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.

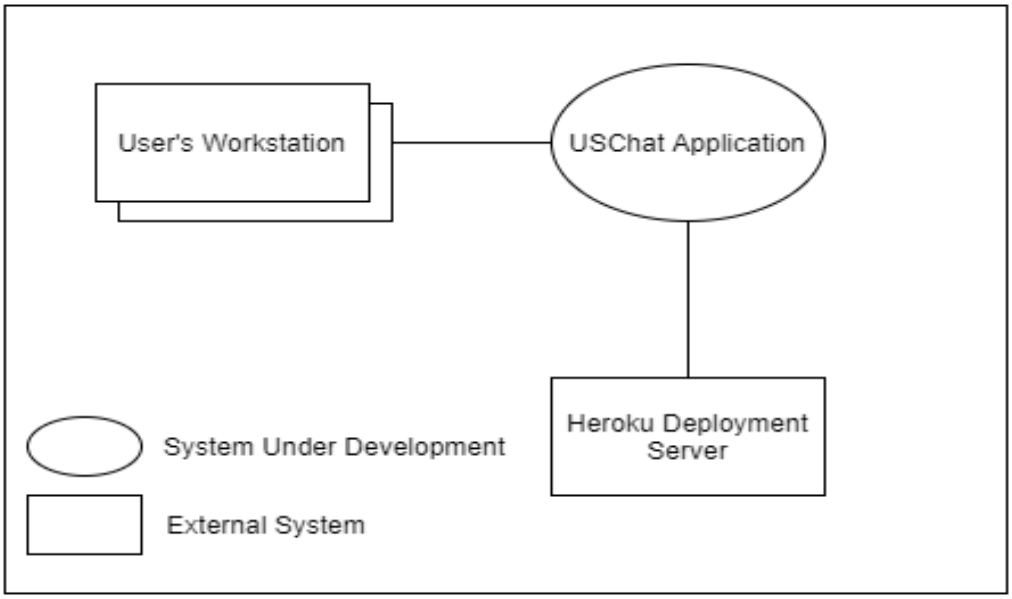

Figure 2.1 *Context diagram for the USChat system*

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:

Alternative — Reason for Discarding

Rich Client Applications — 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.

Rich Internet Applications — 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.

Mobile Applications — This reference architecture is oriented towards handheld devices. This was discarded as this system is not oriented towards mobile devices.</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

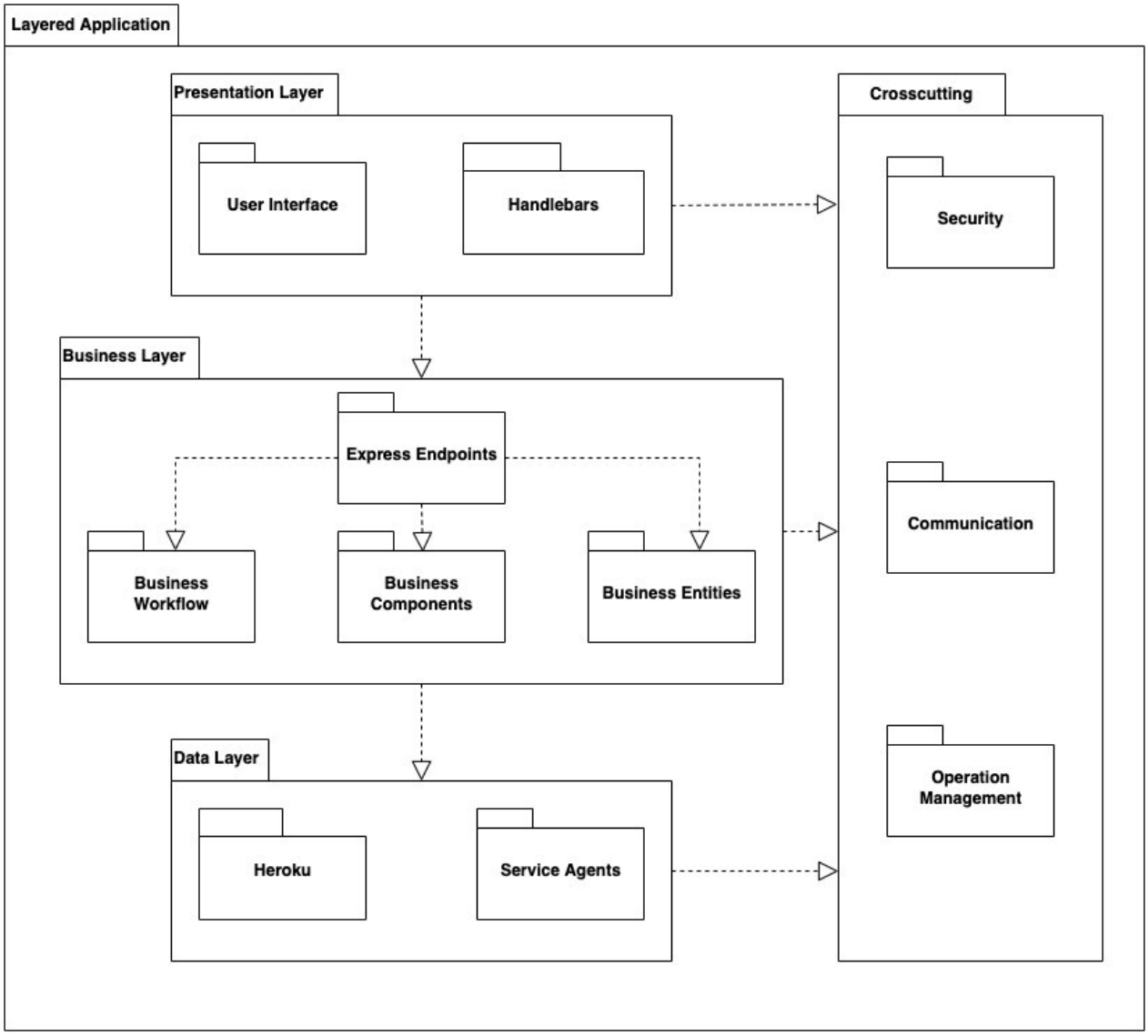

Figure 2.2 *Reference architecture for three-tier architecture*

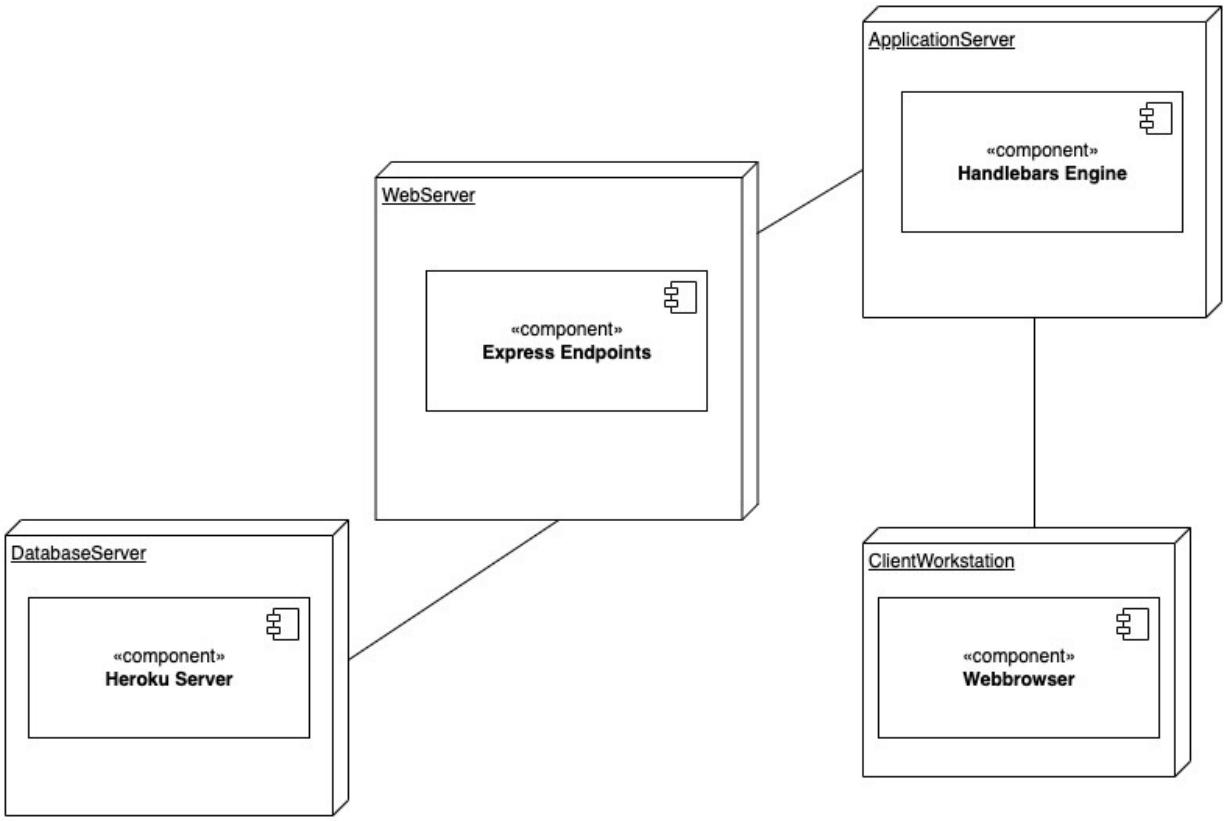

Figure 2.3 *Initial Deployment Diagram*

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.