**­California State University, Sacramento**



**Team #7 / Team Blue**

**- Project Deliverable #2 -**

**Requests and Design Documentation**

**CSC 131 – 02**

**Computer Software Engineering**

**Fall 2022**

**Software Req and Design Brief –V1.1**

Table of Contents

[Introduction 4](#_Toc122007158)

[**Purpose of this Doc** 4](#_Toc122007159)

[Project Scope 4](#_Toc122007160)

[**Overview** 4](#_Toc122007161)

[**System Context Diagram** 4](#_Toc122007162)

[Object Oriented Requirements Analysis (OOA) 4](#_Toc122007163)

[**Overview** 4](#_Toc122007164)

[**Use Cases** 5](#_Toc122007165)

[Use Case Descriptions 5](#_Toc122007166)

[System Requirements 9](#_Toc122007167)

[**Overview** 9](#_Toc122007168)

[**Functional Requirements** 10](#_Toc122007169)

[Overview 10](#_Toc122007170)

[Requirements 10](#_Toc122007171)

[**Non-Functional Requirements** 10](#_Toc122007172)

[Overview 10](#_Toc122007173)

[Requirements 10](#_Toc122007174)

[Data Design 10](#_Toc122007175)

[**Overview** 10](#_Toc122007176)

[**Entity Relationship Diagram (ERD)** 11](#_Toc122007177)

[Overview 11](#_Toc122007178)

[Architectural Design 11](#_Toc122007179)

[**Overview** 11](#_Toc122007180)

[**Frontend Class Diagram** 12](#_Toc122007181)

[**Backend Class Diagram** 13](#_Toc122007182)

[Detailed Design 13](#_Toc122007183)

[**Overview** 13](#_Toc122007184)

[**Frontend Sequence Diagram** 14](#_Toc122007185)

[**Backend Sequence Diagram** 14](#_Toc122007186)

[User Interface Design 14](#_Toc122007187)

[**Overview** 14](#_Toc122007188)

[**Frontend User Interface** 15](#_Toc122007189)

[**Backend User Interface** 16](#_Toc122007190)

[Technology and Tools 18](#_Toc122007191)

[**Overview** 18](#_Toc122007192)

[**Frontend:** 18](#_Toc122007193)

[**Backend:** 19](#_Toc122007194)

[**Version Control:** 19](#_Toc122007195)

[Assumptions and Constraints 19](#_Toc122007196)

[**Overview** 19](#_Toc122007197)

# Introduction

## **Purpose of this Doc**

This request and design document provides our client Dmitry Shikhaleev, part of the Five9 team, a description of the functionality and requirements in correlation with the Agent Call Service Application.

As part of the CSC 131 course taught by Dr. Ahmed Salem, we have been tasked with designing and implementing a reliable and efficient software application using sound engineering principles. This project presents an opportunity for us to demonstrate our ability to create effective diagrams and designs, as well as our understanding of software interaction specifications and the importance of teamwork. In addition to meeting the requirements outlined in this document, our team is committed to delivering a cutting-edge solution that leverages the latest technology and industry best practices.

This semester, our team has been presented with a unique opportunity to create a software application for our esteemed client. The proposed application will provide a supervisor or other relevant individual with a comprehensive, real-time visual representation of their entity population in a user-friendly web-based interface. This innovative solution will enable them to gain valuable insights into their organization, allowing them to make data-driven decisions that drive business success. Our team is committed to delivering exceptional service and support throughout the development process to ensure that our client's needs are met and exceeded.

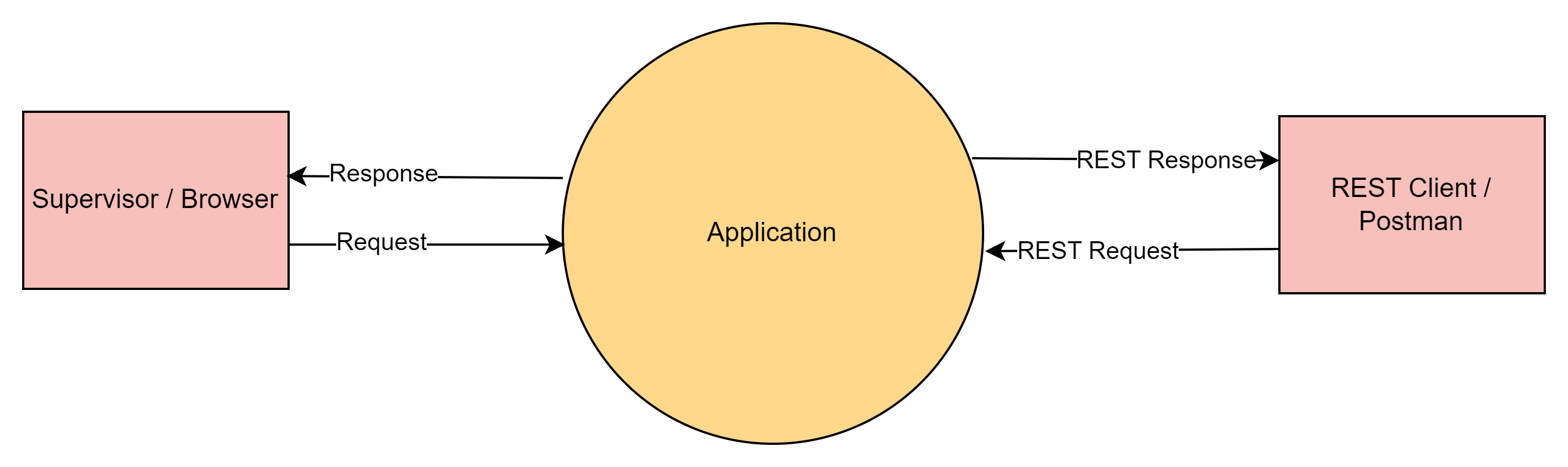
# Project Scope

## **Overview**

* Design and develop a web page that provides meaningful, relevant, and functional user experience. (The “Frontend”.)
* Develop a backend application that manages REST API requests and handles concurrency in a reliable manner. (The “Backend”.)

## **System Context Diagram**

* System context diagram provides a brief, high-level overview of our application's interface and the external actors intended to interact with our system.

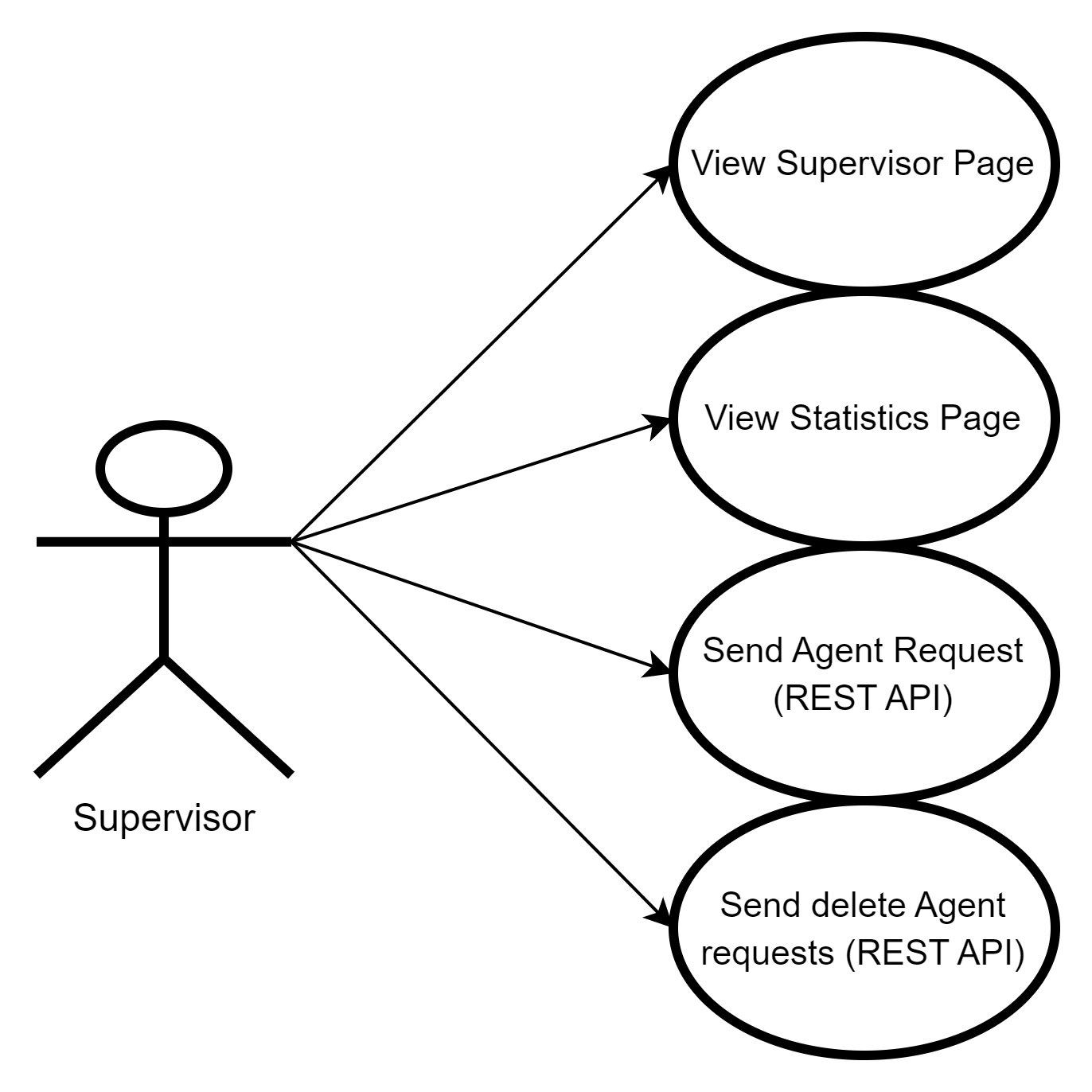


# Object Oriented Requirements Analysis (OOA)

## **Overview**

* In this document, we utilize the powerful Unified Modeling Language (UML) to precisely specify, visualize, construct, and document the critical objects of our product. UML provides a standardized and efficient framework for modeling complex systems and is an invaluable tool for communicating the design of our software to stakeholders and team members.

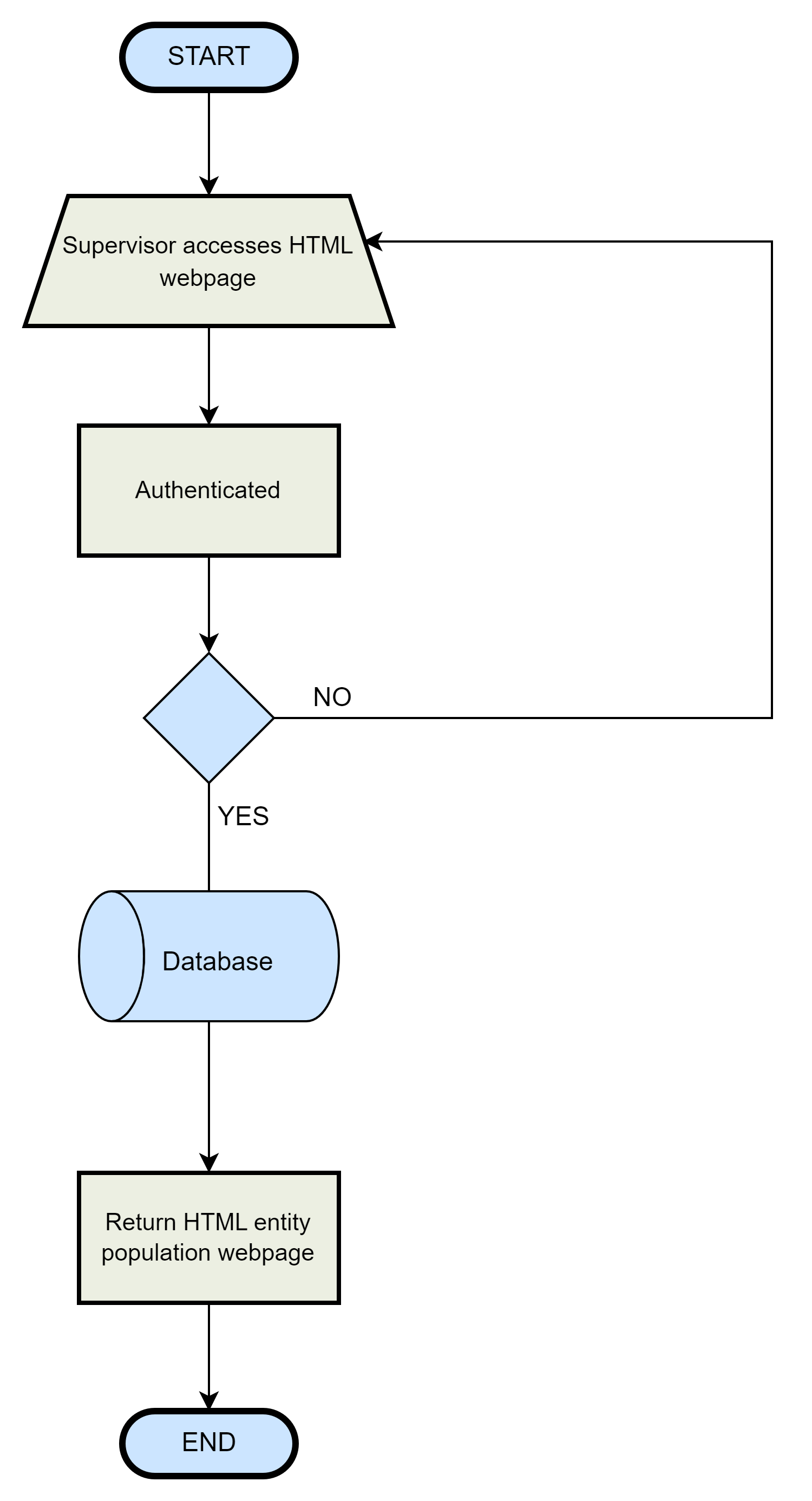
## **Use Cases**



### Use Case Descriptions

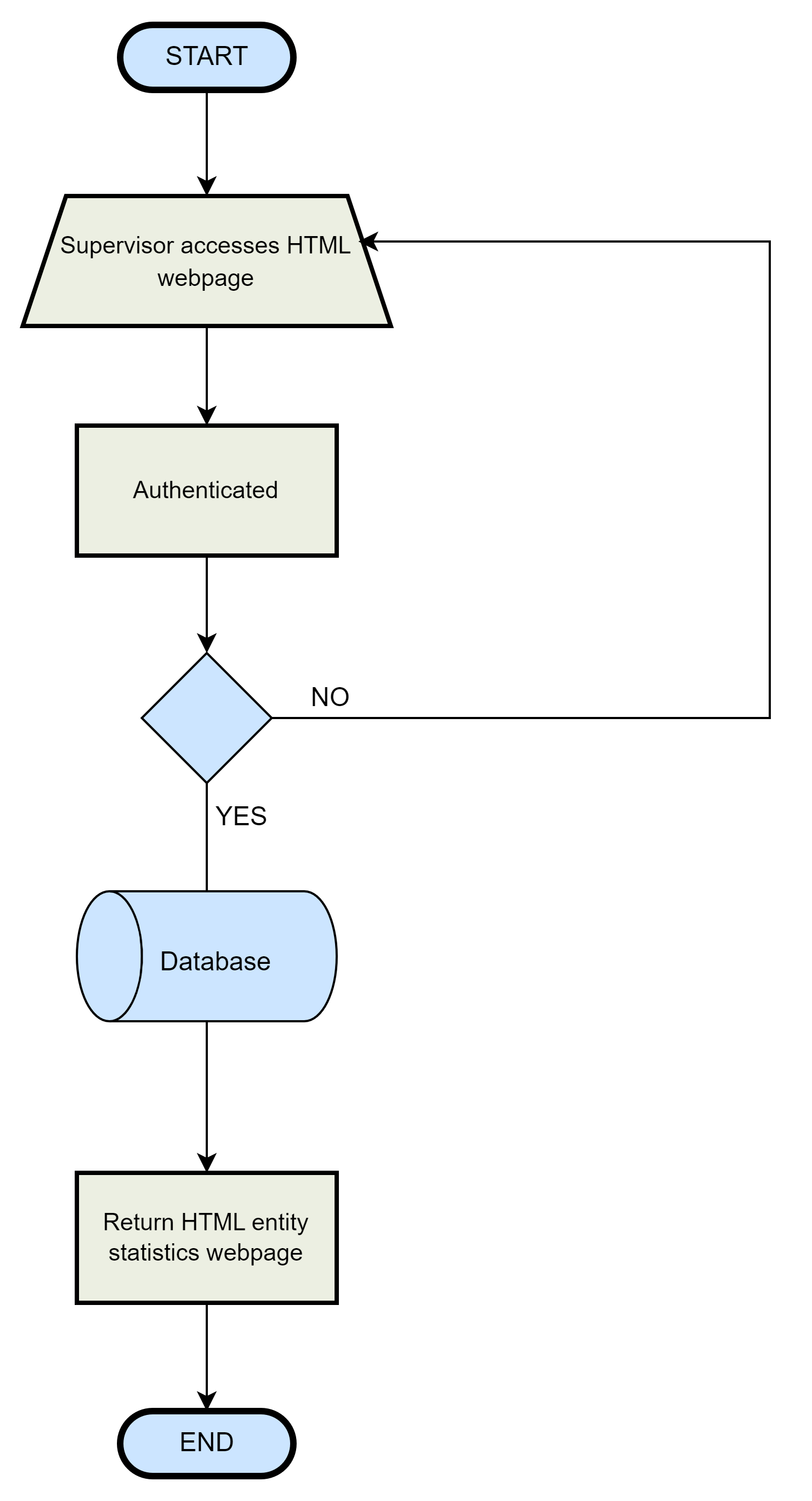
#### View Supervisor Page

* + **Actor(s):** Supervisor (Browser)
  + **Author(s):** Fernando Mendoza
  + **Overview:** This use case captures the process of viewing and deriving an overall sense of the entity population, or in other words the status of the agents of a call service agency.
  + **1.1: Typical Flow Description**



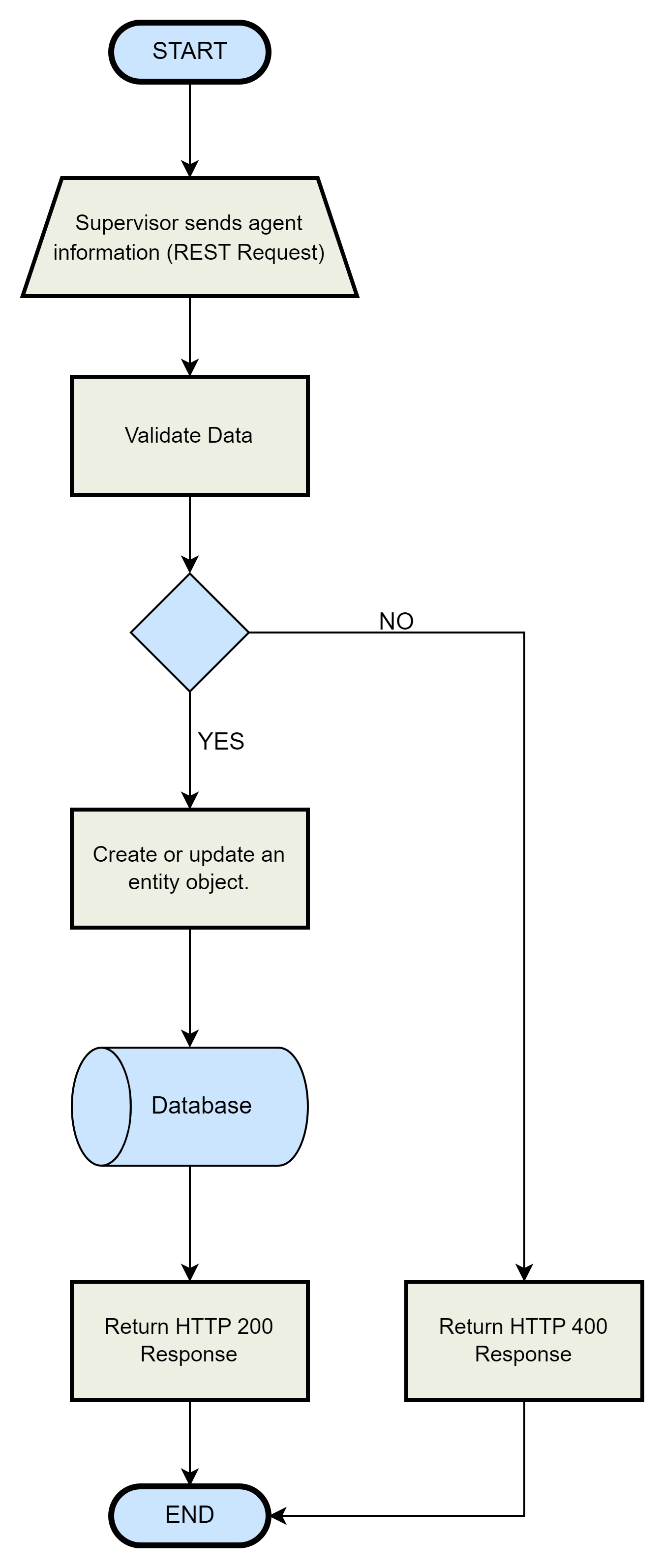
#### View Statistics Page

* + **Actor(s):** Supervisor (Browser)
  + **Author(s):** Fernando Mendoza
  + **Overview:** This use case captures the process of viewing and deriving an overall sense of the entity population, or in other words the status of the agents of a call service agency. This gives a more precise data view of the entity’s current states or statuses.
  + **2.1: Typical Flow Description**



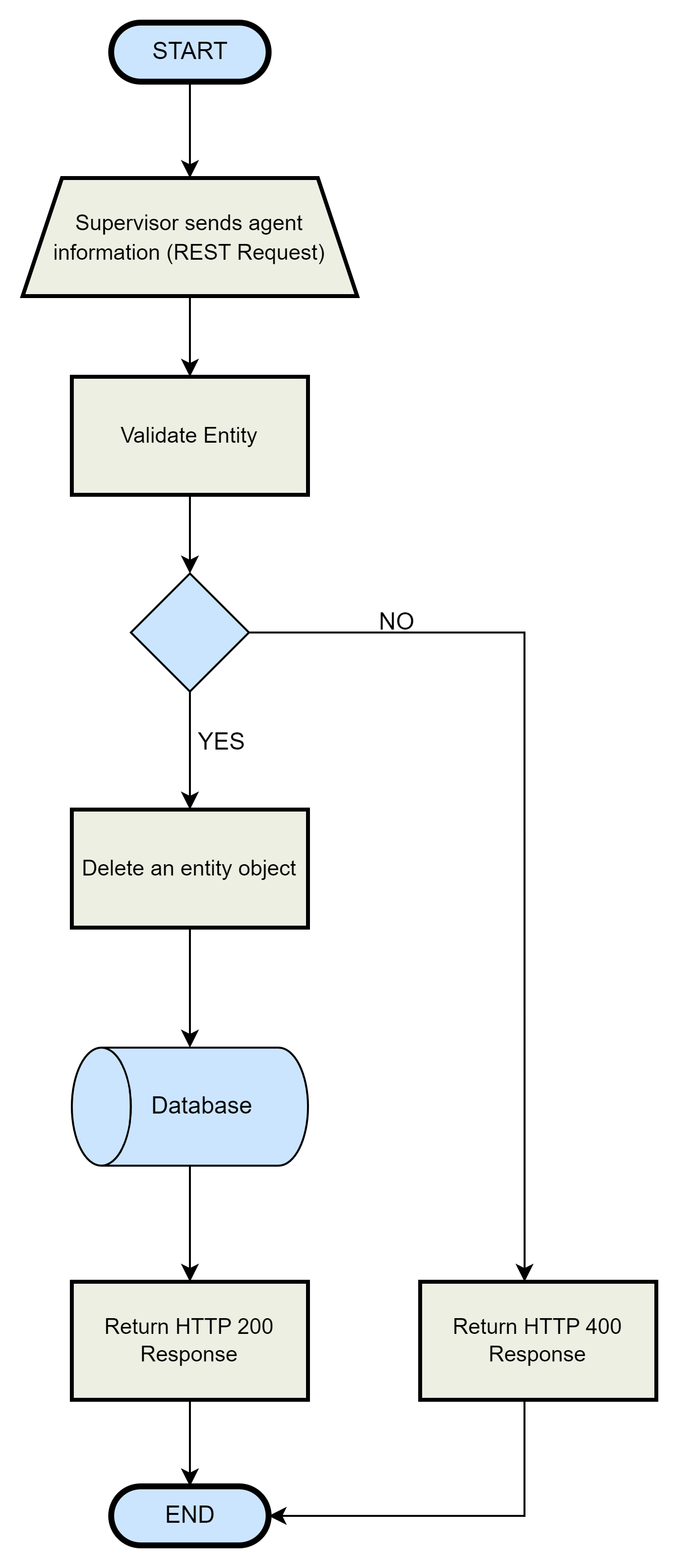
#### Send Agent Request (REST API)

* + - * **Actor(s):** Supervisor or RESTful client (Postman / Curl Client)
      * **Author(s):** Fernando Mendoza
      * **Overview:** This use case captures the process of sending HTTP REST API requests to an application and rendering real time events to an application.
      * **3.1: Typical Flow Description**



#### Send Delete Agent Request (REST API)

* + **Actor(s):** Supervisor or RESTful client (Postman / Curl Client)
  + **Author(s):** Fernando Mendoza
  + **Overview:** This use case captures the process of sending HTTP REST API requests to an application and deleting entities associated with an id from a database and webpage.
  + **4.1: Typical Flow Description**



# System Requirements

## **Overview**

* The following section provides a brief overview of the function and performance of our system and its constraints. It presents the reader with a comprehensive understanding of what our software will do. This overview is intended to give stakeholders and team members a clear picture of the capabilities and potential of our system, as well as any limitations or considerations that must be taken into account during the development process.

## **Functional Requirements**

### Overview

* + - The following sections provides the reader a view of the software’s/system’ functionality and the behaviors it has.

### Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Requirement Statement** | **Must/Want** | **Comments** |
| FR001 | System must accept requests (REST API) | Must |  |
| FR002 | System must display real time changes | Must |  |
| FR003 | System must have authentication | Must |  |
| FR004 | System must relay the entity population information via an HTTP / REST API interface. | Must |  |

## **Non-Functional Requirements**

### Overview

* + - The following section provides the reader a view of the expectations of our system; the desired properties of our system.

### Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Requirement Statement** | **Must/Want** | **Comments** |
| NFR001 | System must be fast | Want | Be able to handle up to 1600 requests a second. Our system can handle simultaneous requests and render changes to a web page. |
| NFR002 | System must be secure | Want |  |
| NFR003 | System must be reliable | Want |  |
| NFR004 | System experience (“Frontend”) must be easy to use | Want |  |
| NFR005 | System must render real time changes within a 1 second response time | Want | Our system renders changes with an average time of 5s. |

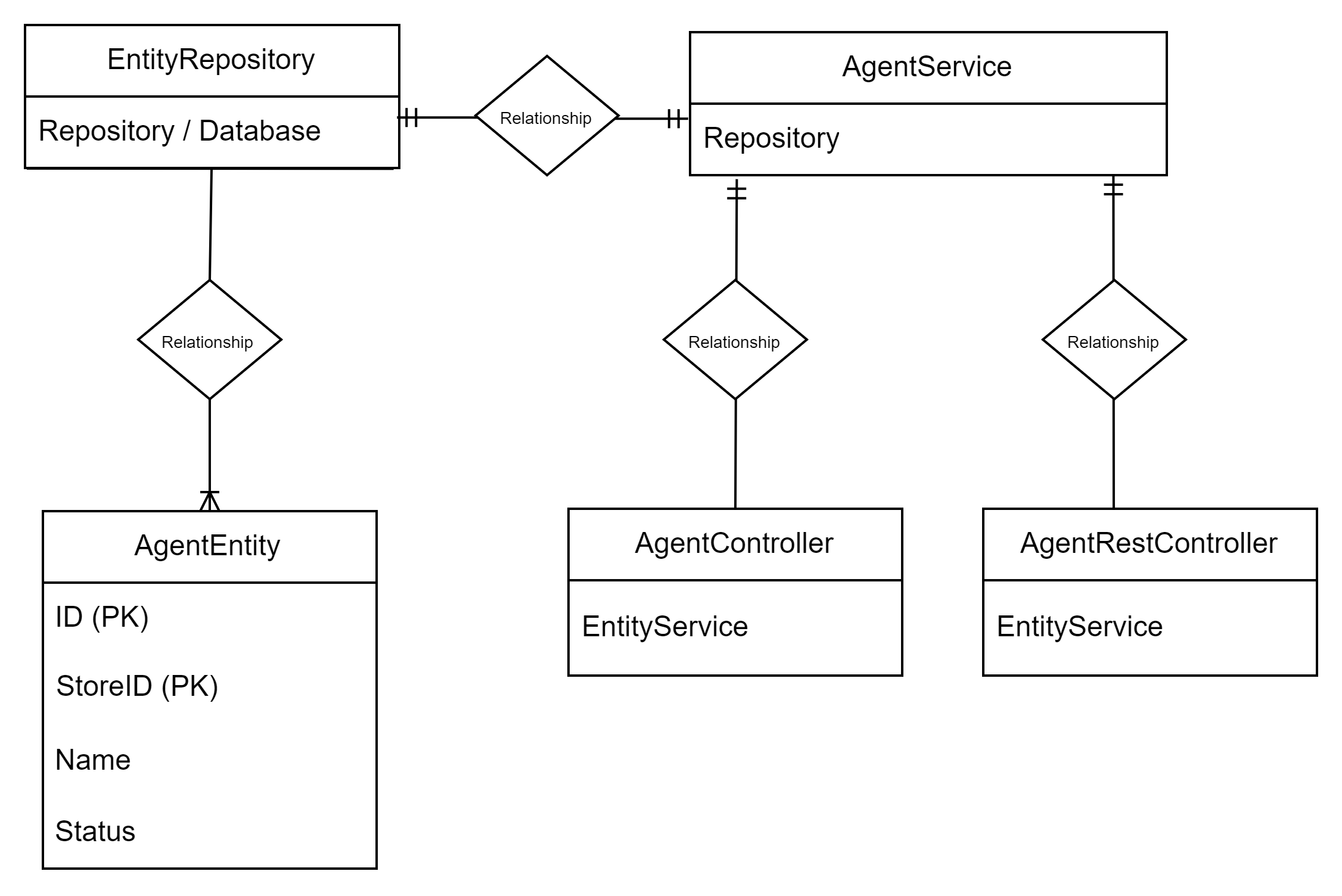
# Data Design

## **Overview**

* The following section provides the reader with a detailed snapshot of the advanced data structures that exhibit the dynamic relationship between entities and our system. This comprehensive analysis showcases the complex structures that reside within our software, as well as their associated attributes and interdependencies. This detailed analysis will give stakeholders and team members a clear understanding of the intricate structures that reside within our software, as well as the related attributes that enable these structures to function at the highest level of efficiency and performance.

## **Entity Relationship Diagram (ERD)**

### Overview

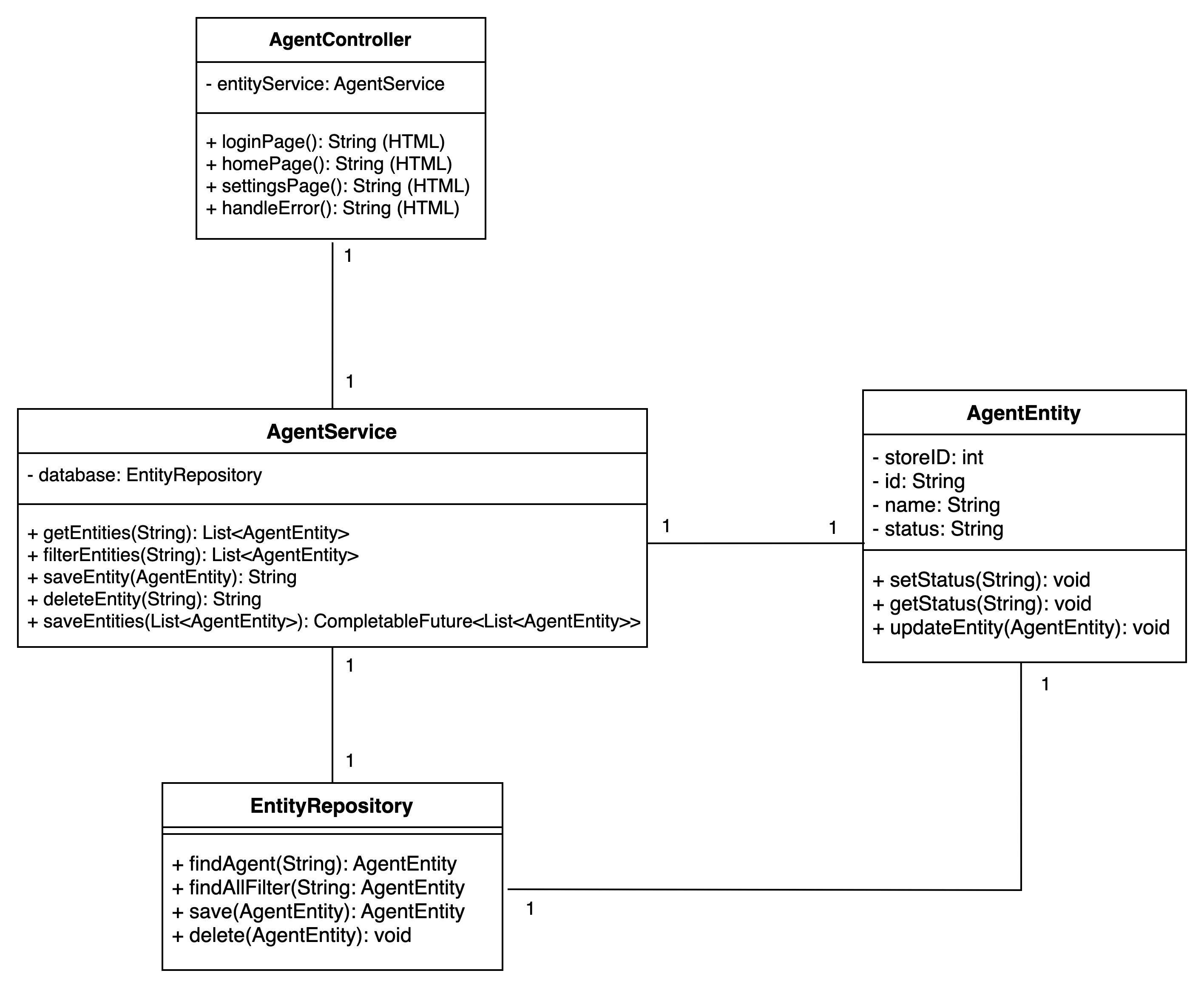


# Architectural Design

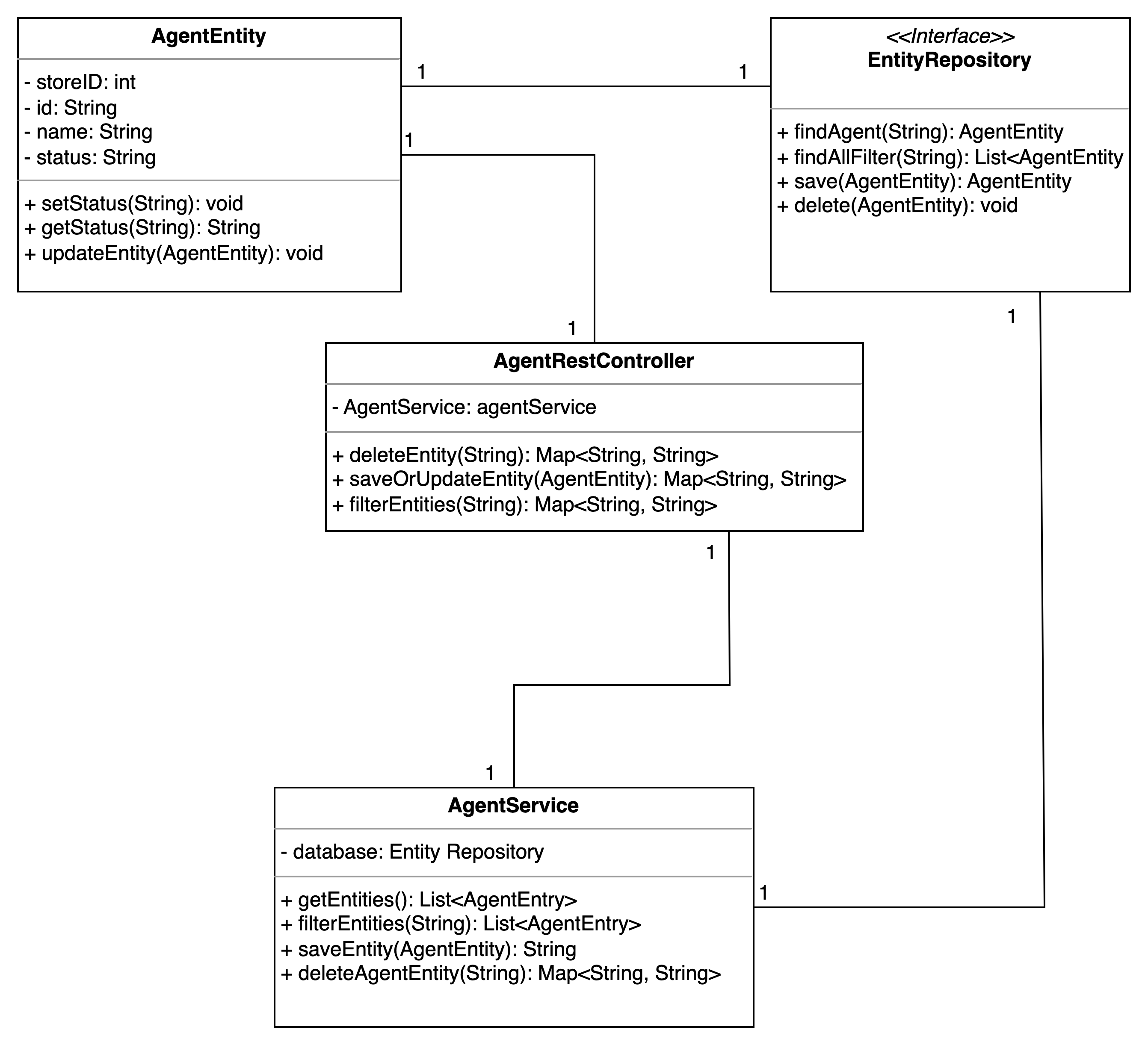
## **Overview**

* The following section provides the reader with a comprehensive outline of the system / software architecture, including the major components and their interconnections. This serves as a bridge between the technical details and the big-picture understanding of the system.

## **Frontend Class Diagram**



## **Backend Class Diagram**

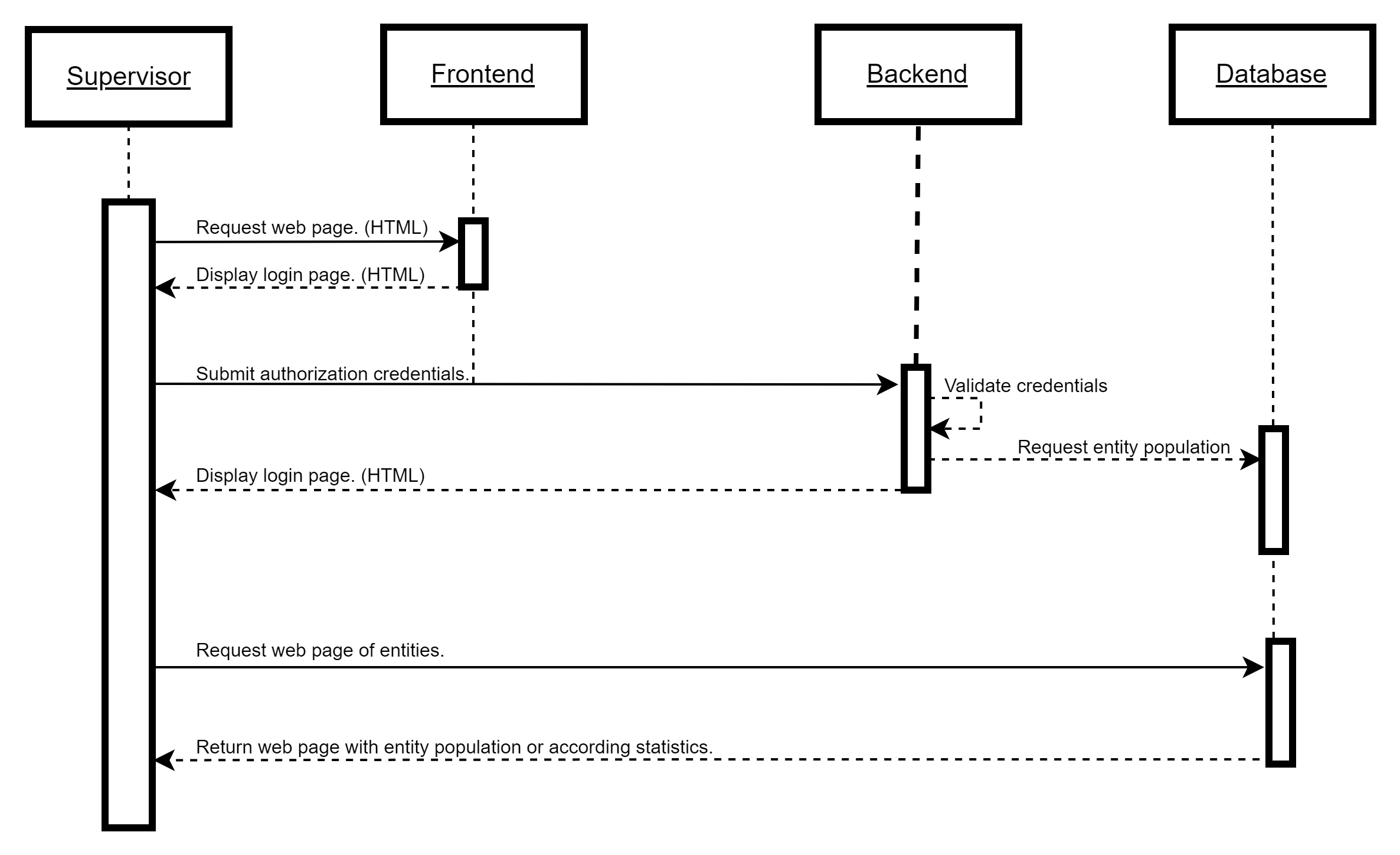


# Detailed Design

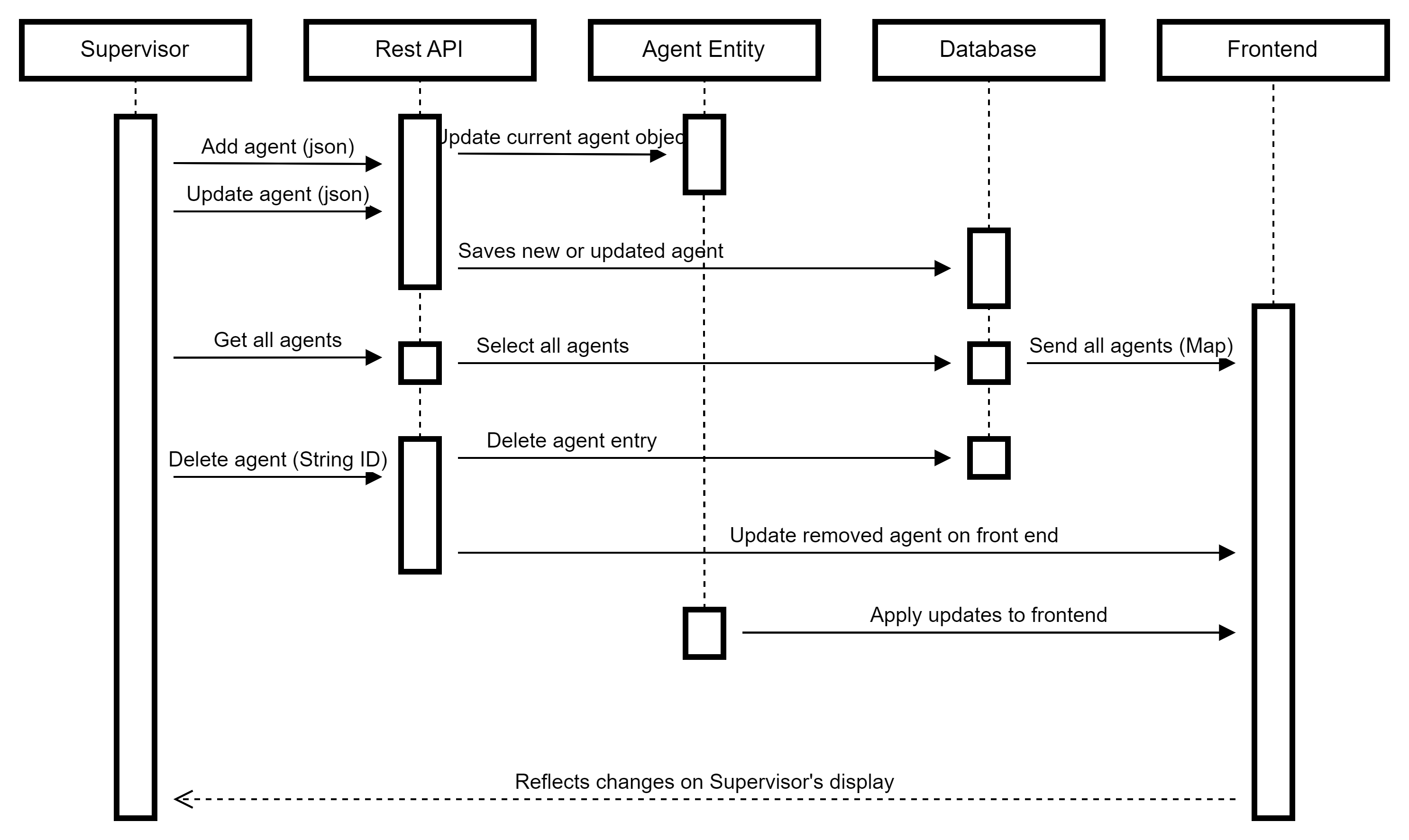
## **Overview**

* The following section provides the reader with a comprehensive overview of the dynamic interactions between objects in our system over time. This analysis utilizes advanced visual representation techniques to clearly and concisely depict the flow of messages or events between objects in our system, enabling stakeholders and team members to gain valuable insights into the behavior and performance of our system. By leveraging the full capabilities of these powerful visualizations, we can ensure the success and scalability of our system.

## **Frontend Sequence Diagram**



## **Backend Sequence Diagram**



# User Interface Design

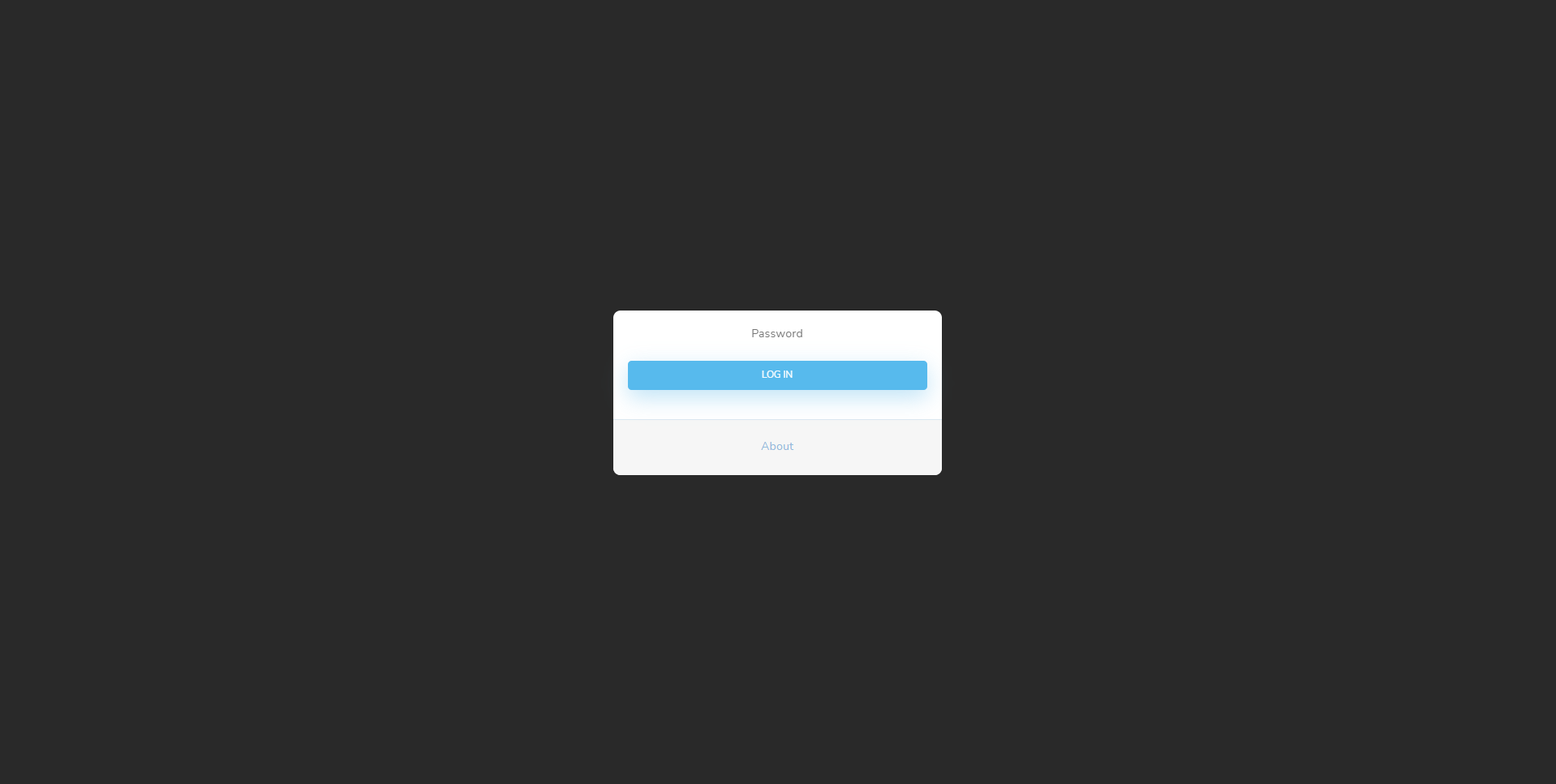
## **Overview**

* The following section provides the reader with a comprehensive overview of the interactions that will take place with our system. It describes the appropriate methods of interacting with the system and the corresponding responses that will occur as a result.

## **Frontend User Interface**

#### Login Page

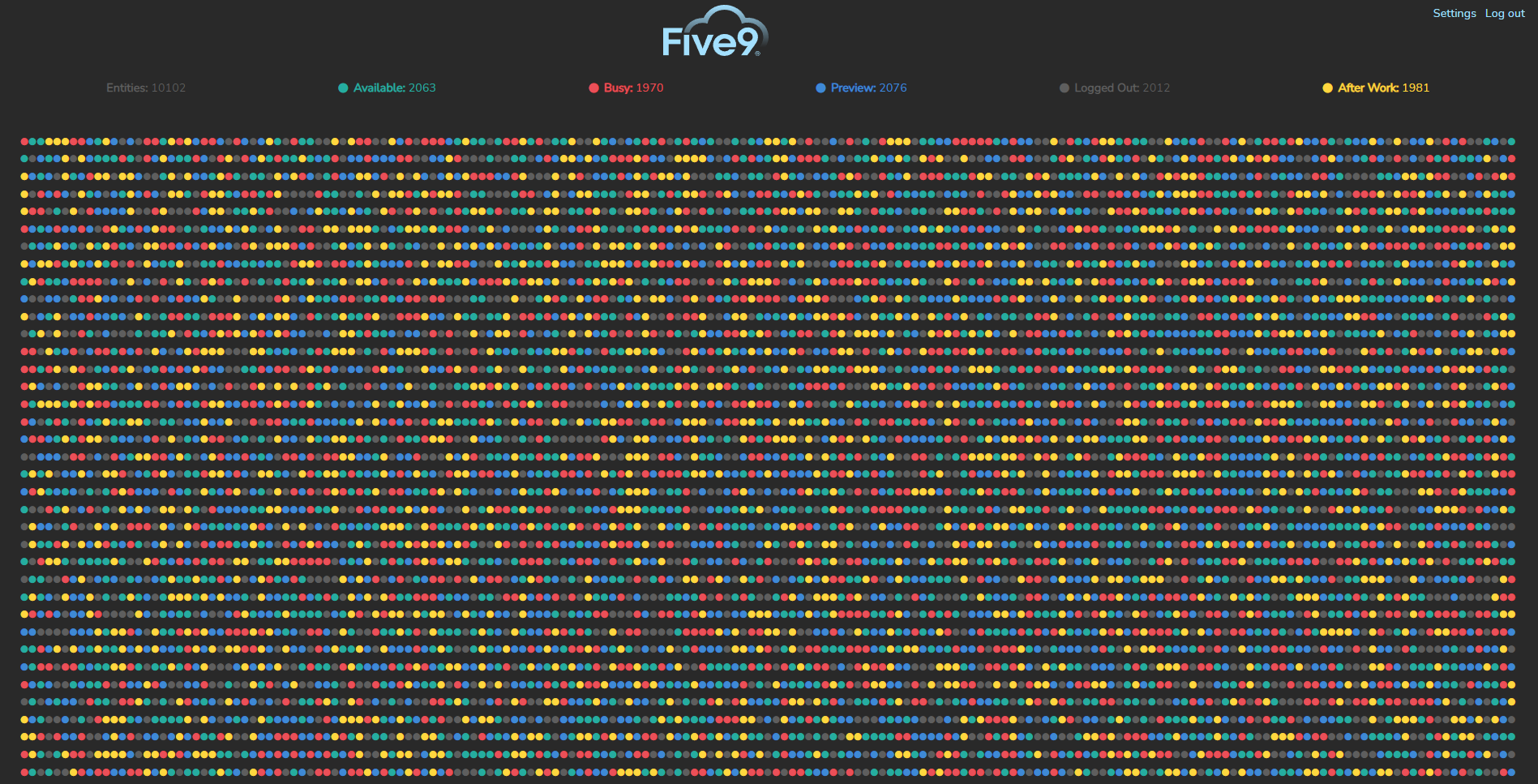
* + When the user first accesses the application in a web browser, if the route is appropriate, the user is presented with a user-friendly login page. This intuitive and seamless user experience enables users to access the application and begin leveraging its powerful functionality easily and securely.



* + - * The user can login and view the supervisor page if valid credentials are provided.

#### Home Page

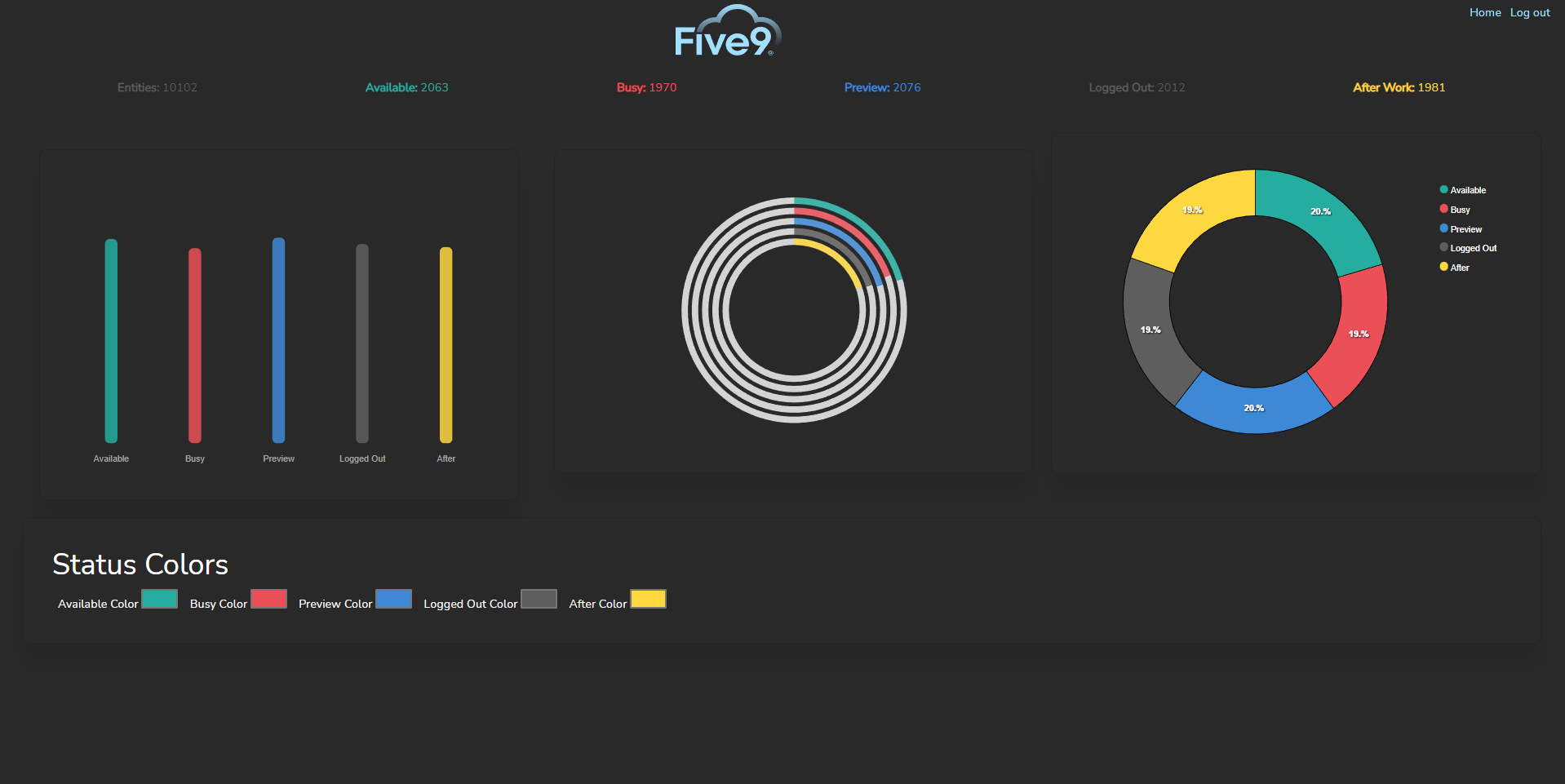
* + When the user provides valid credentials, the frontend will respond with a dynamic HTML response displaying the comprehensive entity population and a user-friendly settings/stats page option..



* + - The user / supervisor can view and derive an overall sense of the entity population. The supervisor shall be able to derive an overall sense of of state of their agents or the statuses of each entity.

#### Stats Page

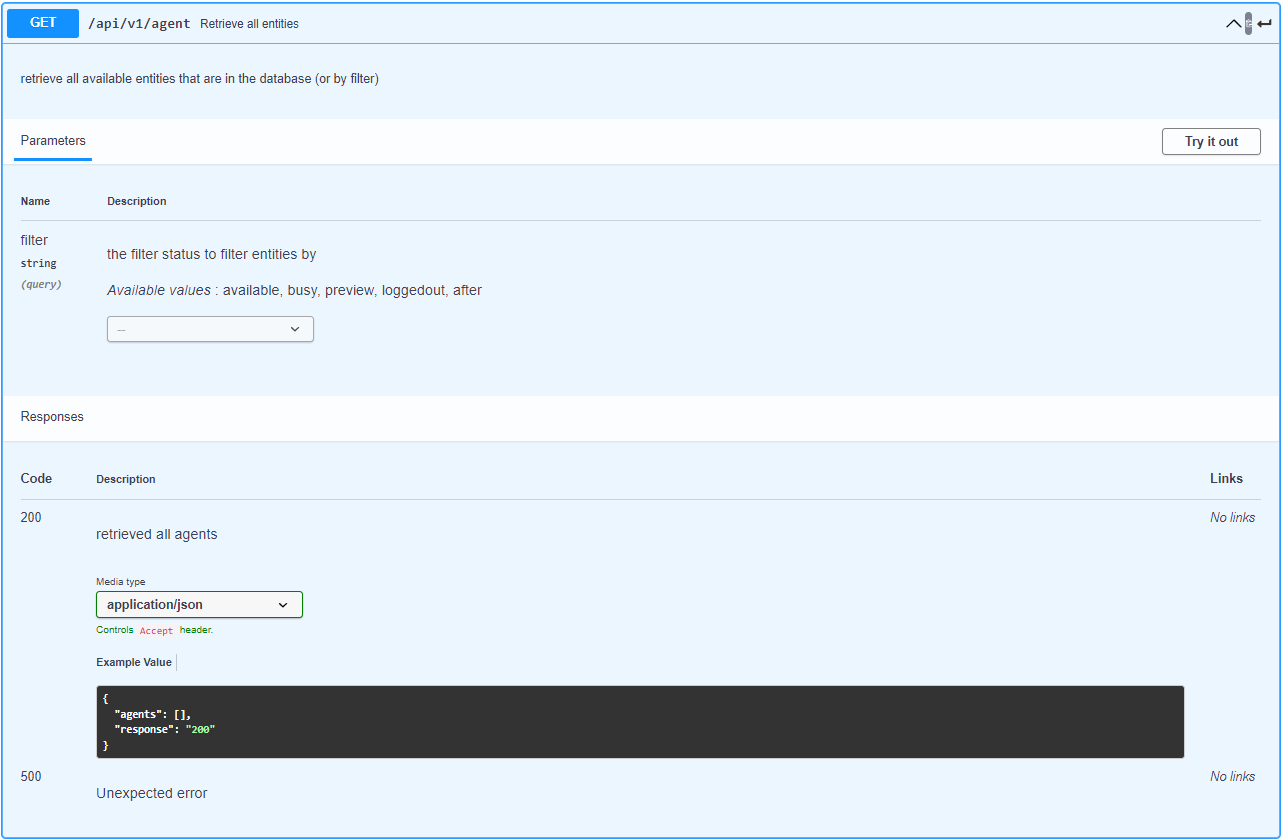
* + When the user accesses the stats route, the backend will respond with a HTML response displaying the entity populations statistics in relation to the statuses of said population. The user is also presented with options to customize the status color or the HTML webpage to their preference.



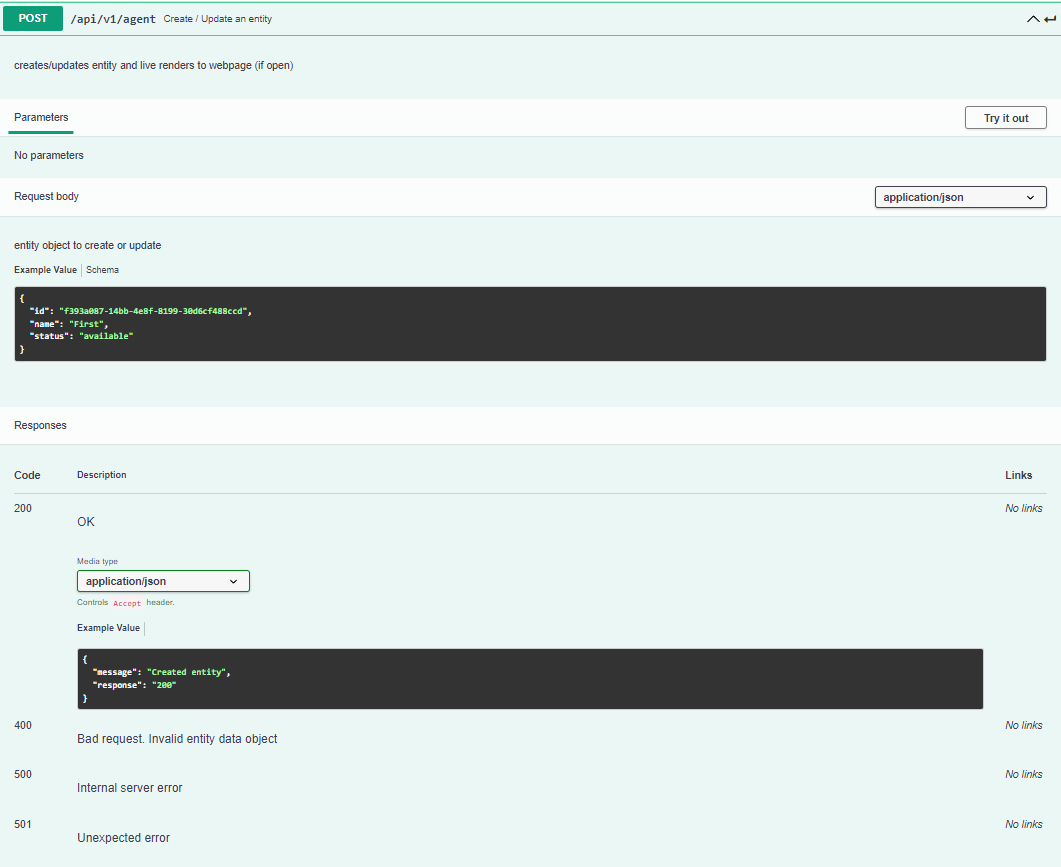
* + - The user / supervisor can view the status information's categories and gain a real-time, actionable insight into the entity's current state.

## **Backend User Interface**

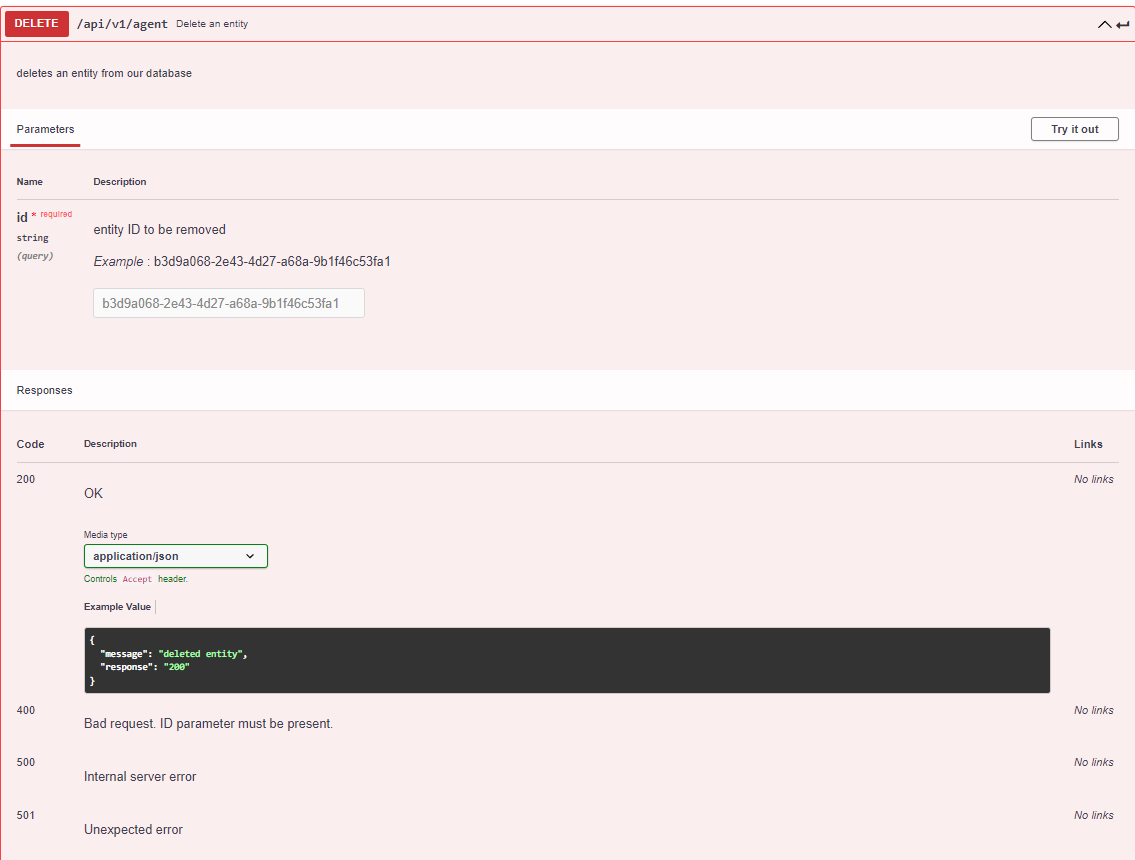
#### Get All Entities



#### Create or Update Entity



#### Delete Entity



#### Extra Information

* API Documentation: <https://app.swaggerhub.com/apis/J_2/Call-Service-Team-Blue/1.0.0>
* API Dynamic Documentation: 

# Technology and Tools

## **Overview**

The following section provides the reader with a more in-depth overview of the advanced technology stack that was incorporated to build our system/software. This comprehensive breakdown details the various technologies and frameworks that were used to develop the application.

## **Frontend:**

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework/Ext.** | **Benefits** |
| HTML5 | * Thymeleaf | * High-performance parsed template cache * Multi-modal template support (i.e. XML, XHTML, HTML5) * Highly developer-friendly |
| CSS3 | * Bootstrap | * Powerful, extensible, and packed with features * Cross browser independent * Highly developer-friendly |
| JavaScript | * jQuery | * Multi-browser support * Code compression/simplification * Highly developer-friendly |

## **Backend:**

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework/Ext.** | **Benefits** |
| Java | * Spring Web MVC * Spring Security * Spring Boot | * Lightweight/flexible development * Task simplification and boilerplate code reduction * Easy integration with other technologies |
| MySQL | * Java Persistence API | * Provides standard method to access/manage RDBMS * Simplifies data access and persistence * Enables developers to easily implement data-driven features |

## **Version Control:**

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework/Ext.** | **Benefits** |
| GitHub | * GitHub Desktop | * Seamless project management/collaboration * Highly documented code commits/changes * Simple, open-source distribution interface |

# Assumptions and Constraints

## **Overview**

* The Frontend must visually contain all the entities with no scrolling. The frontend must render real time web changes however ours is slightly delayed.
* Our backend should handle 1600 simultaneous requests, ours should at minimum be able to handle 400 requests a second and render them accordingly. The backend should also be REST API based.