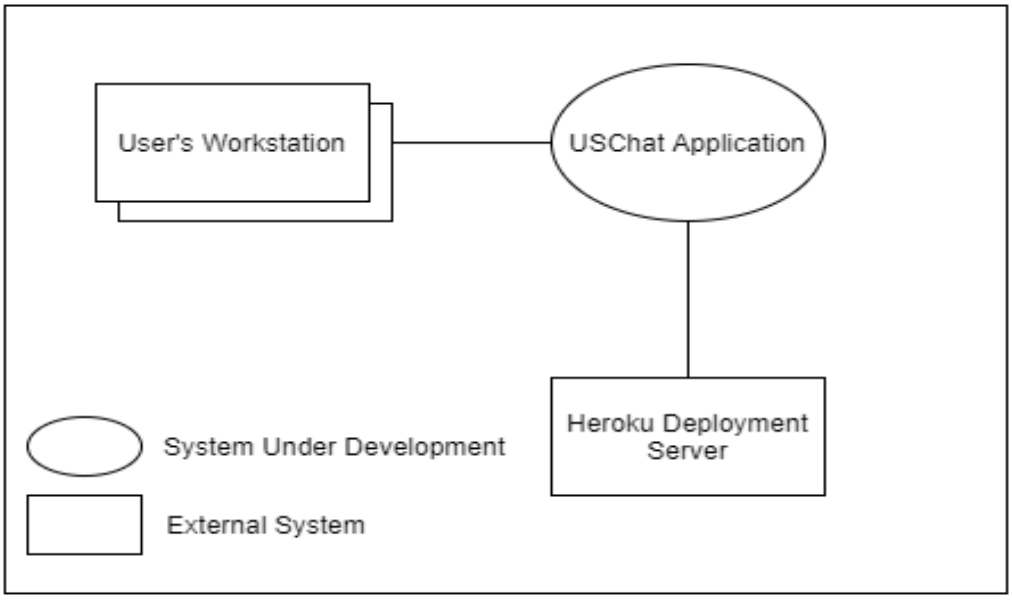


Figure 2.1 *Context diagram for the USChat system*

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

In this case, the element to refine is the entire USChat system. This refinement is performed through decomposition.

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

In this current iteration, the goal of structuring the entire system, design concepts are selected from the criteria presented in Section 1.1. The table below will summarize the selection of design decisions.

<table>
<tr><th>Design Decision and Location</th><th>Rationale</th></tr>
<tr><td>Logically structure the client part of the system using the Web Application reference architecture.</td><td>The Web Application reference architecture supports the development of a web browser that communicates with the Heroku Development Server. This application supports web applications as the majority of it resides on the server, and its architecture is typically composed of three layers.

Discarded Alternatives:
<table><tr><th>Alternative</th><th>Reason for Discarding</th></tr><tr><td>Rich Client Applications</td><td>This reference architecture is oriented towards a local application that is stand-alone. This architecture will not support your system as it requires a server to allow users to use it concurrently. This was discarded due to the usage of local components.</td></tr><tr><td>Rich Internet Applications</td><td>This reference architecture is oriented towards a browser application, however, it is using asynchronous javascript and XML. This was discarded due to not using some of the supported languages.</td></tr><tr><td>Mobile Applications</td><td>This reference architecture is oriented towards handheld devices. This was discarded as this system is not oriented towards mobile devices.</td></tr></table></td></tr>
<tr><td>Physically structure the application using the Three-Tier Deployment pattern.</td><td>Since this system must be accessed from a web browser (CON-5) where (CON-3) and (CON-4) will be considered.</td></tr>
<tr><td>Deploy the application using Heroku Deployment.</td><td>Access to the application is obtained via a web browser (CON-5). This technology also facilitates (CON-1).</td></tr>
</table>

2.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

The instantiation design decisions considered and made are summarized in the following table:

Design Decision and Location	**Rationale**
Create a module that facilitates express endpoints in the business layer within the **Web Application**	The application facade component from the reference architecture is updated to include express endpoints. This will facilitate QA-1 and QA-2 and will help with UC-2, UC-5, UC-6, UC-7, and UC-8
Within the **Three-Tier Deployment,** there will be a module implemented using Handlebars for users	A component is added to the reference architecture to help GUI. This will facilitate QA-1 and QA-4 as well as will help with UC-6.
In the data layer, there is a module in the **Web Application** using Heroku as the main data source	A component from the reference architecture is updated to allow access to Heroku, a hosting application. Facilitates all quality attributes, as well as constraints.

The results of these instantiations decisions are recorded in the next step. In the initial iteration, it is typically too early to precisely define functionality and interfaces. The next iteration will cover how each will be defined in more detail.

2.6 Step 6: Sketch Views and Record Design Decisions

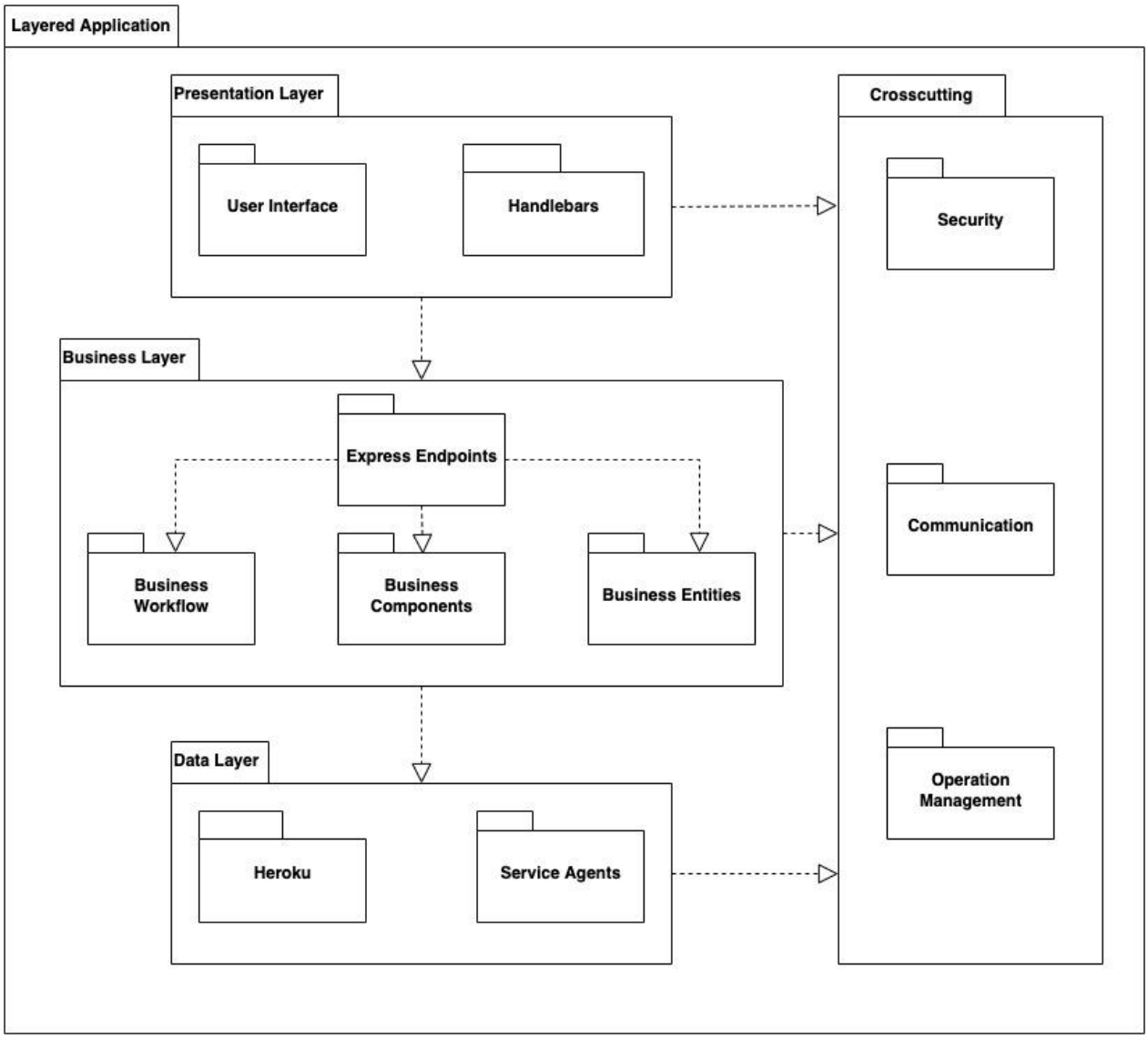


Figure 2.2 *Reference architecture for three-tier architecture*

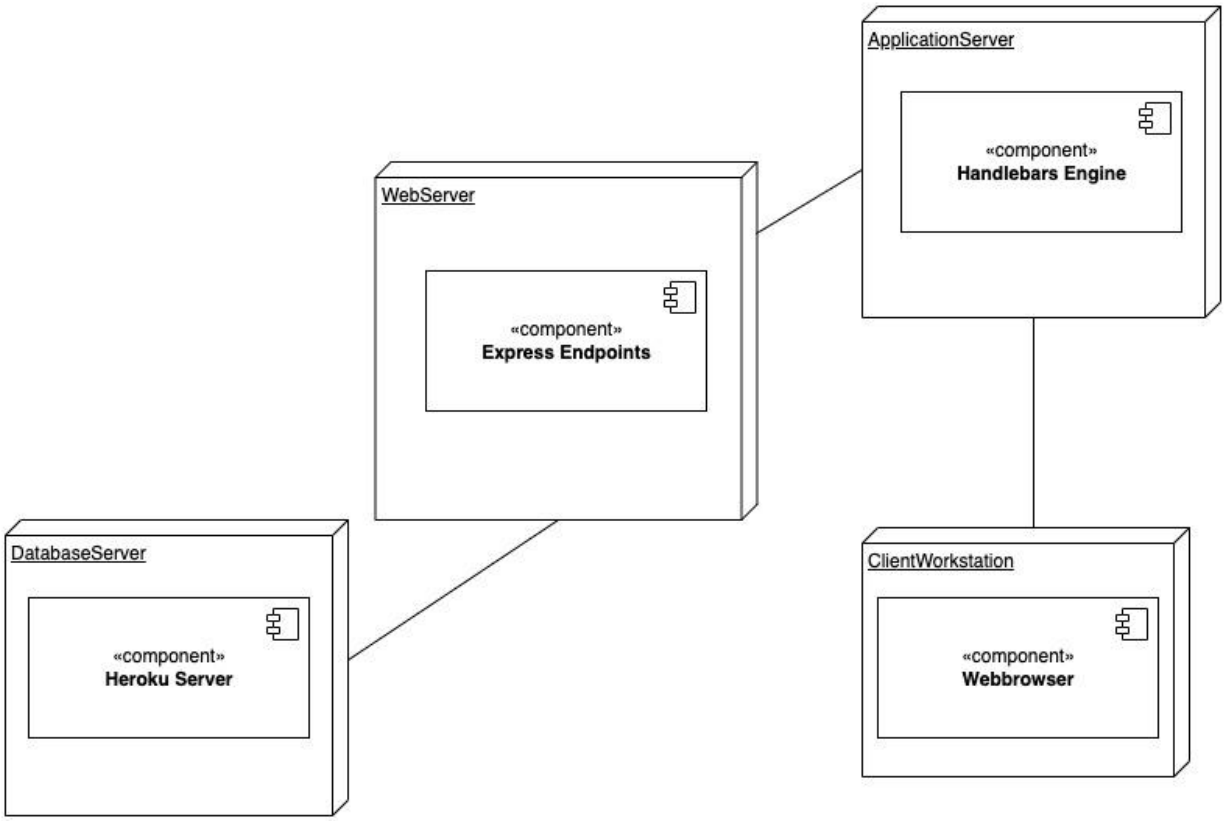


Figure 2.3 *Initial Deployment Diagram*

Element	Responsibility
Presentation Layer	This layer contains modules that control user interaction and use case-control.
User Interface	This module renders the user interface and receives user inputs.
Handlebars	Templating engine, which takes information from server-side and prints it to the presentation layer.
Business Layer	This layer contains modules that perform business logic operations that can be executed to the business layer.
Express.js Endpoints	This is used for endpoints, allowing to build a REST-based project. It allows for defining GET and POST methods. It also lets us send messages between requests and send responses back to the client.
Business Workflow	These components are responsible for managing (long-running) business processes, which may involve the execution of multiple use cases.
Business Components	These components are responsible for retrieving and processing application data and applying business rules to this data.

Business Entities	These components represent the entities from the business domain and their associated business logic.
Data Layer	This layer contains modules that are responsible for data persistence and for communication with the time server.
Heroku	This module is responsible for the persistence of business entities into the relational database.
Service Agents	These components abstract communication mechanisms used to transfer data to external services.
Crosscutting	Modules are applied to the whole system.
Security	These components include cross-cutting functionality that handles security aspects such as authorization and authentication.
Communication	These components include cross-cutting functionality that handles communication mechanisms across layers and physical tiers.
Operation Management	These components include cross-cutting functionality such as exception management, logging, and instrumentation, and validation.

2.7 Step 7: Perform Analysis of Current Design and Review Iteration

The following table summarizes the design progress within iteration 1.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During this Iteration
	UC-1		Selected reference architecture establishes modules that will support this functionality.
	UC-2		Selected reference architecture establishes modules that will support this functionality.
	UC-6		Selected reference architecture establishes modules that will support this functionality.
	UC-9		Selected reference architecture establishes modules that will support this functionality.
	QA-1		Addition of modules within the reference architecture in the data layer to store new users. The details have not been defined.

QA-3		Supported with the inclusion of the Heroku development. But no further details were given.
QA-4		Introduction of three-tier deployment using handler bars. Further details will be provided in later iterations.
QA-5		Addition of data source using Heroku. But no further details were given.
CON-3		Support with the three developments.
CON-4		No relevant decisions were made.
	CON-5	Selection of reference architecture architectures and deployment patterns.

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

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 support 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	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.

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

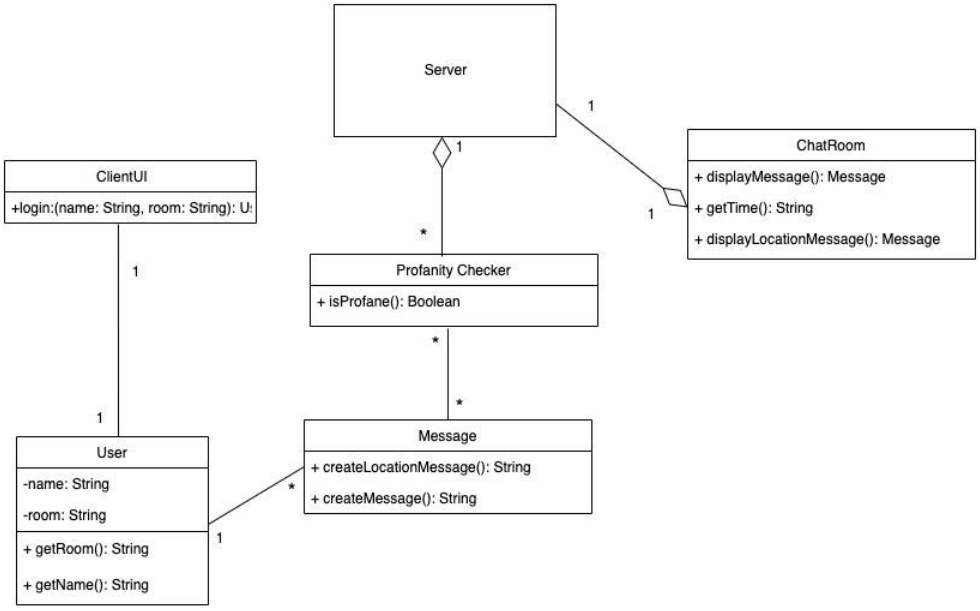


Figure 3.1 *Initial Domain Model*

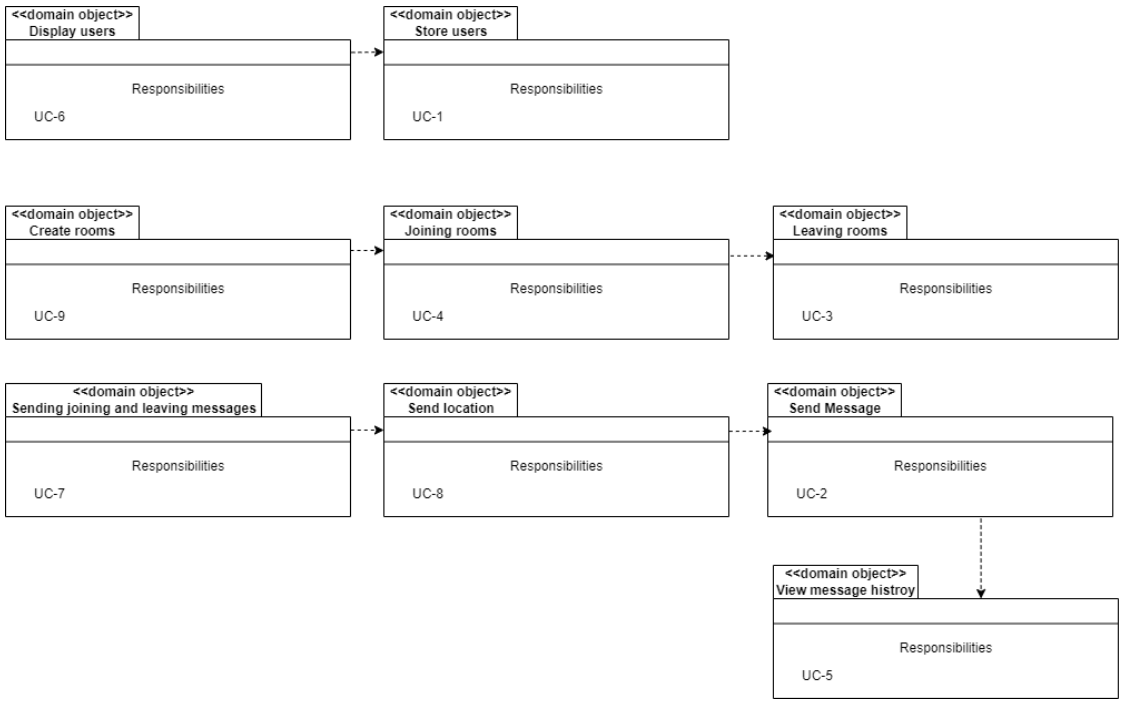


Figure 3.2 *Domain Objects associated with the use case model*

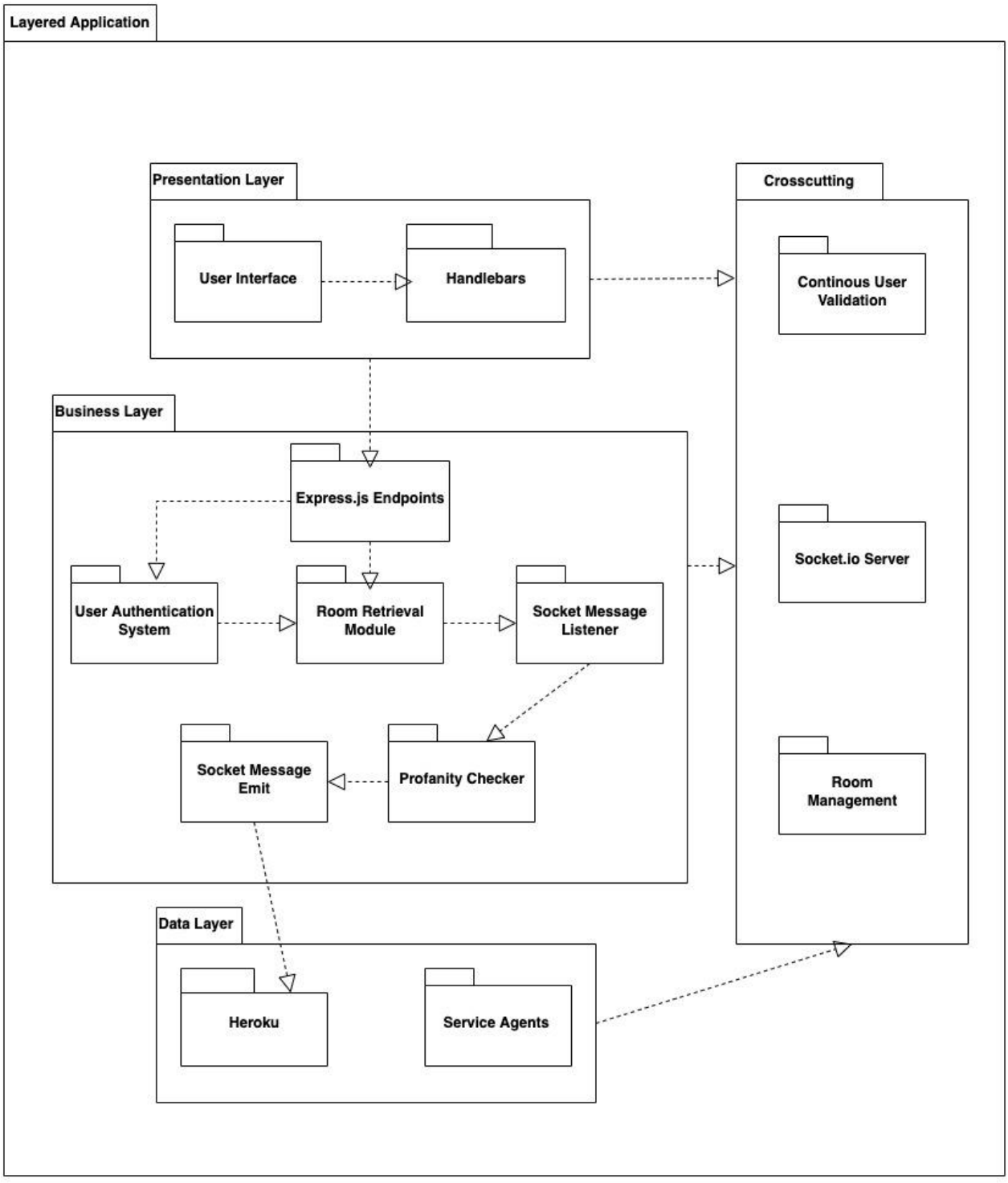


Figure 3.3 *Modules that support the primary use cases*

Element	Responsibility
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 profane 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.

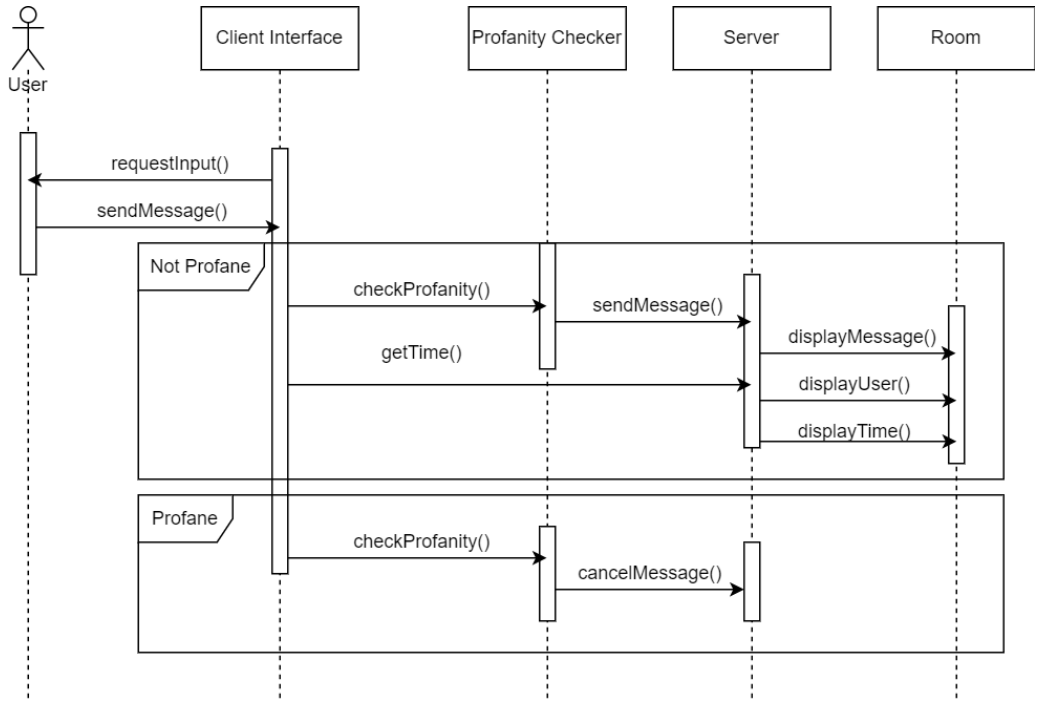


Figure 3.4 *Sequence diagram for UC-2*

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 are sent.

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 modules associated with the rest of the 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 interfaces to support this use case have been identified.
		UC-2	Modules across the layers and preliminary interfaces to support this use case have been identified.
		UC-6	Modules across the layers and preliminary interfaces to support this use case have been identified.
		UC-9	Modules across the layers and preliminary interfaces 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 were made.
	QA-4		The elements that support the associated use cases (All) 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 was made.
CON-5		No relevant decision was 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.

Iteration 3: Addressing Quality Attribute Scenario Driver (QA-3)

In this section, the results of the activities performed in each of the steps in the third iteration of the design process are presented. This iteration is used to build off of the fundamental decisions made in iteration 1 and 2, which leads to an important quality attribute scenario.

Step 2: Establish Iteration Goal By Selecting Drivers

For this iteration, the architect focuses on the QA-3 scenario which sends location coordinates but shifts it slightly.

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

For this availability scenario, the elements that will be refined are the physical nodes that were identified during the second iteration:

- GeoLocation

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

The design concepts used in this iteration are the following:

Design Decisions and Location	Rationale and Assumptions
Introduce the ability to shift user location based on the graphic data taken from GeoLocation	By refining the element system can have increased security as the user location direct location will not be revealed to the other users within the chat app.

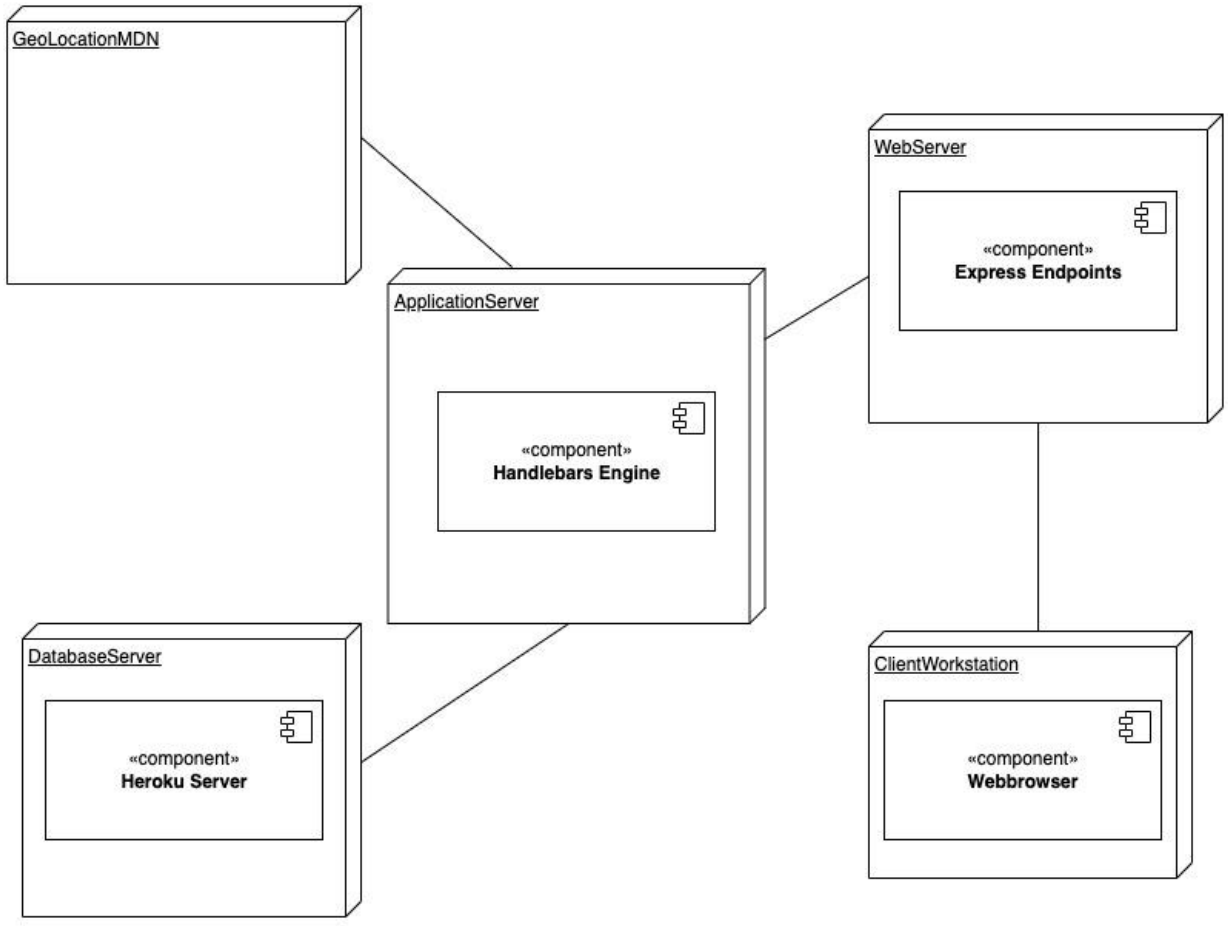
Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

The instantiation design decisions are summarized in the following table:

Design Decisions and Location	Rationale
Deploy a method that will shift the location	The user's location will be taken by the user pressing a button. The location will be processed by GeoLocation MDN, and the geographical coordinates will be shifted. The coordinate will be processed through google's map API and print within the chat

Step 6: Sketch Views and Record Design Decisions

Figure 4.10 shows a refined deployment diagram that includes the introduction of redundancy in the system.



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

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During this Iteration
	QA-1		No relevant decisions were made.
		QA-3	By shifting the location obtained by Geolocation we are able to hide the user's direct location. Thus, giving the user's relative location as opposed to an absolute location.
	QA-4		No relevant decisions were made.
	QA-5		No relevant decisions were made.
	CON-3		No relevant decisions were made.
	CON-4		No relevant decisions were made.
	CON-5		No relevant decisions were made.
	CRN-2		No relevant decisions were made.
	CRN-3		No relevant decisions were made.
	CRN-4		No relevant decisions were made.