Zonk Messaging App

Software Design Document

By David Doria

Last Updated: September 24, 2021

Table of Contents

[Scope 3](#_Toc82006233)

[Summary 4](#_Toc82006234)

[Viewpoints 5](#_Toc82006235)

[<Temporary> Table Template 6](#_Toc82006236)

[0 – System View 7](#_Toc82006237)

[1 – Server 9](#_Toc82006238)

[1.1 – Communicator 11](#_Toc82006239)

[1.2 – Host Services 15](#_Toc82006240)

[1.3 – Guest Services 20](#_Toc82006241)

[1.4 – Server Storage 22](#_Toc82006242)

[2 – Client 32](#_Toc82006243)

[2.1 – User Interface 34](#_Toc82006244)

[2.2 – Host 36](#_Toc82006245)

[2.3 – Guest 45](#_Toc82006246)

[2.4 – Server Request JSON 46](#_Toc82006247)

[2.5 – Inter-Client Message JSON 47](#_Toc82006248)

# Scope

# Summary

[Insert Summary Here]

# Viewpoints

UML 2.0 Component Diagram

Dataflow Diagram

Flowcharts

Pseudocode

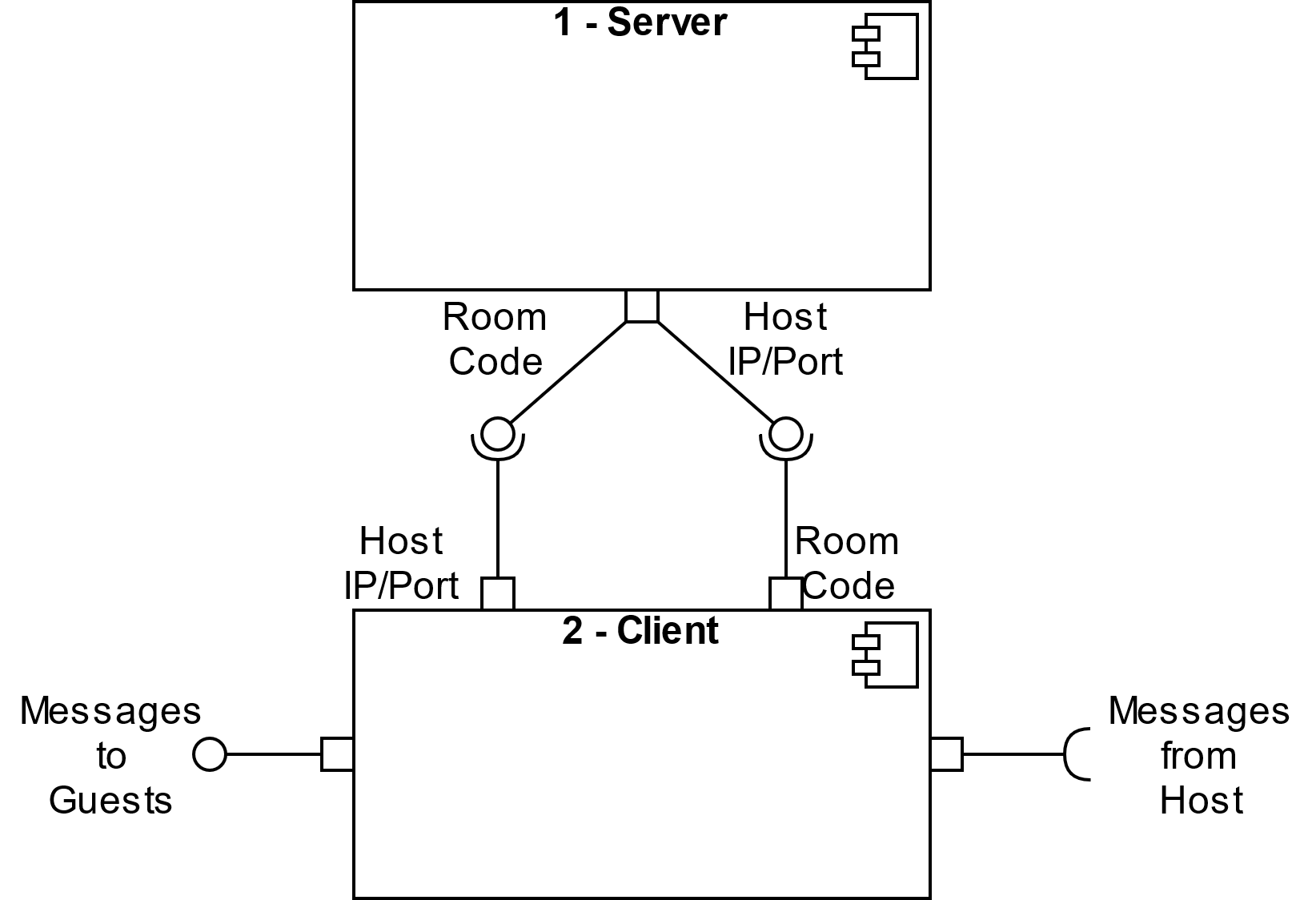
UML Class Diagram

JSON

# <Temporary> Table Template

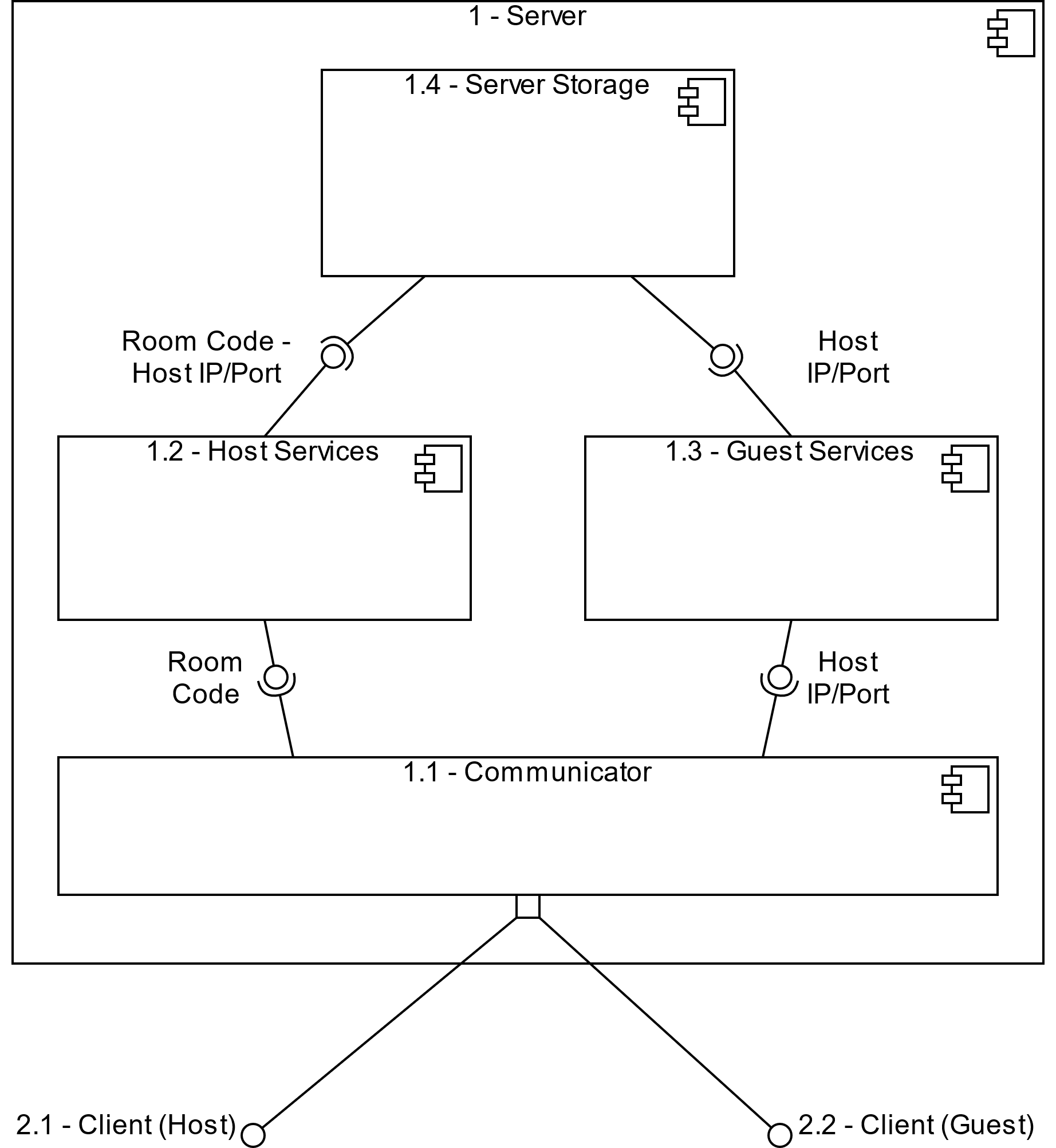
|  |  |
| --- | --- |
| *[View Name]* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

# 0 – System View



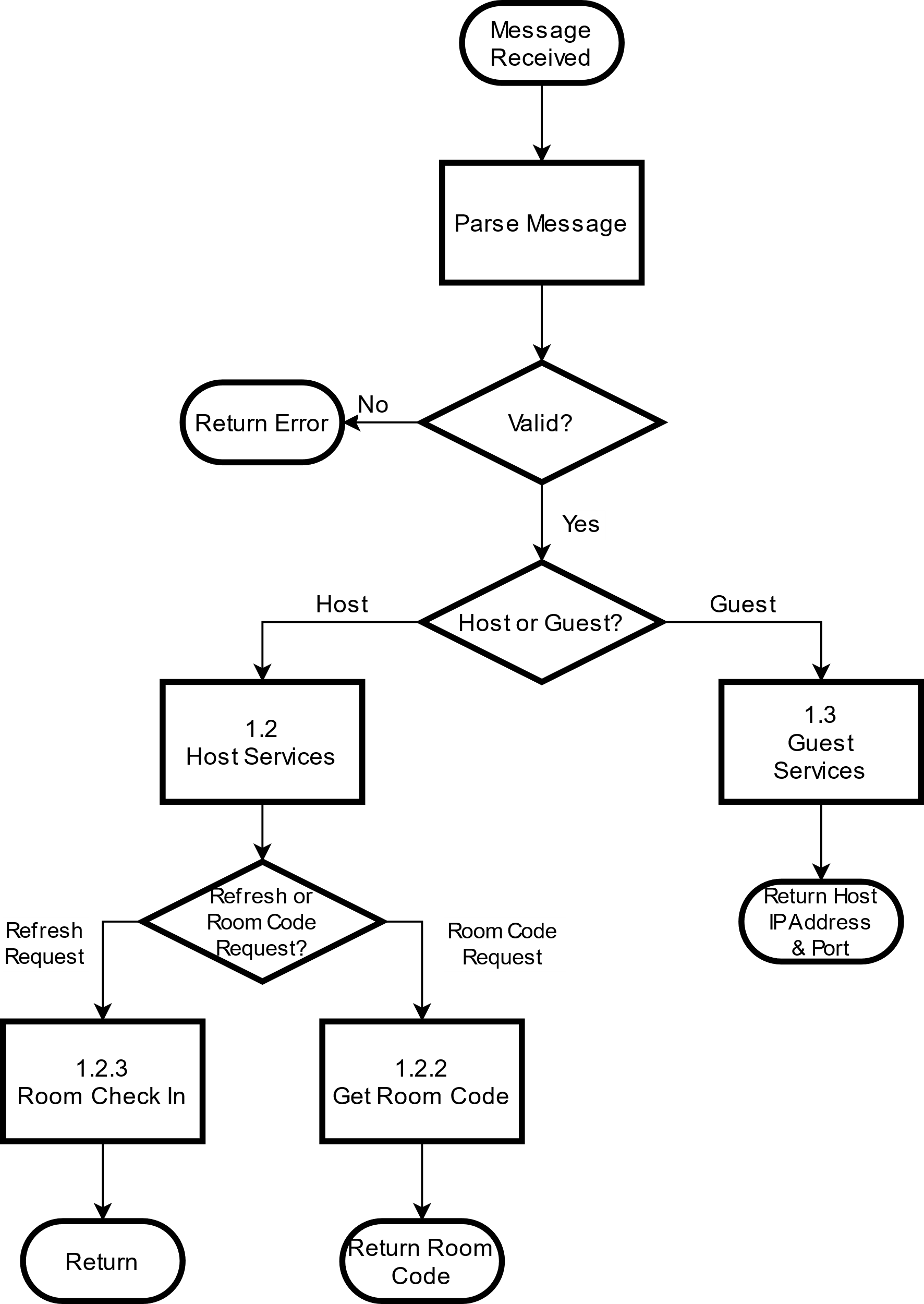
|  |  |
| --- | --- |
| *System View* | |
| Description | Facilitates messaging between two or more [Clients](#_2_–_Client). Uses a one-to-many orchestrated server architecture. A [hosting client](#_2.2_–_Host) communicates with the server to exchange its IP address and port number for a [room code](#_1.2.1_–_Generate_1), which is distributed to other users externally. A [guest client](#_2.3_–_Guest) connects to a [host client](#_2.2_–_Host) by sending the [room code](#_1.2.1_–_Generate_1) to the server in exchange for the host’s IP address and port number. Guests then create a persisting TCP connection with the Host.  Messages sent to or created by the Host are distributed to all other Guests connected to it. |
| Elements | |  |  | | --- | --- | | **1 – Server**  Responsible for maintaining a map containing room codes (the key) and IP addresses/port numbers (the values), validating requests, and distributing room codes, IP addresses and port numbers to requesting clients. | **2 – Client**  The client is what end users will use to communicate with each other. There is only one client, but it has two modes – host and guest. | | **Messages to Guests**  The Host acts as a server to the guests once a persisting TCP connection has been made. Messages sent to the Host from a Guest are distributed to all of the other Guests connected. | **Message from Host**  Messages received from the Host. Any messages sent to the Host, or created by the Host, are distributed to all Guests. | | **Room Code**  A four-letter code used to connect the host client and guest clients. | **Host IP/Port**  The networking information for the Host. It is sent to Guests with the appropriate room code who then connect to the Host directly. | |
| Referenced By | None |
| Viewpoint | UML 2.0 Component Diagram |

## 1 – Server



|  |  |
| --- | --- |
| *Server* | |
| Description | Maintains a map of room codes and [HostEntries](#_1.4.1_–_[Placeholder]). Hosts send their IP address and port number to the server, where [Host Services](#_1.2_–_Host_1) assigns them a room code. The room code is the key for [HostEntry](#_1.4.1_–_[Placeholder]) (which contains the IP address and port number associated with the host) and is stored in a map within [Server Storage](#_1.4_–_[Placeholder]). Guests may send a room code to the server and retrieve the IP address and port number associated with the room code to access the host’s client directly. |
| Elements | |  |  | | --- | --- | | * 1. **- Communicator**   Handles communication with clients. Parses messages from the [Client](#_2_–_Client), and returns fulfilled requests | * 1. **– Host Services**   Assigns a room code to a host’s IP address and port number, and sends the room code, IP address and port number to [Server Storage](#_1.4_–_[Placeholder]). Also returns a room code to the client. | | * 1. **– Guest Services**   Retrieves the IP address and port number of a host from [Server Storage](#_1.4_–_[Placeholder]) when given a room code. | * 1. **– Server Storage**   Maintains a map of unique room codes (keys) and IP addresses/port numbers (values) of hosts. | | **1.4.1 – HostEntry**  This is how the room code and host information is stored in [Server Storage](#_1.4_–_[Placeholder]). It is only used within [Server Storage](#_1.4_–_[Placeholder]). |  | |
| Referenced By | [0 - System View](#_0_-_System) |
| Viewpoint | UML 2.0 Component Diagram |

### 1.1 – Communicator



|  |  |
| --- | --- |
| *Communicator* | |
| Description | Handles receiving, parsing and returning messages from the [Client](#_2_–_Client). |
| Elements | |  |  | | --- | --- | | **Parse Message**  Reads the message sent from the client, validates its contents, and then passes its contents to either [Host](#_1.2_–_Host_1) or [Guest Services](#_1.3_–_Guest). | **1.1.1 – Communicator Format**  The format the [Communicator](#_1.1_–_Communicator) expects data to be in when they are returned to it from [Get Room Code](#_1.2.2_–_Get) or [Get Host Info](#_1.3.1_–_Get). | | **1.2 – Host Services**  Assigns a room code to a host’s IP address and port number. | **1.2.2 – Get Room Code**  Carries out requests for room codes, including updating available room codes in [Server Storage](#_1.4_–_[Placeholder]). | | **1.2.3 – Room Check In**  Waits for host clients to check in their room keys. If a host client fails to check in, its room key will be released. | **1.3 – Guest Services**  Retrieves the IP address and port number of a host from [Server Storage](#_1.4_–_[Placeholder]). | |
| Referenced By | [1 – Server](#_1_–_Server), [1.2 – Host Services](#_1.2_–_Host_1), [1.3 – Guest Services](#_1.3_–_Guest) |
| Viewpoint | Flowchart |

#### 1.1.1 – Communicator Format

[Return Code]-[Payload]

|  |  |
| --- | --- |
| *Communicator Format* | |
| Description | The format the [Communicator](#_1.1_–_Communicator) expects data to be in when they are returned to it from [Get Room Code](#_1.2.2_–_Get) or [Get Host Info](#_1.3.1_–_Get). The entire thing is turned into a String and then returned to the [Communicator](#_1.1_–_Communicator). |
| Elements | |  |  | | --- | --- | | **1.4.4 – Return Codes**  The different status codes that may be returned to the caller. | **Payload**  A String that holds the information to be returned to the requesting [Client](#_2_–_Client).  If there is no payload, a 0 should be returned | |  |  | |
| Referenced By | [1.1 – Communicator](#_1.1_–_Communicator) |
| Viewpoint | Pseudocode |

#### 1.1.2 – Client Response JSON

{

“clientResponse”: [

// HOST if it is in response to a host, or GUEST if it is

// in response to a guest.

“type” : “*type*”,

“time” : *“currentTime”*,

“date” : *“mm/dd/yyyy”*,

// Null for a host response

“networkingInfo” : [

“IP” : *“IPAdress”,*

“Port” : “*portNumber”*

],

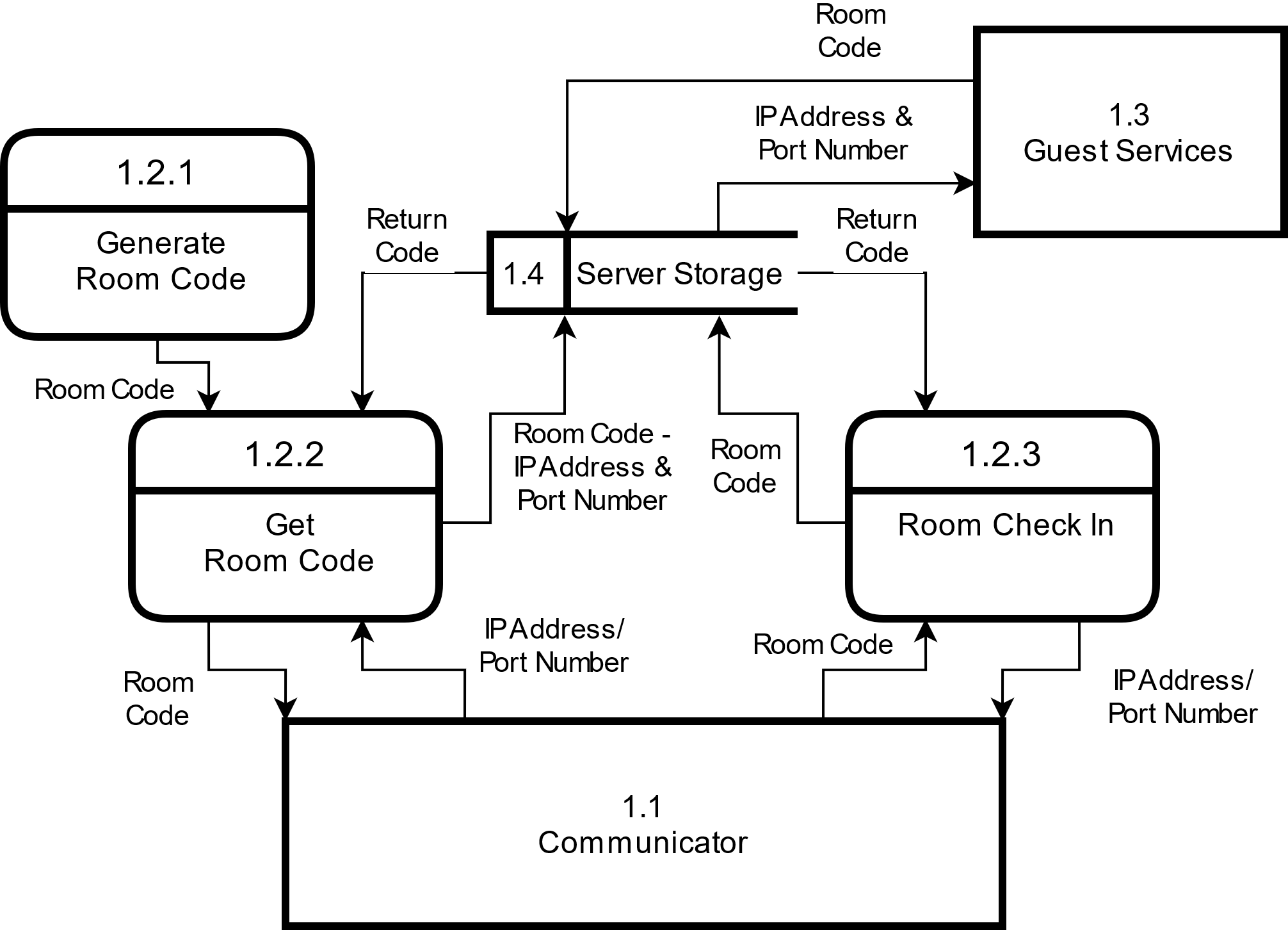
// Null for a guest response

“roomCode” : “*room code*”,

]}

|  |  |
| --- | --- |
| *Client Response JSON* | |
| Description | JSON format for a response from the [Server](#_1_–_Server) to a requesting [Client](#_2_–_Client). |
| Elements | |  |  | | --- | --- | | **type**  The type of [client](#_2_–_Client) that made the request the [Server](#_1_–_Server) is now responding to ([Host](#_2.2_–_Host), [Guest](#_2.3_–_Guest), or [refresh](#_2.2.6_–_[Placeholder])) | **networkingInfo**  An array that holds the IP Address and the Port Number | | **Room Code**  A somewhat random, 4-character, alphabetic room code that is used to connect [Hosts](#_2.2_–_Host) and [Guests](#_2.3_–_Guest). (See [1.2.1 – Generate Room Code](#_1.2.1_–_Generate_1)) |  | |
| Referenced By | [1.1 – Communicator](#_1.1_–_Communicator), [2.4 – Server Request JSON](#_2.4_–_Server) |
| Viewpoint | JSON |

### 1.2 – Host Services



|  |  |
| --- | --- |
| *Host Services* | |
| Description | Assigns a room code to a host’s IP address and port number. Periodically, host clients check in with the server. If a host fails to check in, the room code will be released to be used by other hosts. |
| Elements | |  |  | | --- | --- | | **1.1 – Communicator**  Handles communication with clients. Parses messages from the client and returns fulfilled requests. | **1.2.1 – Generate Room Code**  Generates a somewhat random, 4-character, alphabetic room code. | | **1.2.2 – Get Room Code**  Carries out requests for room codes, including updating [Server Storage](#_1.4_–_[Placeholder]). | **1.2.3 – Room Check In**  Waits for host clients to check in their room keys. If a host client fails to check in, its room key will be released. | | **1.3 – Guest Services**  Handles requests from guest clients. | **1.4 – Server Storage**  Maintains a map of unique room codes (keys) and IP addresses/port numbers (values) of hosts. | | **1.4.4 – Return Codes**  The different status codes that may be returned to the caller by [Server Storage](#_1.4_–_[Placeholder]). It is exists as an enumeration. |  | |
| Referenced By | [1 – Server](#_1_–_Server), [1.1 – Communicator](#_1.1_–_Communicator) |
| Viewpoint | Dataflow Diagram |

#### 1.2.1 – Generate Room Code

// Generates a room code for the client.

// 4 characters, uppercase letters only

**String generateRoomCode()**

// 24 Hour Time hashed

char1 = intToAlph((3\*currentHour + 1) mod 25)

// 60 Minutes hashed

char2 = intToAlph((currentMin\*2) mod 25)

// Random number from 0-1000

char3 = intToAlph(random(0, 1000) mod 25)

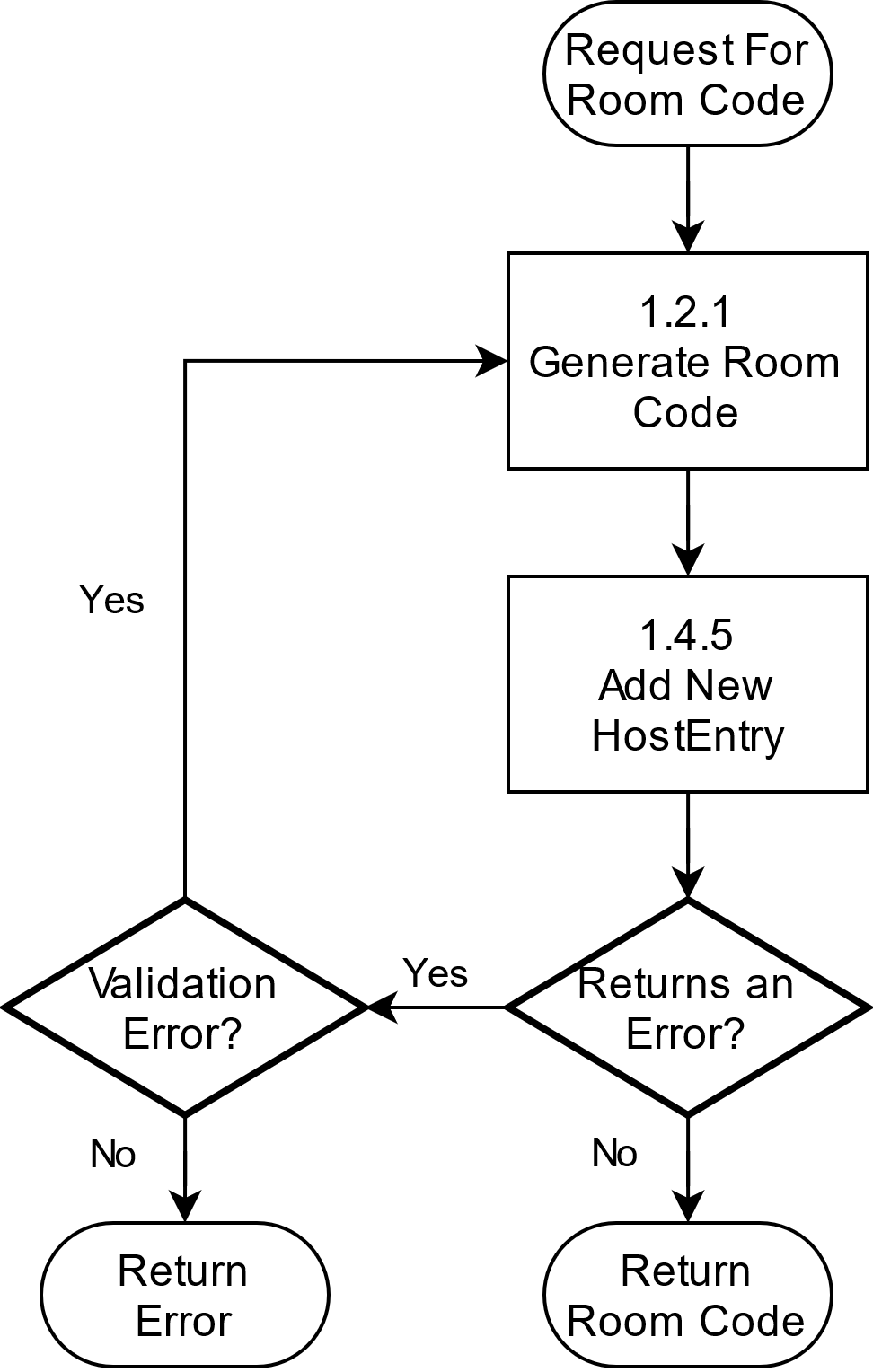
// Random number from 0-1000

char4 = intToAlph(random(0, 1000) mod 25)

**return uppercase( char1 + char2 + char3 + char4 )**

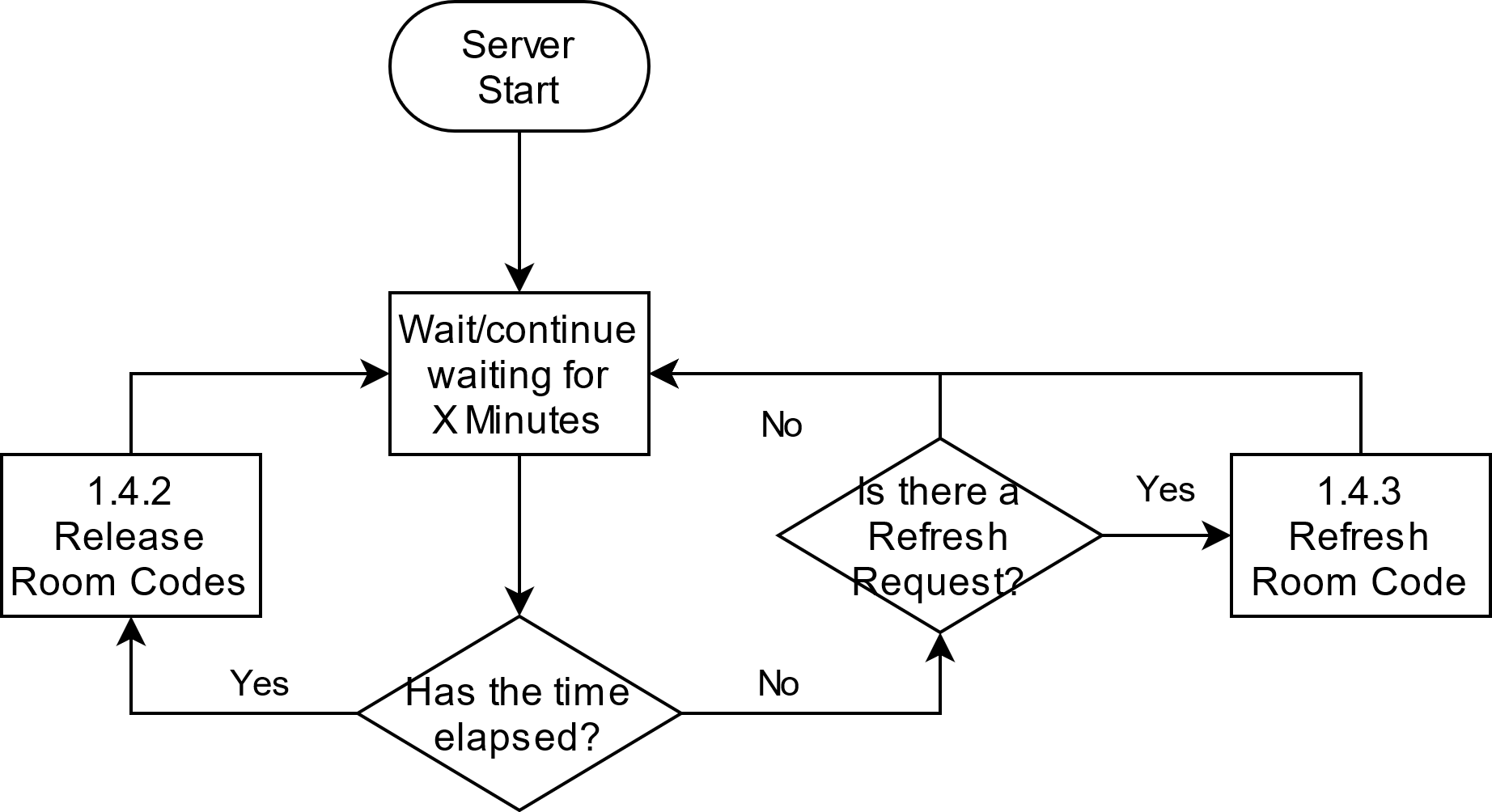
|  |  |
| --- | --- |
| *Generate Room Code* | |
| Description | Generates a somewhat random, 4-character, alphabetic room code. The first two letters are based on the time, while the second two letters are random. |
| Elements | |  |  | | --- | --- | | **currentHour/currentMin**  The current hour/min in the server’s time zone | **intToAlph(int)**  Turns a number (1-26) to its corresponding letter | |
| Referenced By | [1.2 – Host Services](#_1.2_–_Host), [1.2.2 – Get Room Code](#_1.2.2_–_Get) |
| Viewpoint | Pseudocode |

#### 1.2.2 – Get Room Code



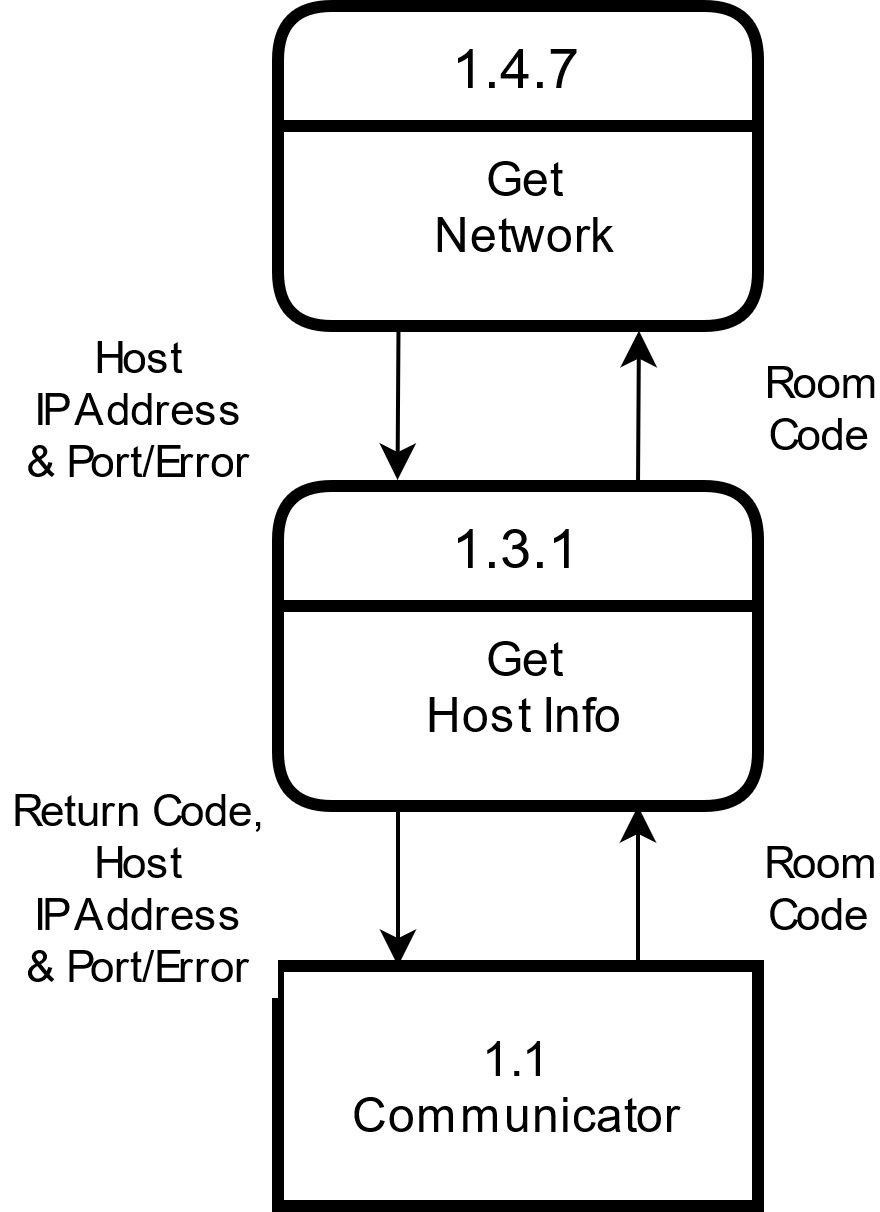
|  |  |
| --- | --- |
| *Get Room Code* | |
| Description | Carries out requests for room codes, including updating available room codes in [Server Storage](#_1.4_–_[Placeholder]). If [Add New HostEntry](#_1.4.5_–_Add) returns a validation error, a new room code will be generated until it is successful. Upon successful completion of the request, the room code is [formatted](#_1.2_–_Host) and returned to the [Communicator](#_1.1_–_Communicator). |
| Elements | |  |  | | --- | --- | | **1.2.1 – Generate Room Code**  Generates a somewhat random, 4-character, alphabetic room code. | **1.4 – Server Storage**  Maintains a map of unique room codes (keys) and IP addresses/port numbers (values) of hosts. | | **1.4.5 – Add New HostEntry**  Adds a new HostEntry to the Map in [Server Storage](#_1.4_–_[Placeholder]). |  | |
| Referenced By | [1.2 – Host Services](#_1.2_–_Host), [1.4 – Server Storage](#_1.4_–_[Placeholder]) |
| Viewpoint | Flowchart |

#### 1.2.3 – Room Check In



|  |  |
| --- | --- |
| *Room Check In* | |
| Description | Waits for host clients to check in their room keys. If a host client fails to check in, its room key, and the [HostEntry](#_1.4.1_–_[Placeholder]) associated with it, will be released. Room keys are released periodically. The exact time is set in minutes at server start. |
| Elements | |  |  | | --- | --- | | **1.4.2 – Release Room Codes**  An operation triggered by Room Check In and carried out by [Server Storage](#_1.4_–_[Placeholder]). It releases room codes and their associated [HostEntries](#_1.4.1_–_[Placeholder]) if they are not in use. | **1.4.3 – Refresh Room Code**  Fulfils Refresh Requests and sets a [HostEntry](#_1.4.1_–_[Placeholder])’s “refreshed” field to true. | | **X Minutes**  To be determined at Server Start. Determines how often Room Codes are released. | **Refresh Request**  A request sent automatically by the host to refresh the [HostEntry](#_1.4.1_–_[Placeholder])’s “inUse” field, which is used in determining which codes should be released. | |
| Referenced By | [1.2 – Host Services](#_1.2_–_Host), [1.4 – Server Storage](#_1.4_–_[Placeholder]) |
| Viewpoint | Flowchart |

### 1.3 – Guest Services



|  |  |
| --- | --- |
| *Guest Services* | |
| Description | Retrieves the IP address and port number of a host from [Server Storage](#_1.4_–_[Placeholder]) when given a room code. If the room code is not in use, it will return an error. |
| Elements | |  |  | | --- | --- | | **1.1 – Communicator**  Handles receiving, parsing, and returning messages from the client. | **1.3.1 – Get Host Info**  Makes the request to [Server Storage](#_1.4_–_[Placeholder]) for a Host IP Address and Port number. | | **1.4 – Server Storage**  Maintains a map of unique room codes (keys) and IP addresses/port numbers (values) of hosts. | **1.4.4 – Return Codes**  The different status codes that may be returned to the caller. | |
| Referenced By | [1 – Server](#_1_–_Server), [1.1 – Communicator](#_1.1_–_Communicator), [1.2 – Host Services](#_1.2_–_Host_1), [1.4 – Server Storage](#_1.4_–_[Placeholder]) |
| Viewpoint | Data Flow Diagram |

#### 1.3.1 – Get Host Info

// Requests Host Networking Information from Server Storage

**String getHostInfo(String roomCode)**

status, hostInfo = serverStorage.get(roomCode)

if status is an error

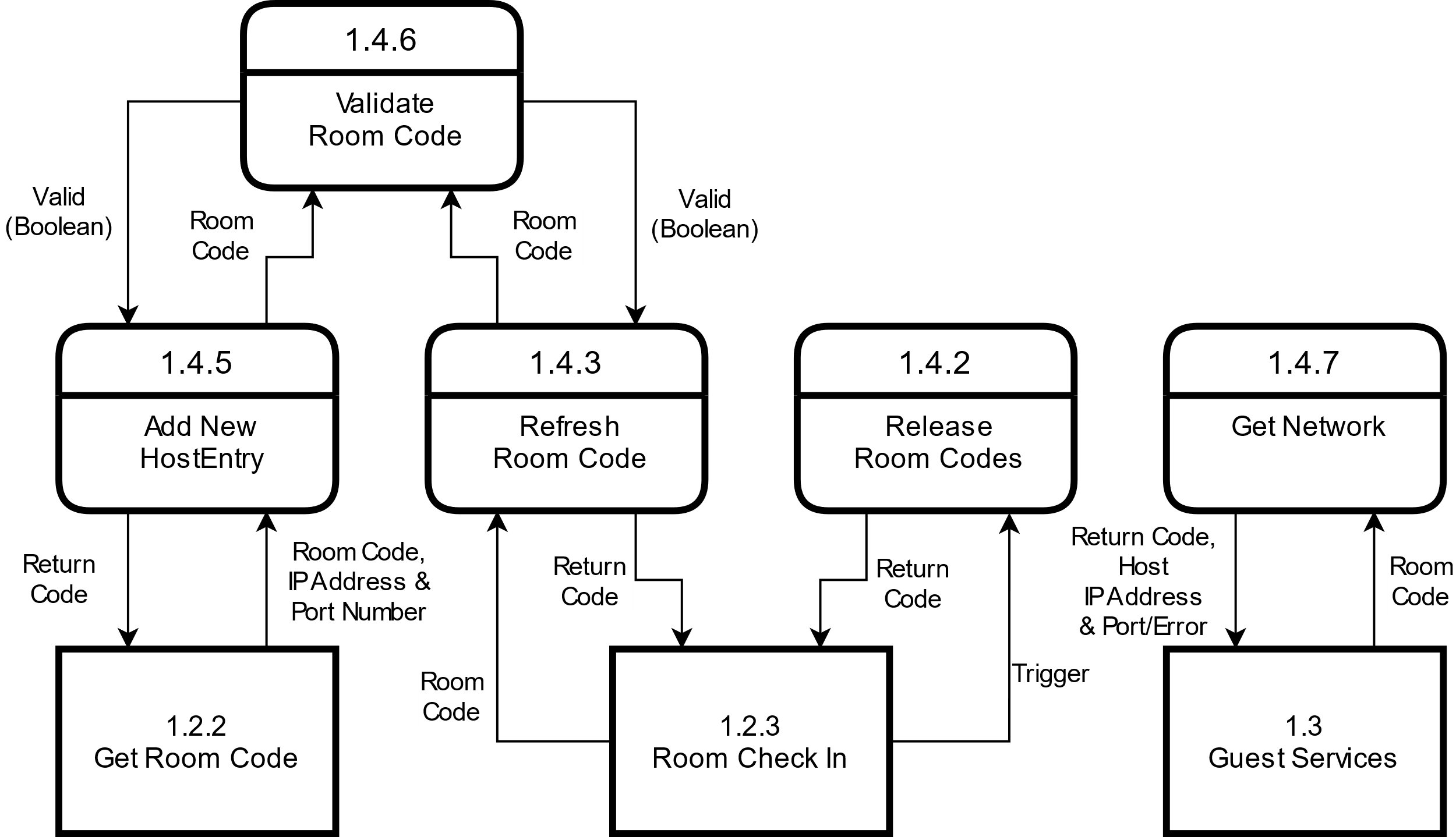
**return errorCode**

else

**return hostInfo**

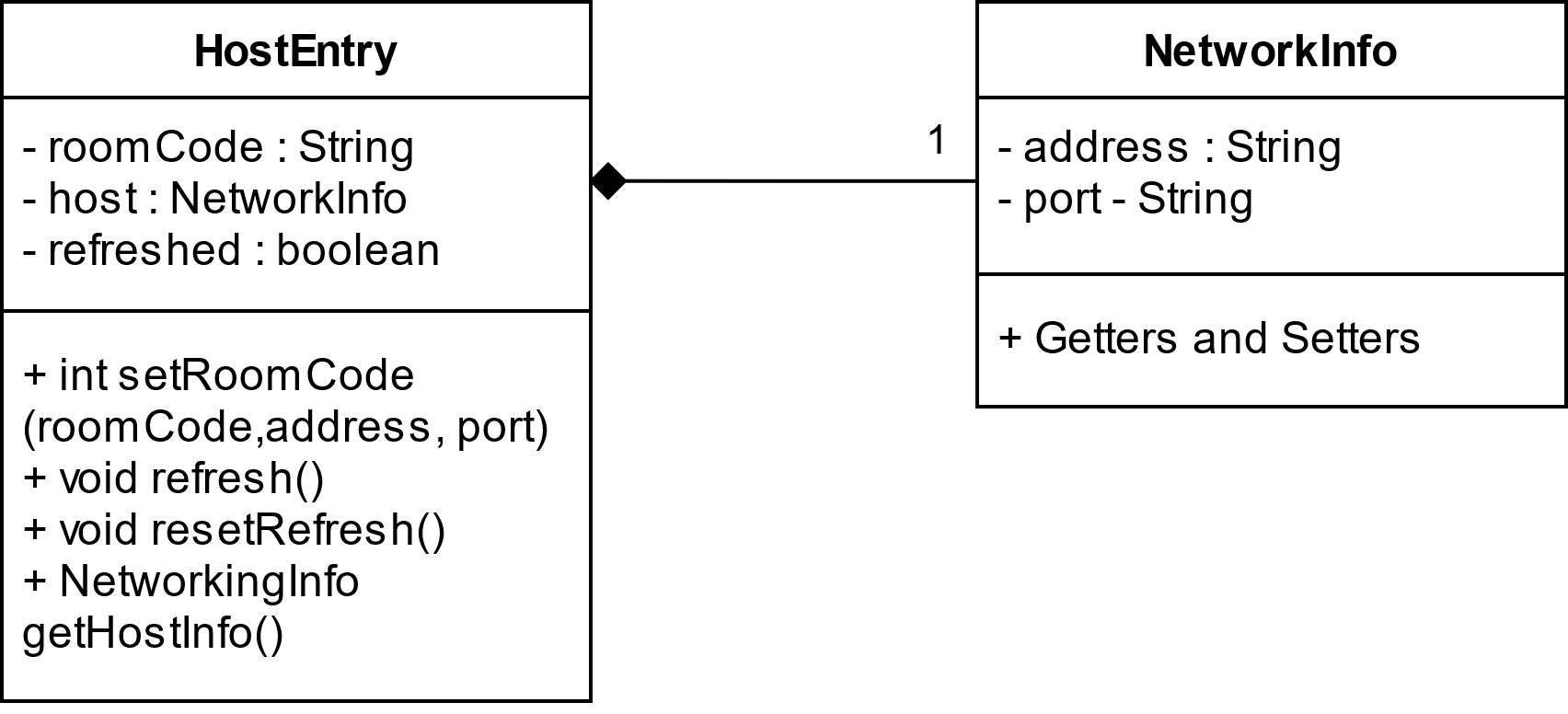
|  |  |
| --- | --- |
| *Get Host Info* | |
| Description | The algorithm for getting Host networking information from [Server Storage](#_1.4_–_[Placeholder]). If [Server Storage](#_1.4_–_[Placeholder]) returns an all-clear status (along with the hostInfo), it is [formatted](#_1.2_–_Host) and returned to the Communicator. If it returns an error, that error is [formatted](#_1.1.1_–_Communicator) and returned to the Communicator |
| Elements | |  |  | | --- | --- | | **roomCode**  a somewhat random, 4-character, alphabetic room code. | **serverStorage**  Maintains a map of unique room codes (keys) and IP addresses/port numbers (values) of hosts. See [1.4 – Server Storage](#_1.4_–_[Placeholder]) | | **status**  An int returned by [Server Storage](#_1.4_–_[Placeholder]). See [1.4.4 – Return Codes](#_1.4.4_–_Return). | **hostInfo**  The IP Address and Port number of the Host Client | |
| Referenced By | [1.3 – Guest Services](#_1.3_–_Guest) |
| Viewpoint | Pseudocode |

### 1.4 – Server Storage



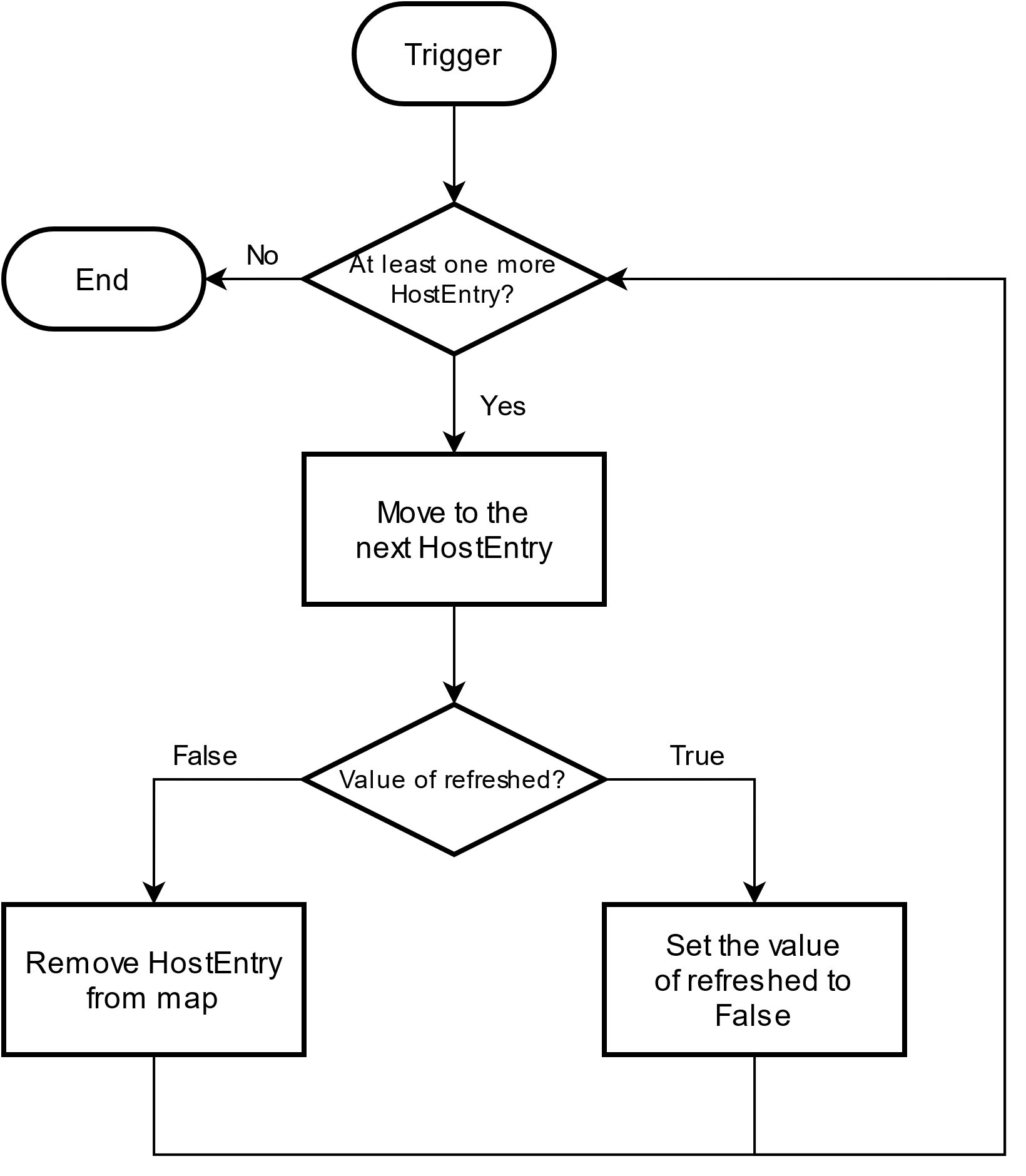
|  |  |
| --- | --- |
| *Server Storage* | |
| Description | Maintains a map of unique room codes (keys) and IP addresses/port numbers (values) of hosts. |
| Elements | |  |  | | --- | --- | | **1.4.1 – HostEntry**  The [HostEntry](#_1.4.1_–_[Placeholder]) class. This is how the room code and host information is stored in Server Storage. It is only used within Server Storage. | **1.4.2 – Release Room Codes**  Gets rid of [HostEntries](#_1.4.1_–_[Placeholder]) that have not checked in with the server via [Refresh Room Code](#_1.4.3_–_[Placeholder]). | | **1.4.3 – Refresh Room Code**  Triggered by a Refresh Request by the [Client](#_2_–_Client). Sets the refreshed value within [HostEntry](#_1.4.1_–_[Placeholder]) to true. | **1.4.4 – Return Codes**  The different status codes that may be returned to the caller. | | **1.4.5 – Add New HostEntry**  Adds a new [HostEntry](#_1.4.1_–_[Placeholder]) to Server Storage. | **1.4.6 – Validate Room Code**  Makes sure that the room code is valid and unique within the map of room codes. | | **1.4.7 – Get Host Information**  Retrieves the Host’s network information. |  | |
| Referenced By | [1 – Server](#_1_–_Server) |
| Viewpoint | Data Flow Diagram |

#### 1.4.1 – HostEntry



|  |  |
| --- | --- |
| *HostEntry* | |
| Description | This is how the room code and host information is stored in Server Storage, and acts as the value to the room key’s key within the map. It is only used within Server Storage. |
| Elements | |  |  | | --- | --- | | **NetworkInfo**  An object that only holds networking information | **refreshed : Boolean**  It is set to true when a refresh request is made, and set to false after it been checked by [Release Room Codes](#_1.4.2_–_[Placeholder]) | | **setRoomCode ( )**  Sets a new Host Entry within the map. | **getHostInfo ()**  Returns NetworkInfo. | |
| Referenced By | [1.4 – Server Storage](#_1.4_–_Server) |
| Viewpoint | UML Class Diagram |

#### 1.4.2 – Release Room Codes



|  |  |
| --- | --- |
| *Release Room Codes* | |
| Description | Gets rid of [HostEntries](#_1.4.1_–_[Placeholder]) that have not checked in with the server via [Refresh Room Code](#_1.4.3_–_[Placeholder]). HostEntries that have checked in before Release Room Codes is called have their “refreshed” field set to false. |
| Elements | |  |  | | --- | --- | | **1.4.1 – HostEntry**  How the room code and host information is stored in [Server Storage](#_1.4_–_[Placeholder]), and acts as the value to the room key’s key within the map. It is only used within [Server Storage](#_1.4_–_[Placeholder]). | **refreshed**  A property of [HostEntry](#_1.4.1_–_[Placeholder]). It is set to true when a refresh request is made, and set to false after it been checked by Release Room Codes | |
| Referenced By | [1.4 – Server Storage](#_1.4_–_[Placeholder]) |
| Viewpoint | Flowchart |

#### 1.4.3 – Refresh Room Code

// Sets the refresh field within HostEntry to true

**Void refreshRoomCode(String roomCode)**

Get HostEntry with roomCode as its key from the Map

HostEntry.refresh()

**return**

|  |  |
| --- | --- |
| *[View Name]* | |
| Description | Triggered by a Refresh Request by the [Client](#_2_–_Client). Sets the refreshed field within [HostEntry](#_1.4.1_–_[Placeholder]) to true. |
| Elements | |  |  | | --- | --- | | **Map**  [Server Storage](#_1.4_–_[Placeholder])’s main data structure. It maintains a map of unique room codes (keys) [HostEntries](#_1.4.1_–_[Placeholder]) (values). | **­ refresh()**  A method of [HostEntry](#_1.4.1_–_HostEntry). It sets the refresh field to true. | |
| Referenced By | [1.4 – Server Storage](#_1.4_–_[Placeholder]) |
| Viewpoint | Pseudocode |

#### 1.4.4 – Return Codes

1 – Success

2 – Invalid Room Code

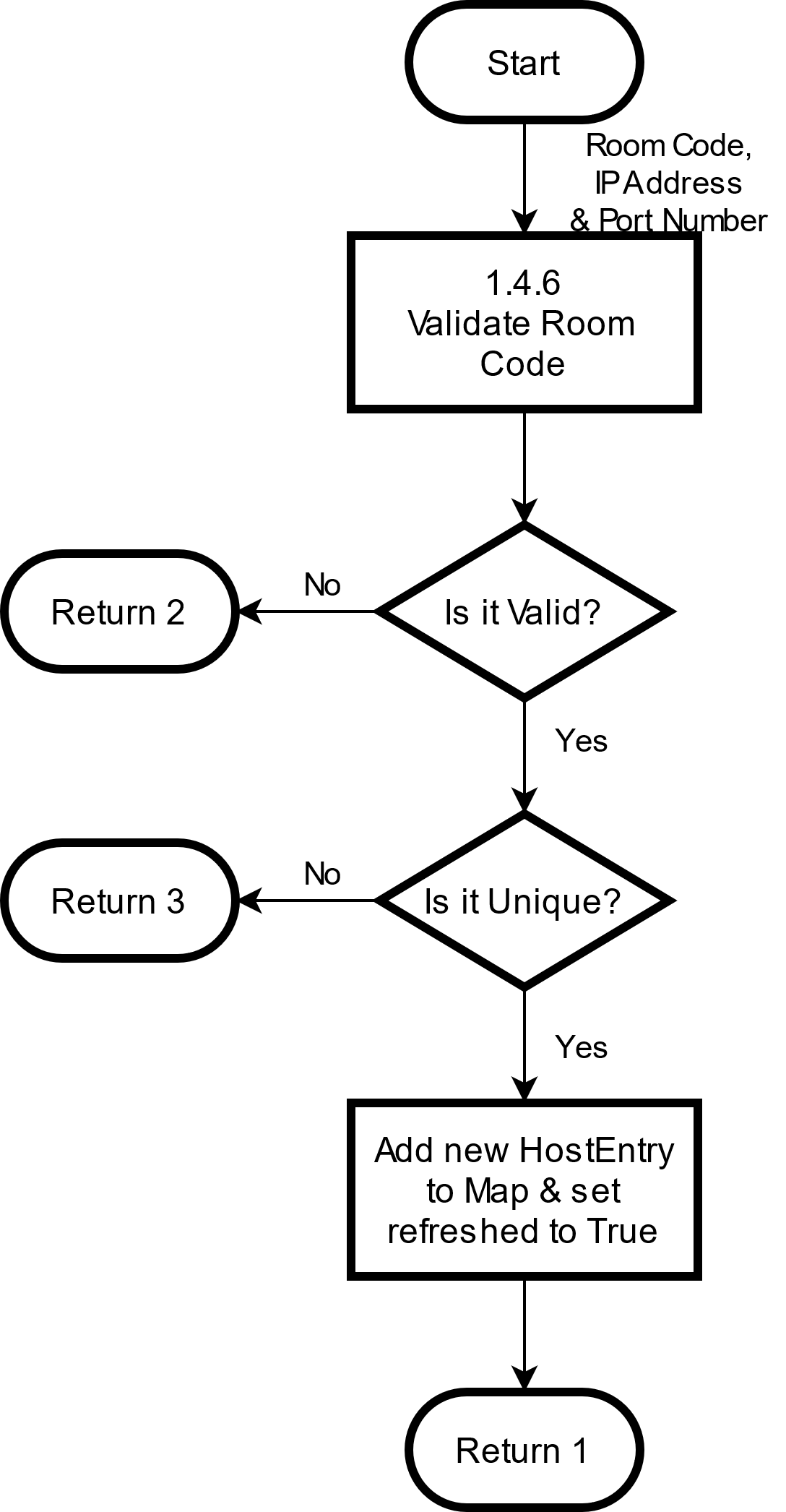
3 – Not a Unique Room Code

4 – Room Code Not Found

5 – Unspecified Error

|  |  |
| --- | --- |
| *Return Codes* | |
| Description | The different status codes that may be returned to the caller by [Server Storage](#_1.4_–_[Placeholder]). It is exists as an enumeration. |
| Referenced By | [1.4 – Server Storage](#_1.4_–_[Placeholder]) |
| Viewpoint | Pseudocode |

#### 1.4.5 – Add New HostEntry



|  |  |
| --- | --- |
| *Add New HostEntry* | |
| Description | Adds a new [HostEntry](#_1.4.1_–_[Placeholder]) to Server Storage. |
| Elements | |  |  | | --- | --- | | **1.4.1 – HostEntry**  How the room code and host information is stored in [Server Storage](#_1.4_–_[Placeholder]), and acts as the value to the room key’s key within the map. It is only used within [Server Storage](#_1.4_–_[Placeholder]). | **1.4.6 – Validate Room Code**  Makes sure that the room code is valid and unique within the map of room codes. | | **1.4.4 – Return Codes**  The different status codes that may be returned to the caller by [Server Storage](#_1.4_–_[Placeholder]). | **refreshed**  A property of [HostEntry](#_1.4.1_–_[Placeholder]). It is set to true when a refresh request is made, and set to false after it been checked by Release Room Codes. | | **Map**  [Server Storage](#_1.4_–_[Placeholder])’s main data structure. It maintains a map of unique room codes (keys) [HostEntries](#_1.4.1_–_[Placeholder]) (values). |  | |
| Referenced By | [1.4 – Server Storage](#_1.4_–_[Placeholder]) |
| Viewpoint | Flowchart |

#### 1.4.6 – Validate Room Code

// Makes sure that an incoming room code is both valid (It adheres to the standard format for room codes. See [1.2.1 – Generate Room Code](#_1.2.1_–_Generate_1)) and unique (It does not exist within the Map of room codes stored within the server

**Boolean validateRoomCode( String roomCode )**

// Validation Checks

If roomCode.length != 4

**Return False**

Else If roomCode is not fully alphabetic

**Return False**

// Uniqueness Check

Else If roomCode exists as key in Map

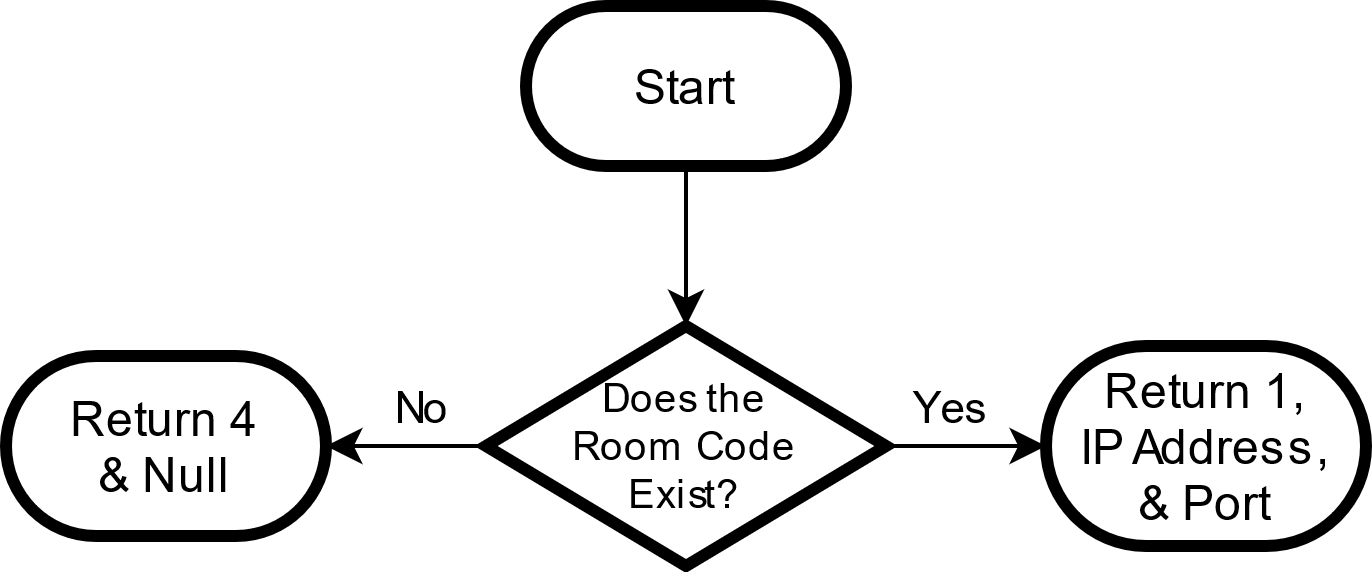
**Return False**

Else

**Return True**

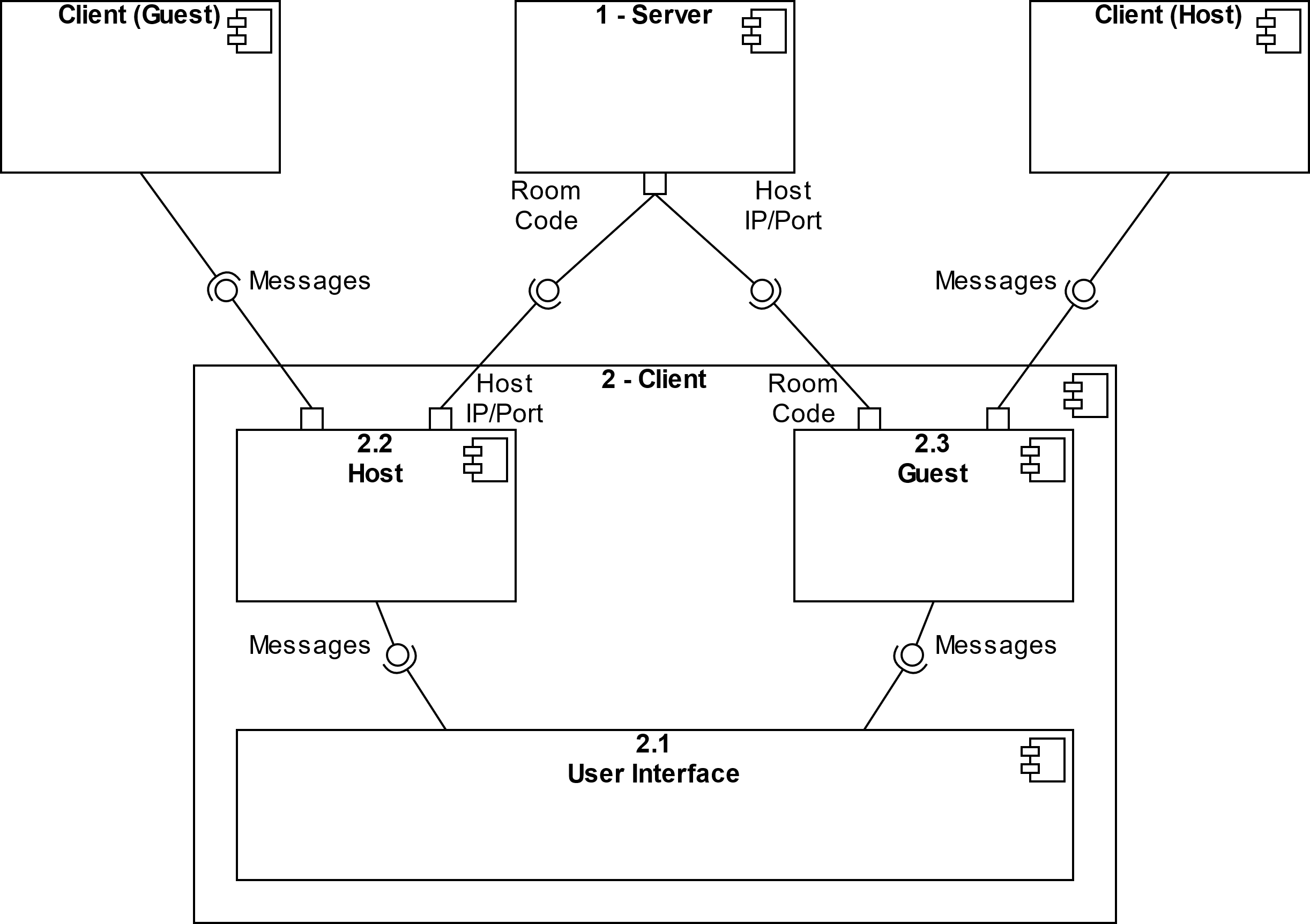
|  |  |
| --- | --- |
| *Validate Room Code* | |
| Description | Makes sure that an incoming room code is both valid and unique. It checks for the room code’s length (It must be 4 characters long), composition (it must only be comprised of letters), and uniqueness (It does not already exist within the map). If it fails any of these checks, a False is returned, and [Get Room Code](#_1.2.2_–_Get) must call [Generate Room Code](#_1.2.1_–_Generate_1) again. |
| Elements | |  |  | | --- | --- | | **Map**  [Server Storage](#_1.4_–_[Placeholder])’s main data structure. It maintains a map of unique room codes (keys) [HostEntries](#_1.4.1_–_[Placeholder]) (values). |  | |
| Referenced By | [1.4 – Server Storage](#_1.4_–_[Placeholder]) |
| Viewpoint | Pseudocode |

#### 1.4.7 – Get Network



|  |  |
| --- | --- |
| *Get Network* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

## 2 – Client



|  |  |
| --- | --- |
| *Client* | |
| Description | Connects with other Clients to create a one-to-many client-server architecture with the [Host](#_2.2_–_Host) acting as the server, and [Guests](#_2.3_–_Guest) as clients. Once a [Room Code](#_1.2.1_–_Generate_1) is obtained by the [Host](#_2.2_–_Host) user, it is distributed externally to other users, who then connect to the [Host](#_2.2_–_Host) via a persistent TCP connection using the [Host](#_2.2_–_Host)’s IP Address and Port Number obtained from contacting the [Server](#_1_–_Server) with a valid [Room Code](#_1.2.1_–_Generate_1).  Messages sent to the [Host](#_2.2_–_Host) from [Guests](#_2.3_–_Guest) or created by the [Host](#_2.2_–_Host) itself are distributed to all other [Guests](#_2.3_–_Guest). |
| Elements | |  |  | | --- | --- | | **2.1 – User Interface**  Handles user interaction with the System, including input and output. | **2.2 – Host**  Acts as a server for Guests once a persistent TCP connection has been made. Reroutes incoming messages to all other guests. | | **2.3 – Guest**  Clients that connect to the Host. Messages sent to the host are rerouted to all other guests. | **1 – Server**  Responsible for maintaining a map containing room codes (the key) and IP addresses/port numbers (the values), validating requests, and distributing room codes, IP addresses and port numbers to requesting clients. | | **Messages**  Strings sent between users via the Client. | **Room Code**  A somewhat random, 4-character, alphabetic room code. | |
| Referenced By | [0 – System View](#_0_–_System) |
| Viewpoint | UML 2.0 Component Diagram |

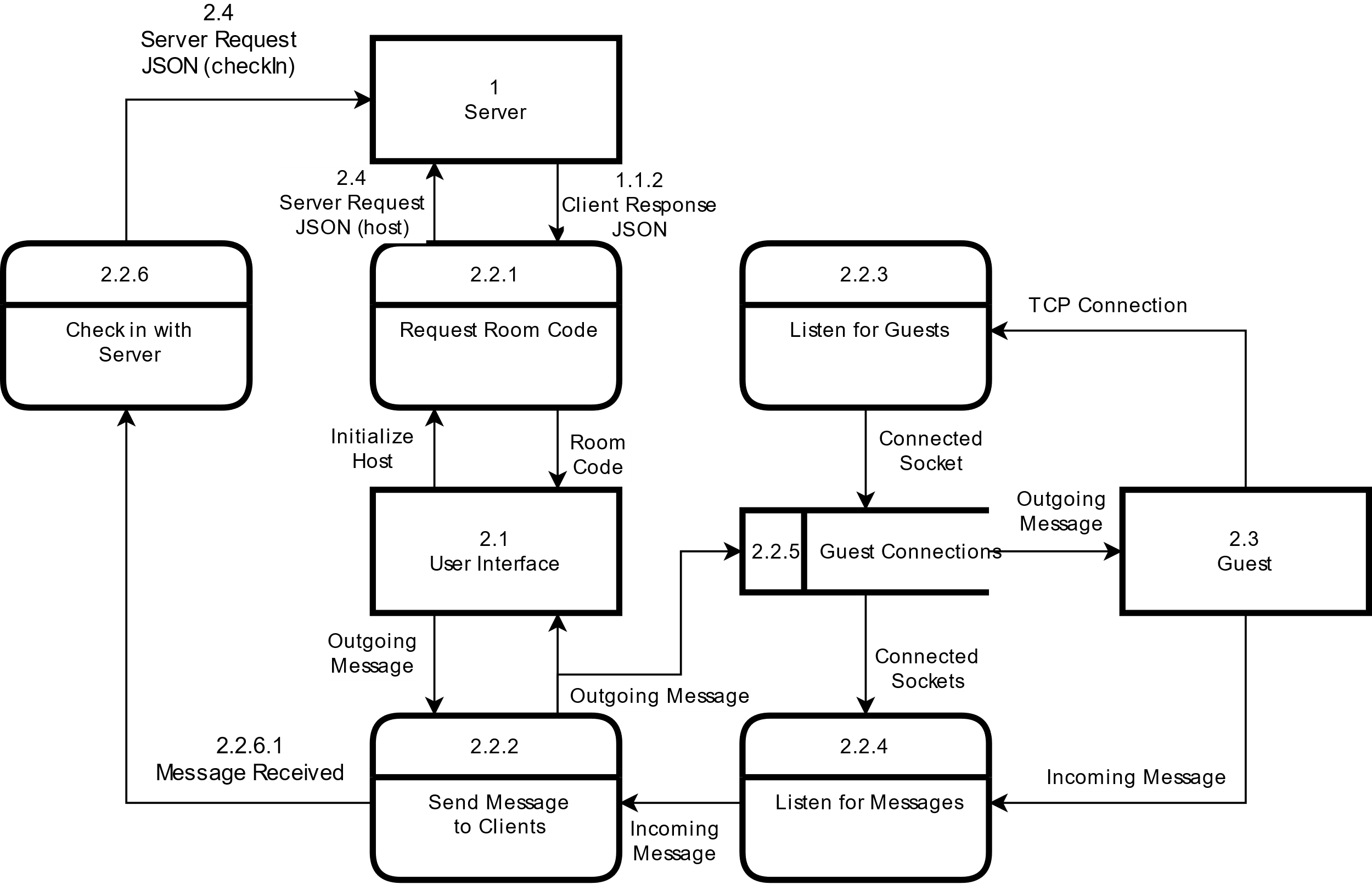
### 2.1 – User Interface

|  |  |
| --- | --- |
| *[View Name]* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

#### 2.1.1 – Chat Room

|  |  |
| --- | --- |
| *[View Name]* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

### 2.2 – Host



|  |  |
| --- | --- |
| *Host* | |
| Description | The Host is responsible for routing messages they receive to all of the [Guests](#_2.3_–_Guest) connected to it. When the Host is initialized, it calls [Request Room Code](#_2.2.1_–_[Placeholder]) to get a [Room Code](#_1.2.1_–_Generate_1) from the [Server](#_1_–_Server). The [Room Code](#_1.2.1_–_Generate_1) is distributed externally to other users. [Clients](#_2_–_Client) then connect to the Host using TCP via [Listen for Guests](#_2.2.3_–_[Placeholder]). Connected sockets are stored in [Guest Connections](#_2.2.5_–_[Placeholder]), and are used by [Listen for Messages](#_2.2.4_–_[Placeholder]) to receive incoming messages. [Send Message to Clients](#_2.2.2_–_[Placeholder]) sends the incoming message to all connected [Guests](#_2.3_–_Guest) (including the one who sent it) as well as to [User Interface](#_2.1_–_User). |
| Elements | |  |  | | --- | --- | | **1 – Server**  Responsible for maintaining a map containing [room codes](#_1.2.1_–_Generate_1) (the key) and IP addresses/port numbers (the values), validating [requests](#_2.4_–_Server), and distributing [room codes](#_1.2.1_–_Generate_1), IP addresses and port numbers to requesting [clients](#_2_–_Client). | **2.1 – User Interface**  Handles user interaction with the System, including input and output. | | **2.2.1 – Request Room Code**  Sends and receives a [request](#_2.4_–_Server) and [response](#_1.1.2_–_Client) for a [Room Code](#_1.2.1_–_Generate_1) from the server | **2.2.2 – Send Message to Clients**  Sends both incoming messages, as well as messages originating from [User Interface](#_2.1_–_User) to all connected [Guests](#_2.3_–_Guest) and the [User Interface](#_2.1_–_User). This ensures that messages are received by all [clients](#_2_–_Client) in the order that they are received by the Host. | | **2.2.3 – Listen for Guests**  Listens for incoming TCP connections from [guests](#_2.3_–_Guest), and transfers the resulting socket to [Guest Connections](#_2.2.5_–_[Placeholder]). | **2.2.4 – Listen for Messages**  Listens for messages sent by guests through the sockets connected by [Listen for Guests](#_2.2.3_–_[Placeholder]). | | **2.2.5 – Guest Connections**  An array that holds TCP sockets connected to [guests](#_2.3_–_Guest). A maximum of 8 other guests can be connected to the [host](#_2.2_–_Host). | **2.2.6 – Check in with Server**  Triggers [Room Check In](#_1.2.3_–_Room) within [Host Services](#_1.2_–_Host_1) within the [Server](#_1_–_Server). | | **2.3 – Guest**  [Clients](#_2_–_Client) that connect to the [host](#_2.2_–_Host). Messages sent to the [host](#_2.2_–_Host) are rerouted to all other guests. | **2.4 – Server Request JSON**  The JSON schema to be sent to the server to request services. | | **1.1.2 – Client Response JSON**  The JSON schema that will be returned to the [host](#_2.2_–_Host) from the [server](#_1_–_Server) in response to requests for service. | **2.2.6.1 – Message Received**  A method within [Check in with Server](#_2.2.6_–_[Placeholder]) that is called whenever [Send Message to Clients](#_2.2.2_–_[Placeholder]) has a new message to send. | | **Room Code**  A somewhat random, 4-character, alphabetic room code that is used to connect [Hosts](#_2.2_–_Host) and [Guests](#_2.3_–_Guest). (See [1.2.1 – Generate Room Code](#_1.2.1_–_Generate_1)) | **Initialize Host**  Sets the [client](#_2_–_Client) to a Host state. It receives a [room code](#_1.2.1_–_Generate_1) from the [server](#_1_–_Server) and opens the [chat room](#_2.1.1_–_Chat) for the host where it waits for [guests](#_2.3_–_Guest) to connect to it. | |
| Referenced By | [2 – Client](#_2_–_Client), [2.3 – Guest](#_2.3_–_Guest) |
| Viewpoint | Dataflow Diagram |

#### 2.2.1 – [Placeholder] Request Room Code

|  |  |
| --- | --- |
| *[View Name]* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

#### 2.2.2 – [Placeholder] Send Message to Clients

|  |  |
| --- | --- |
| *[View Name]* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

#### 2.2.3 – [Placeholder] Listen for Guests

|  |  |
| --- | --- |
| *[View Name]* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

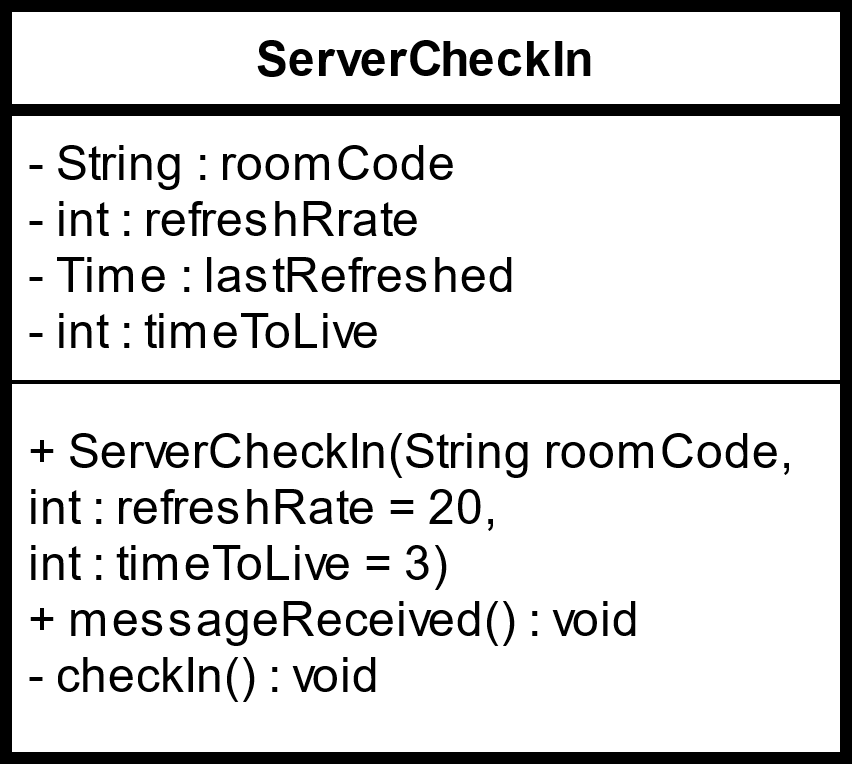
#### 2.2.4 – [Placeholder] Listen for Messages

|  |  |
| --- | --- |
| *[View Name]* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

#### 2.2.5 – [Placeholder] Guest Connections

|  |  |
| --- | --- |
| *[View Name]* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

#### 2.2.6 – Check in with Server



|  |  |
| --- | --- |
| *Check in with Server* | |
| Description | Whenever the [Send Message to Clients](#_2.2.2_–_[Placeholder]) receives a new message, it calls [Message Received](#_2.2.6.1_–_Message). If certain conditions are met (see [2.2.6.1 – Message Received](#_2.2.6.1_–_Message)) checkIn() is called, which sends a [refresh-type request](#_2.4_–_Server) to the [server](#_1_–_Server). Time : lastRefreshed is the hour, minute and second that the last successful [refresh request](#_2.4_–_Server) was made, and timeToLive refers to the number of tries ServerCheckIn has to make a successful [refresh request](#_2.4_–_Server) to the [server](#_1_–_Server). |
| Elements | |  |  | | --- | --- | | **String : roomCode**  A somewhat random, 4-character, alphabetic room code. | **int : refreshRate**  The frequency, in minutes, for allowing a new [refresh request](#_2.4_–_Server) to the [server](#_2.4_–_Server). The default is every 20 minutes. | | **Time : lastRefreshed**  The hour, minute and second that the last successful [refresh request](#_2.4_–_Server) was made. | **int : timeToLive**  The number of tries ServerCheckIn has to make a successful [refresh request](#_2.4_–_Server) to the [server](#_1_–_Server). | | **ServerCheckIn (Constructor)**  The constructor for the class. It is passed the [host’s](#_2.2_–_Host) [room code](#_1.2.1_–_Generate_1), and, optionally, a custom refreshRate and timeToLive. | **2.2.6.1 – Message Received**  This method is called whenever [Send Message to Clients](#_2.2.2_–_[Placeholder]) sends a message. It determines whether a [refresh request](#_2.4_–_Server) should be made to the [server](#_1_–_Server) or not. | | **checkIn()**  The method that handles sending [refresh requests](#_2.4_–_Server) to the [server](#_1_–_Server). |  | |
| Referenced By | [2.2 – Host](#_2.2_–_Host) |
| Viewpoint | UML Class Diagram |

##### 2.2.6.1 – [Placeholder] Message Received

|  |  |
| --- | --- |
| *Message Received* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

### 2.3 – Guest

|  |  |
| --- | --- |
| *[View Name]* | |
| Description |  |
| Elements | |  |  | | --- | --- | | **[Element 1]** | **[Element 2]** | | **[Element 3]** | **[Element 4]** | |
| Referenced By |  |
| Viewpoint |  |

### 2.4 – Server Request JSON

{

“serverRequest”: [

“type” : ”*type*”,

“time” : *“currentTime”*,

“date” : *“mm/dd/yyyy”*,

// used for a request from a host

“networkingInfo” : [

“IP” : *“IPAddress”,*

“Port” : “*portNumber”*

],

// used for a request from a guest and checkIn

“roomCode” : “*room code”*

]}

|  |  |
| --- | --- |
| *Server Request JSON* | |
| Description | JSON format for a request to the server. It is used by both [Guests](#_2.3_–_Guest) and [Hosts](#_2.2_–_Host) to pull information from the [Server](#_1_–_Server). |
| Elements | |  |  | | --- | --- | | **type**  The type of request being made to the server ([host](#_2.2_–_Host), [guest](#_2.3_–_Guest), or [refresh](#_2.2.6_–_[Placeholder])) | **networkingInfo**  An array that holds the IP Address and the Port Number | | **Room Code**  A somewhat random, 4-character, alphabetic room code that is used to connect [Hosts](#_2.2_–_Host) and [Guests](#_2.3_–_Guest). (See [1.2.1 – Generate Room Code](#_1.2.1_–_Generate_1)) |  | |
| Referenced By | [2 – Client](#_2_–_Client), [1.1.2 – Client Response JSON](#_1.1.2_–_Client) |
| Viewpoint | JSON |

### 2.5 – Inter-Client Message JSON

{

“message” : “*message*”

}

|  |  |
| --- | --- |
| *Inter-Client Message JSON* | |
| Description | How messages are sent between [Clients](#_2_–_Client). [Guests](#_2.3_–_Guest) only send their messages to the [Host](#_2.2_–_Host), which then echoes that message to all of the [Guests](#_2.3_–_Guest) connected to it. [Hosts](#_2.2_–_Host) that create a message send that message to all other [Guests](#_2.3_–_Guest) as well. No extra information is needed. |
| Elements | |  |  | | --- | --- | | **message**  A string that is sent to either the [Host](#_2.2_–_Host) or [Clients](#_2_–_Client) connected to the [Host](#_2.2_–_Host). |  | |
| Referenced By | [2 – Client](#_2_–_Client) |
| Viewpoint | JSON |