Software Requirement Specification

Louw, Matthew Jason
u16217498

Bresler, Mathilda Anna
u16313382

Braak, Pieter Albert
u16009917

Reva, Kateryna u17035989

Li, Xiao Jian
u16099860

April 2019

1 Introduction

1.1 Purpose

This document is written with the COS 301 lectures, the CS department, and EPI use as clients in mind. This SRS will provide all parties concerned with details concerning the Real time fire escape routes project. The SRS contains information relevant to the current scope of the project, and is thus not a complete representation of the final project since there are expected to be changes to the system as the project takes shape. In this document we hope to convey our current understanding of the project and its implementation to the previously mentioned clients. This will guide the developers and the clients to obtain a mutual understanding of what is expect from the system.

1.2 Scope

The Real-time Fire Escape Route (Real-time FER) System is a new approach to solve an age old problem. The interface is in the form of mobile application as well as desktop application. The main goal of the Real-time FER is to indicate to the agent using the application what the most efficient route would be to take in the case of an emergency. The system will perform this functions by anonymously tracking the building populations to be aware of the building's current state. It will be done with the use of a heatmap, which is generated using the Bluetooth sensors installed throughout the monitored location. This will show the quantity and distribution of the population throughout the building.

The escape routes are pre-planned by the building designers, since there are special considerations that need to be made, including which walls have fire proofing, and which areas should be avoided for safety reasons. The heatmap is then to be consumed by an AI algorithm to assign a fire escape route to each agent in the building. The building's population distribution and available routes must be taken into consideration when assigning a route to each individual.

With the building heatmap and time sensitive fire escape route in place, each agent in the building will be sent a push notification to his/her phone indicating what escape path to follow in case of emergency.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
FER	Fire Escape Route
WebUI	Web User Interface
RFC	Request for Comments
AI	Artificial intelligence
CS	Computer science
COS 301	Abbreviation for the module named "Software Engineering"

1.4 References

Department of CSE. (2015). Software Requirements Specification ATM. In: Department of CSE Software Engineering and Project Management Lab Manual. Indore: Department of CSE. 1-17.

1.5 Overview

The sensors will be placed in various locations, the server will then iterate through the sensors fetching data from each sensor, the server will map the data from various sensors into relevant locations. The server will then process the Map data generating various routes depending on agent locations and send it to the applications of the various agents. The application will sync the latest version from the server upon opening. The WebUI will be used to view a live view of the map as well as allow new agents to register into the system.

- **Section 1:** Describes the main purpose of the system, it's scope, and relevant information to interpret the rest of the document.
- **Section 2:** Specifies the different users that will make use of the system and it's sub-systems, and their expected adaptability to the new system.
- **Section 3:** Specifies the components and the interactions among them that are required for the system to be functional.
- **Section 4:** Outlines the Quality requirements that the system needs to fulfill, and a means of testing whether the system adheres to these requirements.
- **Section 5:** Shows the requirements of the system, and whether they are met by the Use-cases.

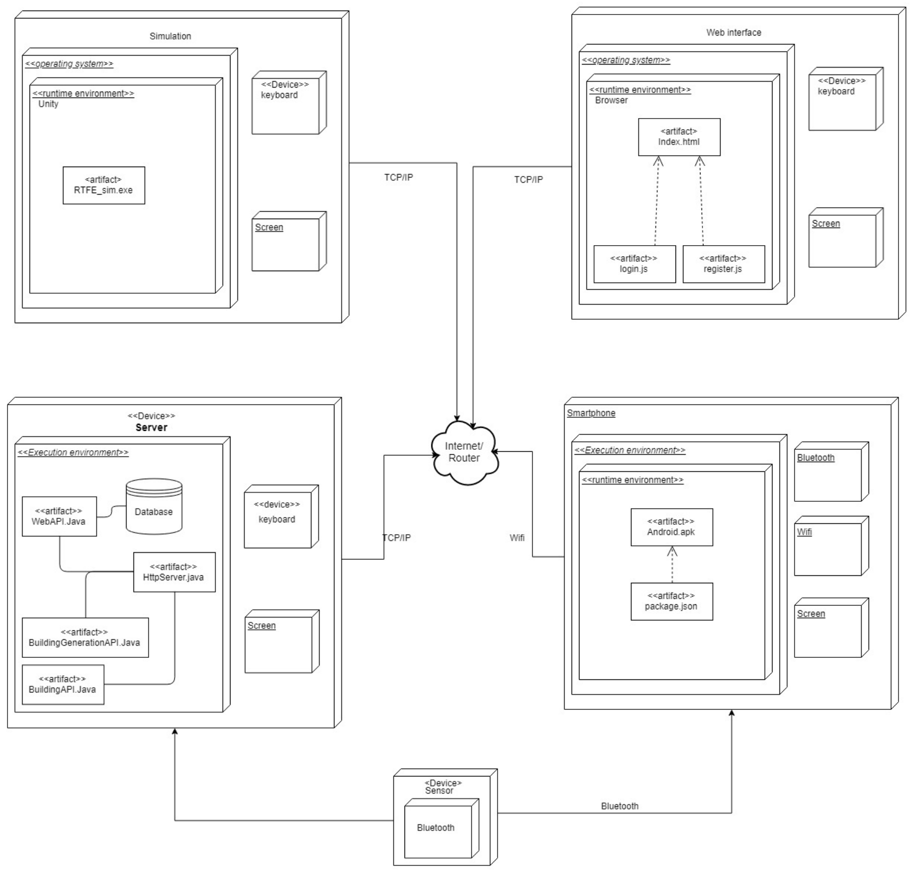
2 User characteristics

The two user group interacting with the application are:

Administrative users: these users will have full access to the system. They can add locations, upload escape routes, manage the user account on the system. Such users will mostly interact with Real-time FER web application. However, in case of emergency such users will also use Real-time FER application and will receive assigned fire escape route in the same manner as regular users. Potential administrative users would possibly be higher level managers, HR managers and security officers that occupy and work in a business that uses the Real-time FER system.

Agent/regular users: these users will interact with the application interface primarily in the event of emergencies. In order to become a user of Real-time FER application one must be registered by one of the administrative user. Even further, registered individual will be considered as active agent only if one is physically present in a building that uses the Real-time FER system. Regular user will not be able to create one's own account in the system. Thus, potential agents would possibly be employees of the company.

3 Deployment Diagram



4 Architectural Requirements

4.1 Architectural Design Objectives

It is of paramount importance to establish a clear picture of architectural design objectives that will allow to get a good understanding of the overall structure of the Real-time FER system. Architectural design objectives of this system are derived from the software requirements including quality and security requirements. Therefore, to maintain logical flow of the document, *Quality Requirement* section was included below. Those requirements and factors should be considered before proceeding further with architectural design process.

4.2 Quality Requirements

4.2.1 Performance

Performance is a measure of how well the system works as well as the time it takes for operations to take place, under any circumstance(high volume usage, poor signal areas). Essential factors that needed to be tested to ensure that the performance is of a high enough standard are as follows: Testing the system under large population volumes, the speed at which route assignments are calculated and the assignment of the determined routes to the individual agents. A stress test can be formulated in order to see how well the system works under heavy load with the quantitative measure being the systems success: failure rate. However the main metric that will be used to measure the performance testing will be time. The speed at which the system responds and completes its intended task is the critical measurement.

- The system shall be capable to track the current state of the building as well as update the heatmap that represents the state in a real time. The maximum delay that the system is acceptable to have to perform this function would be 30 sec, ideally 5-15 sec, delay.
- The system shall be capable to automatically assign escape routes to active agents in maximum of 30 sec delay from the moment fire detectors detected a fire breakage.
- The system shall be able to push assigned route to agents with maximum delay of 10 sec.
- During the same period of time, that the system took to assign the route, the notification about assigned route should also appear on agent's device. The delay between those two event should not exceed 2 sec.
- In case if system was off and was only switched on during an emergency, the system shall be able to take no more than 125 sec to be fully loaded, that will include the performance of all the steps mentioned above.

4.2.2 Reliability

It is imperative that the system works as expected and maintains how it works over the course of its expected life time. The importance of reliability can not be overstated, since if the system fails it can result in the loss of life of the buildings occupants. In the case of the system failing to process data in a timely manner, the system needs to assign the users default routes to begin evacuation. We will check that the software can perform failure-free operation for a specified amount of time across a range of mobile devices. We can then simulate an unacceptable response time and ensure that the system does what has been previously described.

- The system shell be able to detect current position of the agent and assign the most appropriate and safe escape route to that agent based on one's current position. The allowed fault of position detection is +/- 2.5m. However, the system requires to assign 100% accurate route to each agent.
- The system shall be able to work in priority manner and push notification first to those agents that are in potentially more hazardous and dangerous places in the building.
- In case of connection failure (in case of Bluetooth signals being lost), the system shall send an SMS to the agents with the short, bullet point descriptions of the escape routes they must take from their location. However, such functionality can only be applied to companies that store their employees phone numbers in their databases.

4.2.3 Scalability

Scalability refers to the expandability of the system, this is determined by how well it can be implemented in buildings which are densely populated and/or large locations with a large amount of different escape routes. This also extends to how the system will work in areas with perfect signal opposed to limited signal areas. To test this quality requirement we will test the performance(failure/success rate) of the systems in the various specified areas which have been named above as well as the amount of people able to make use of the new system effectively.

- Ideally the system shall be able to broadcast newly assigned routes to all the agents at once. However, realistically it is impossible to achieve. Therefore, the system shall be able to notify at least 50 agents at a time.
- The system shall be able to work with various building structures. Therefore, there should be 100% adaptability of the system to any random building structure the moment it is registered to the system.

4.2.4 Security

This quality requirement refers to how the system will protect confidential information its database stores as well as its users from any potential danger including: information leak and hacker attacks. It is very important to pay special attention to security requirements of the system due to a "large number of discovered vulnerabilities" in today's systems (Silva & Danziger, 2015). The lack of security in a system may lead to an information leak and that may cause financial loss of a business that consumes that poorly designed system. Even further, in extreme cases, it may result in the loss of individual lives.

- The system shall not collect any extra data from agents except their current location in the building.

- The system shall protect all the confidential information, like fire escape routes images of a building and agents information, that was provided by a business that uses the system. Ideally, the system shall provide 100% protection to mentioned above information.
- The system shall only provide access to Real-time FER Web Ui and Application interfaces only if individual/organisation that requested access is actually registered in the system.

4.2.5 Maintainability

This quality requirement refers to how easy the system can be maintained. How flexible the system is to changes. Wisely established maintainability requirements ensure proper version control, ease of maintenance and bug fixing and successful system modification.

- The system shall be simple to maintain. The system will be divided into subsystems that would be independent from each other. Therefore, when there is an update in one of the system's functionality, other subsystems will not be affected by that update and the system as a whole would not fail.

4.2.6 Usability

This quality requirement refers to how easily the system can be learned and effectively used. Any system components which the user interacts with must be designed in such a manner that it feels familiar thus making its operation easy. Since many people have embraced the use of mobile applications, it should not be that difficult to extend this functionality. The system should include the use of a help tab which will provide clear tips/instructions to assist in specific problems. Usability tests will be implemented to measure the effectiveness of the clients interaction with the system. These tests measure 5 performance based on the success rate at performing various tasks and are normally used in the form of questionnaires(after sufficient time using the new system the clients will be prompted to answer and can accept or decline).

- In most of the time the system shall be fully automated. It should perform all the required functions without any or with little guidance. However, in special case like: there is an ignition, but fire detectors failed to perform their functions , system shall provide a simple, one level deep, user interface with all the functions listed on intro page.
- There should be no more than one step (clicking) to evoke any of the main functions of the system.
- The system shall constantly notify a user about any event it performed, or even more important, failed to perform; like in case if system was

unable to send newly generated escape routes to agents, the system will send notification to user which will be displayed on status bar (in case of smartphone) or on Windows Push Notification Service.

4.3 Constraints

The major constraints of the system include:

- The Real-time FER requires each agent to switch on Bluetooth for the proper activation of the application.
- The Real-time FER needs to be installed on a device that has Bluetooth connection.
- The Real-time FER will not send any notifications to a registered agent who is not in the building.
- The Real-time FER system will not accommodate agents with fire-escape routes that are not registered in the business's Real-time FER UserDataDatabase.
- The Real-time FER will not activate its functionality in a building whose fire escape routes are not uploaded to the system's SystemDataDatabase.
- The Real-time FER's databases must be hosted on a business own servers.
- The Real-time FER will only function if a building, in which the system is planned to be used, is equipped with Bluetooth sensors.

4.4 Technological Decisions

4.4.1 Software

- **Java:** there were several reasons why Java was selected among other programming languages. Firstly, Java is an object-orientated programming language. OOP concept helps us to design our subsystems as independent entities. Such conditions will ensure low coupling between our subsystems in the overall Real-time FER system. Secondly, Java simplifies the whole concept of threading. We use threads to increase the performance of our system. Thirdly, Java is relatively simple language to work with and debug, therefore we use it to speed up our development process.
- **IntelliJ IDEA:** Java integrated development environment (IDE) that has very powerful debugger. Such feature helps us to eliminate any logical and syntax errors in our code.

- **Android Studio:** this IDE was designed precisely for Android application development. Its main goal is to accelerate the application development while making it simpler at the same time.
- **Unity:** this program was chosen due to its powerful visualisation of provided code. We chose Unity in order to visualise our system's functionality as well as to train our AI to be able to function in different situations.
- **C#:** we writing code that we apply in Unity in this language, as it is the language that Unity supports.
- **Git-Hub:** it was chosen for its features that allow version control of the code base and collaboration between team members.
- **Travis CI:** due to its availability, high quality performance and relatively simple usability, we chose Travis CI as the tool that will support our continuous deployment of our latest code. With such tool we will be able to rapidly detect any possible conflicts between our commit and the rest of the code on the brunch. That will help us to fix the code on our side without breaking our overall test.

4.4.2 Libraries

- **JSONObject:** this is a subclass from java.util.HashMap package. This subclass is used by us to parse JSON string to the string that Java can manipulate with.

4.4.3 Hardware

- **Bluetooth Sensors:** required hardware with which our system will going to communicate through the use of Bluetooth signals. This hardware will allow our system to anonymously track the population of the building in the real-time and generate population heatmap that, in case of emergency, will be used as one of the inputs for route assignment.

4.5 Type of Application System

It is an important procedure to establish clear type of the Real-time FER system, as established type will influence "modeling, analysing, design, implementation and testing of the system" (Kung, 2014). The Real-time FER system is an event-driven system that contains interactive subsystem, transformational subsystem and object-persistence (database) subsystem.

4.5.1 Why the Real-time FER is an event-driven system?

The Real-time FER system clearly has the characteristics that are required to have in order to be considered as event-driven system. Those are the following required characteristics, as stated by C.Kung in his "Object-Oriented Software Engeneering" book, for event-driven system together with an explanation where we can find those characteristics in the Real-time FER system:

- The Real-time FER system must receive and control external entities that collaborate or somehow related to the system. The Real-time FER has such characteristic. It receives events from fire detectors, in case of fire breakage, and it controls Bluetooth signal strength changes that are send from Bluetooth sensors when they capture Bluetooth signals of active agents (devices with Bluetooth on) that are registered in the system's user database.
- Another important characteristic of event-driven system is that the requests from external entities has no fixed sequence. Usually they randomly arrive at the system. The Real-time FER also encapsulates such criterion. Request arrive at the system randomly. There is no fixed flow of requests in the system. Let us look at the following example. It can be assumed that one of the business's buildings,that uses the Real-time FER system, at specific period of time is empty (no human-beings currently occupy this building). There is also no fire breakage; therefore, there will be no requests send from external entities to the system. However, later during the day that building might be overcrowded with people and there are sudden fire breakout. In such situation system will receive quite a lot of requests at once.
- According to Kung, "Event driven systems do not have to respond to every incoming event". In order to proof that such characteristic is also present in the Real-time FER, the first example stated above can be extended. It can be assumed that one of the business's buildings,that uses the Real-time FER system, at specific period of time is empty (no human-beings currently occupy this building). However, there is a fire breakage; therefore, fire-detectors are evoked and will send request to the system, but the system would not react to that request as there is no active agents to be notified.
- Event-driven systems are often interact with more than one hardware or software devices. The Real-time FER system will interact with Bluetooth sensors and fire detectors.
- One of the characteristics of event-driven system is that system's state does not reflect progress of a computation. The Real-time FER system may be in idle state, than if, for example, someone of the agents will enter the building with one's Bluetooth on, the system will be activated by Bluetooth sensors, it will start to generate heatmap, however the moment

that agent will live the building, the system will return to its idle state.

- Event-driven systems are required to meet some timing constraints, temporal and timed-temporal constraints. The Real-time FER system also requires to meet such constraints. In our *Quality Requirements* section, under *Performance* subsection, it is stated that the moment fire breakage has occurred, the system must automatically assign escape routes to active agents. Such requirement can be also seen as timing constraint. Example of temporal constrain would be that the moment route to agent was assigned, agent must get a notification on one's device that notifies that route was assigned to one. Example of timed-temporal constraint is also stated in *Quality Requirements* section, under *Performance* subsection. It states that, "During the same period of time, that the system took to assign the route, the notification about assigned route should also appear on agent's device. The delay between those two event should not exceed 2 sec." This requirement specifies timing constraint between those two events.

4.5.2 Why the Real-time FER contains interactive subsystem?

The general description of interactive subsystem is that it carry out fixed sequence of interaction between an actor and a system. Actor evokes a system and finishes the session with that system. In the Real-time FER there are two subsystems that doing that: Application Subsystem and Web UI subsystem. Both of those subsystems resemble qualities that are stated for interactive system in "Object-Oriented Software Engeneering" book by C. Kung. Those qualities are:

- Fixed sequence of actor requests and system responses. In Real-time FER Application Subsystem and Web UI subsystem good example would be login use case. Actor inputs one's login details and presses login button (actor requested access). The next event would be when system responds to actor's input by providing actor with requested access or rejecting the request.

- Ability to process and respond to each request from the actor. In Real-time FER Application Subsystem and Web UI subsystem such qualities are present. Going back to the login example. When actor provides login details and presses login button, subsystem (either Web UI or Application) would first process the information that was send by the actor. It sends information through Communication subsystem to Database Manager side where that information is validated. Afterward, the subsystem responds to the actor by providing requested access to the interface, or oppositely, informing actor about incorrect information that was being provided.

- Supports client-server relationship. Thus, it means that subsystem allows actor to request services and the subsystem will provide this service. The Real-time FER Application subsystem and Web UI subsystem possess such quality. For example, it is planned that Web UI's interface will have an option that will allow administrators to evoke route assignment function, in case if the subsystem failed to do so automatically, and the subsystem will perform this function (provide this service).

- Interactive subsystem's state reflects the progress of a process that was being evoked by actor. The Real-time FER Application Subsystem and Web UI subsystem possess such quality. If to consider login use case again, the progress between actor's request to login and subsystem's response, in terms of providing such access, is clearly visible; at the beginning of the process actor was on login page, after request was processed the subsystem's state changes and actor observes index page of the subsystem.

4.5.3 Why the Real-time FER contains transformational subsystem?

Transformation subsystem is used in Real-time FER system. This system is clearly visible in the Processing subsystem. This subsystem retrieves data from communication subsystem (that was received from Bluetooth sensors) and interprets this data into its visual representation. According to Kung, transformational subsystem resembles following attributes:

- Usually such system type will have a network of information-processing activities. Where each activity transforms received input into output.
- Usually those networks of activities involve conditional branching, parallel threading, synchronous and asynchronous behaviour.
- There usually no interaction between a transformational system and an actor during transformation of input into output.
- Such system type may require number of computational intensive algorithms.

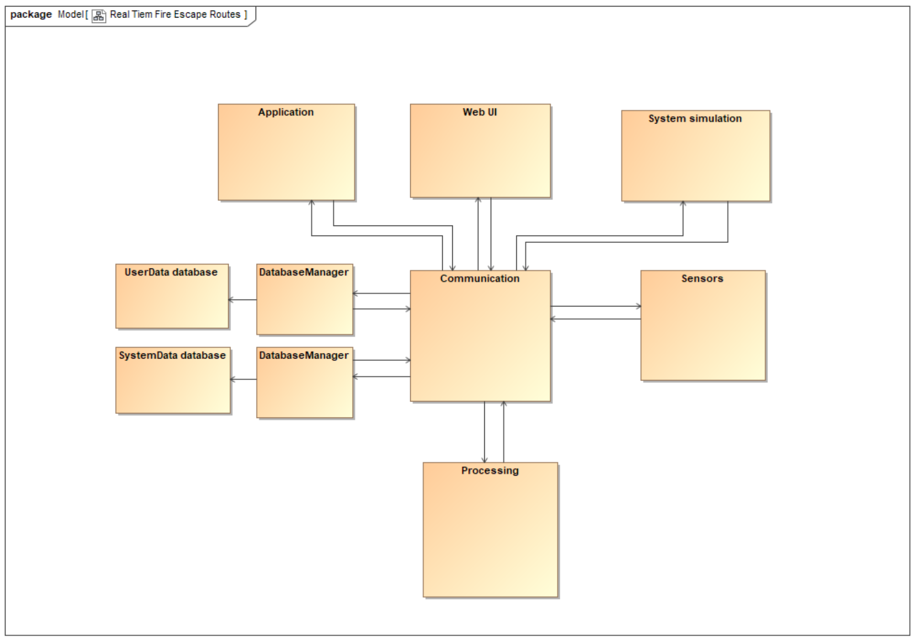
4.5.4 Why the Real-time FER contains object-persistence subsystem?

In general, object-persistence subsystems are subsystems that are capable of storing and retrieving data, as well as hide and protect stored data from unwanted changes. In case of the Real-time FER object-persistence subsystem is represented by the Real-time FER Database subsystem. The following characteristics of the object-persistence subsystem can be observed in Database subsystem:

- It should hide and protect database from the rest of subsystems, so those subsystems will not alter the data stored in database.
- Such system is responsible for storing, retrieving and updating database content.

4.6 Architectural Style

In order to proceed with the development of a system of specific type (or mixture of types) it requires to identify suitable architectural style that will match the system under development.



The Real-time FER Domain Model

The main architectural style used by the Real-time FER is the **Event-driven system architecture**. One of the characteristics of event-driven architecture is that it consists of the state-based controller and number of other subsystems under its control. This can be seen by the fact that the Real-time FER system uses a **state-based controller**, represented as *Communication* subsystem. Another characteristic of event-driven architecture is that the subsystems that are under control send events to the controller which processes the event and send back to a subsystem appropriate response. The Real-time FER *Communication* subsystem receives, in random manner, requests from all the other subsystems in the Real-time FER system, including:

- *Web UI and Application*, that usually would request controller to send through data or another request to the Database Manager.
- *Sensors*, that usually would invoke the *Communication* subsystem itself to change the current state of the system.
- *Database Systems*, that usually would send back to *Communication* sub-

system a data that was requested or notify the *Communication* subsystem about any changes or failures on the data-side.

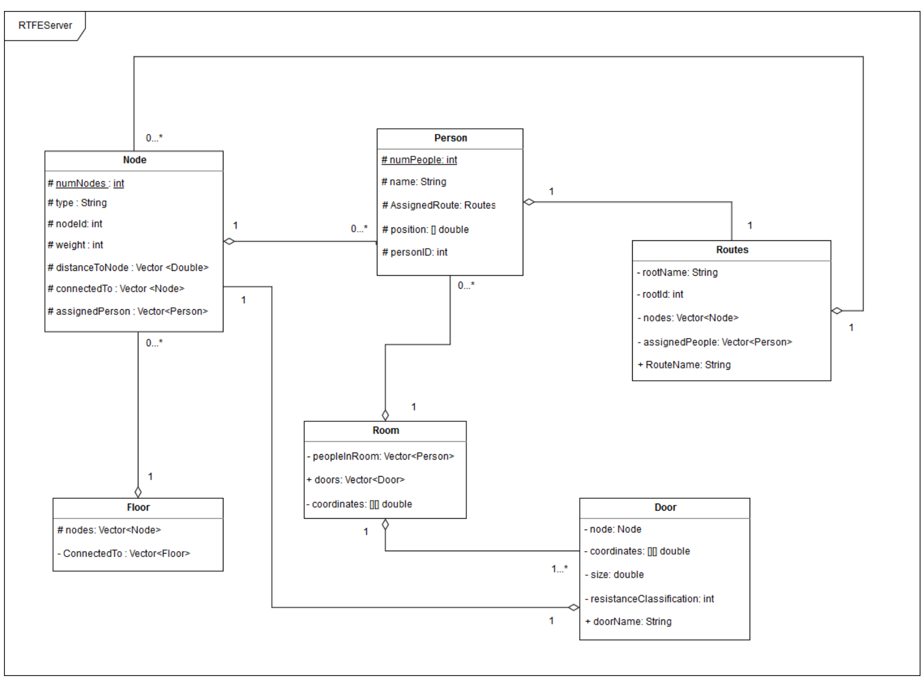
There are also **object persistent architectural styles** present when it comes to the databases and their management. It is clearly noticeable that the Real-time FER Database Subsystem uses such architectural style. the Real-time FER Database Subsystem has DatabaseManager class that receives requests from Communicator to retrieve, store, modify or delete some (or all) of the data on one of the system's databases. According to Kung, such architectural style "greatly simplifies the design, implementation, and maintenance" of the system (Kung, 2014).

N-tier architecture is resembled in the Web UI and Application subsystems where each component of named systems is represented as loosely coupled layer with well defined functionality. For example, login page and than index page, get building page, and so on. All those pages represent one subsystem, however if any changes would be made to one page the other pages will not be affected.

5 Subsystems

5.1 Communication - S1

5.1.1 Domain Model



5.1.2 Functional Requirements

- R1. The Subsystem will facilitate data exchange between the Sensor Subsystem and the Processing subsystem.
- R2. The Subsystem will facilitate data exchange between the Processing Subsystem and the Application Subsystem.
- R3. The Subsystem will facilitate data exchange between the Processing Subsystem and the Web UI Subsystem.
- R4. The Subsystem will facilitate data exchange between the Processing Subsystem and Database Subsystems.

5.1.3 Actor-System Interaction models

Table 1: UC1: Pushing notifications to the application

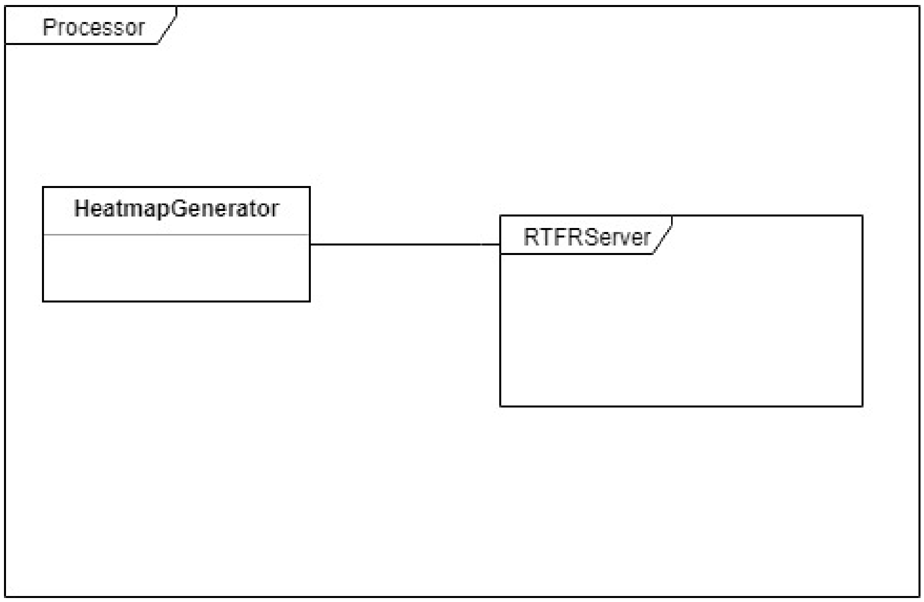
Pre-condition	
The user must be logged in to the application	
Actor:	**System:**
	0. The system receives alert from sensor subsystem
	1. The system send notifications to the application
2. The application displays the notification	
Post-condition	
The user receives an alert on the application	

5.1.4 Subsystem Traceability Matrix

UsesCases vs requirements	**R1**	**R2**	**R3**	**R4**
UC1	x	x	x	x
Total	1	1	1	1

5.2 Processing Subsystem - S2

5.2.1 Domain Model



5.2.2 Functional Requirements

- R1. The system will fetch population distribution information from Communication subsystem.
- R2. The system will be able to break retrieved information into appropriate data for constructing heatmaps.
- R3. The system will be able to interpret retrieved data into its visual representation.
  - R3.1. The system will build a heatmap from a retrieved data.
- R4. The system will push newly generated map to communication subsystem
- R5. The system will calculate the most efficient route for each agent.
- R6. The system will update available routes depending on sensor data.

5.2.3 Actor-System Interaction models

Table 2: UC2: Building a map

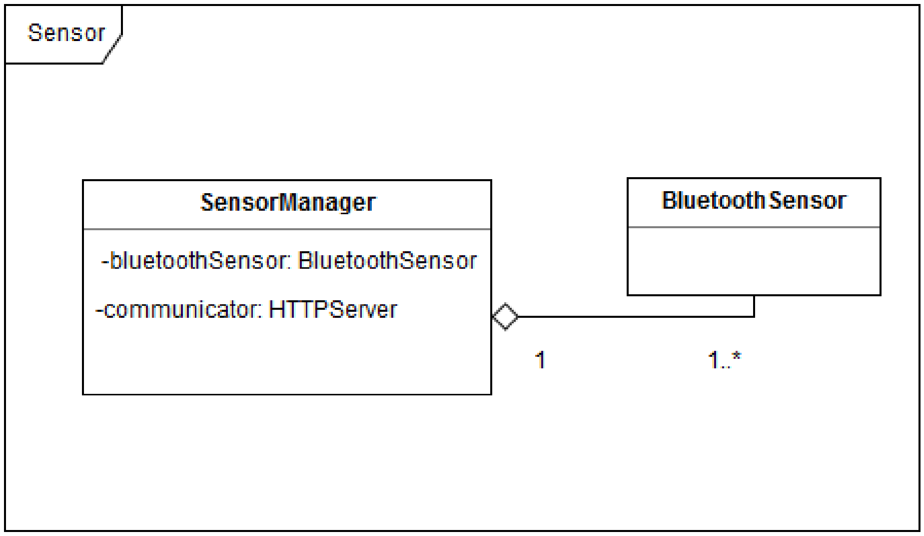
Pre-condition	
Admin is logged into system.	
Actor:	**System:**
0. The admin user inputs building coordinate data	
	1. The system processes the input data.
	2. The system constructs the building using the data 3.
	4. The system saves the map data.
	5. The system returns a success/error message
Post-condition	
The system contains a map structure	

5.2.4 Subsystem Traceability Matrix

UsesCases vs subsystems	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**
UC1	x	x	x	x	x	x

5.3 Sensors Subsystem - S3

5.3.1 Domain Model



5.3.2 Functional Requirements

- R1.The Sensor will be able to communicate with physical Bluetooth sensors through the use of Bluetooth signal.
- R2. The Sensor will push collected data, from Bluetooth Sensors to Communication Subsystem.

5.3.3 Actor-System Interaction models

Table 3: UC3: Sensor emergency Trigger

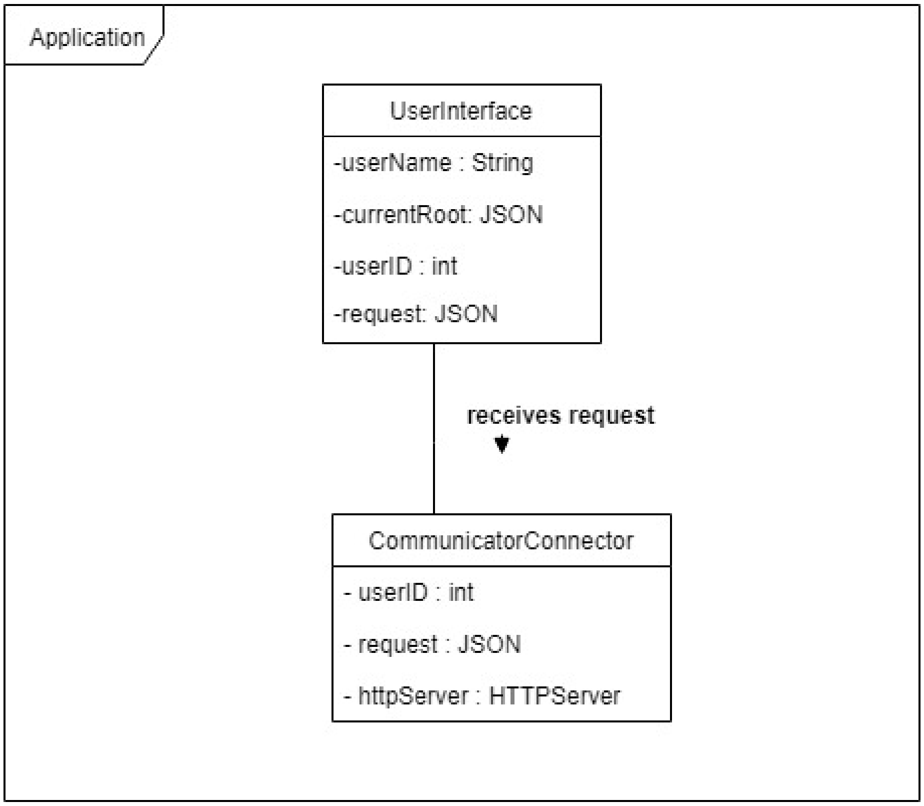
Pre-condition	
The sensors are active and connected to a network	
Actor:	**System:**
	0. The sensors detect a fire
	1. The sensors send the data to the processing subsystem
2. The system receives data from the sensors	
3. The system processes and responds to the data	
Post-condition	
The system responds to an emergency	

5.3.4 Subsystem Traceability Matrix

UsesCases vs subsystems	**R1**	**R2**
UC3	x	x

5.4 Application Subsystem - S3

5.4.1 Domain Model



5.4.2 Functional Requirements

The Application Subsystem will:

- R1. The subsystem will allow registered agent to login
  - The subsystem will take agent on to the main screen of the application, in case of success.
  - The subsystem will notify agent about incorrectly provided information if the information provided by the agent failed to be identified in the database.
  - In case of successful logging, subsystem should safe logging details - That will allow agent to close one's application, but one will still going to receive notifications in case of fire emergency.

- R2. The subsystem will allow logged agent to receive assigned route from the *Communication* subsystem in case of fire breakage.
- R3. The subsystem will allow logged agent to request a fire-escape route if one did not receive any assigned routes from the *Communication* subsystem in case of fire breakage.
- R4. The subsystem will notify logged agent about assigned route that one should take in case of fire emergency.

5.4.3 Actor-System Interaction models

Table 4: UC4: User (Agent) logging in to the mobile application

Pre-condition	
User has application installed on their mobile device	
Actor:	**System:**
0. The user opens the application	
	1. The system displays instructions for the user to identify themselves
2. The user inputs their data	3. The system sends the data to the processing subsystem
	4. The system receives success/error message
	5. The system displays the appropriate message
Post-condition	
The user is logged into the application system	

5.4.4 Actor-System Interaction models

Table 5: UC5: Viewing map on application

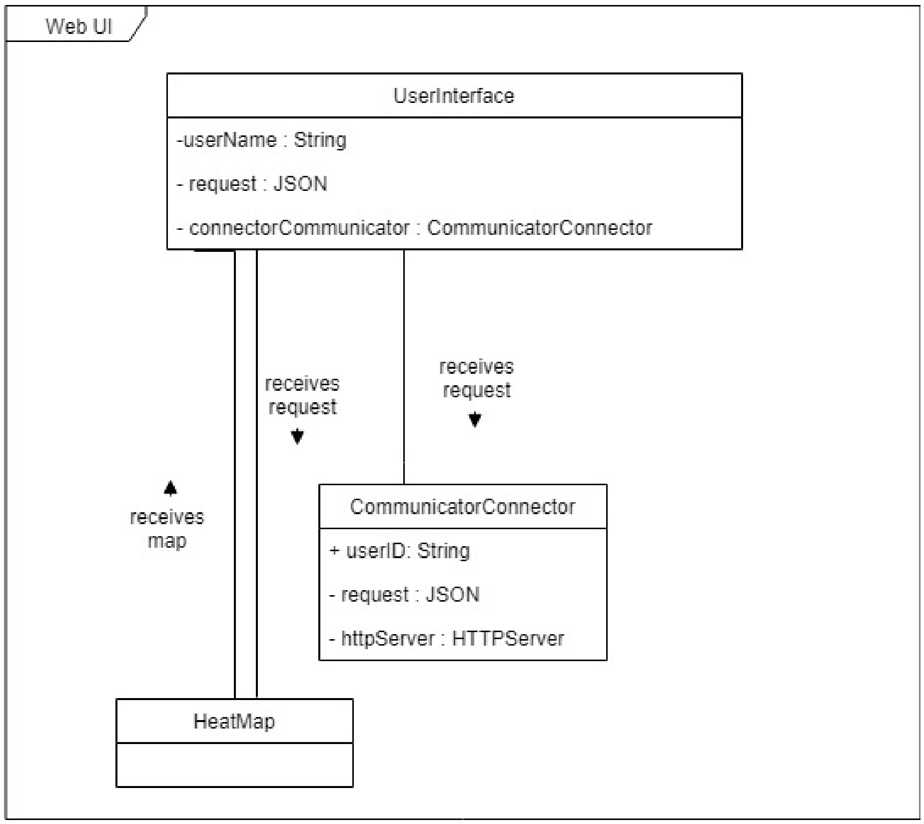
Pre-condition	
Agent is logged into system	
Actor:	**System:**
0. The user selects the view map option	1. The system makes a request to the processing subsystem
	2. The system responds with the appropriate data
	3. The system displays the data
Post-condition	
Map is displayed to the user	

5.4.5 Subsystem Traceability Matrix

UsesCases vs subsystems	**S1**	**S2**	**S3**	**S4**	**S5**
UC1	x				
UC2		x	x	x	x

5.5 Web UI Subsystem - S5

5.5.1 Domain Model



5.5.2 Functional Requirements

The Web UI Subsystem will:

- R1. The system will display a page that allows login data input from users.
- R2. The system will send data to the processing subsystem.
- R3. The system will display a page that allows data input for new users.
- R4. The system will send data to the communication subsystem.
- R5. The system will display appropriate messages to the user.

5.5.3 Actor-System Interaction models

Table 6: UC6: Administrative login to the system

Pre-condition	
User must have access to Web UI	
Actor:	**System:**
	0. System displays the instructions for the user to log in
1. The user inputs their information	2. The system send the information to the processing subsystem
	3. The system receives a success/error message
	4. The system displays the appropriate message
Post-condition	
User is logged in to the system	

Table 7: UC7: Registering a new user on the system

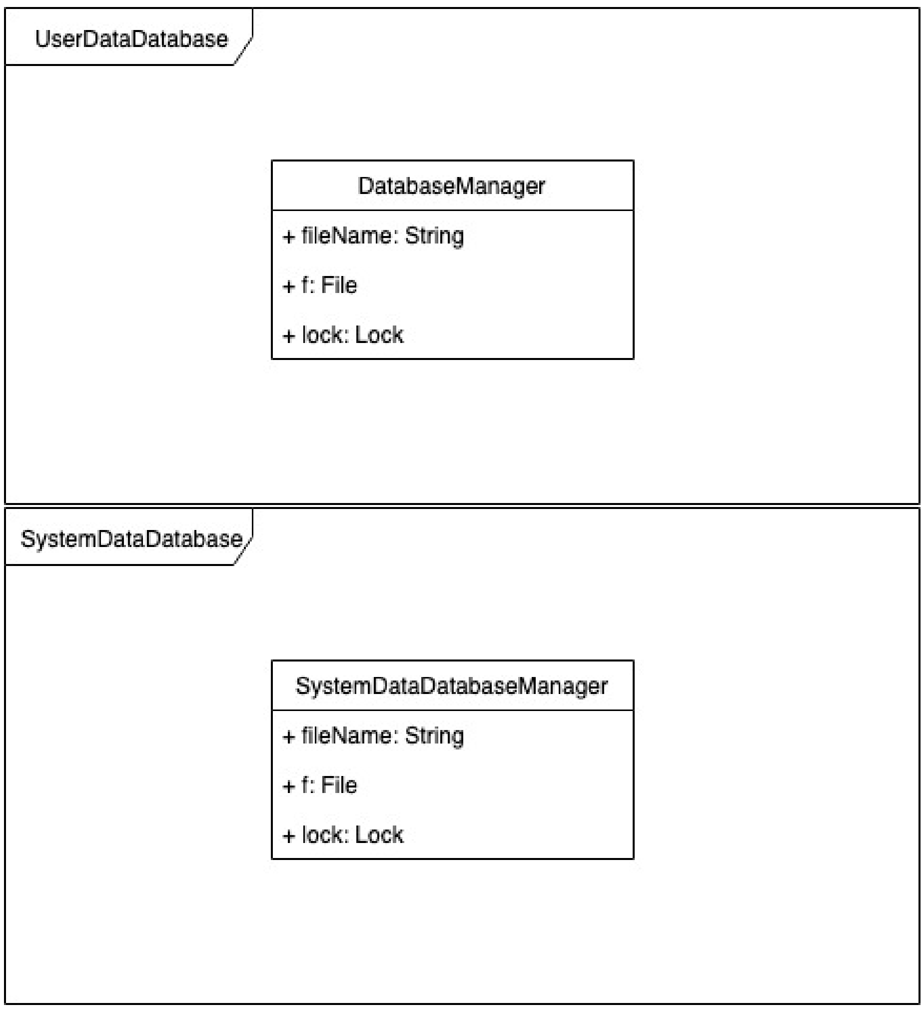
Pre-condition	
Admin is logged into system.	
Actor:	**System:**
0. The user navigates to the correct page	
	1. The system displays the instructions for data input
2. The user inputs the new users information	3. The systems sends the data to the processing subsystem
	4. The system receives a success/error message
	5. The system displays the appropriate error message
Post-condition	
A new user is registered on the system	

5.5.4 Subsystem Traceability Matrix

UsesCases vs subsystems	S1	S2	S3	S4	S5
UC1	x	x			x
UC2			x	x	x

5.6 Database Subsystem - S6

5.6.1 Domain Model



5.6.2 Functional Requirements

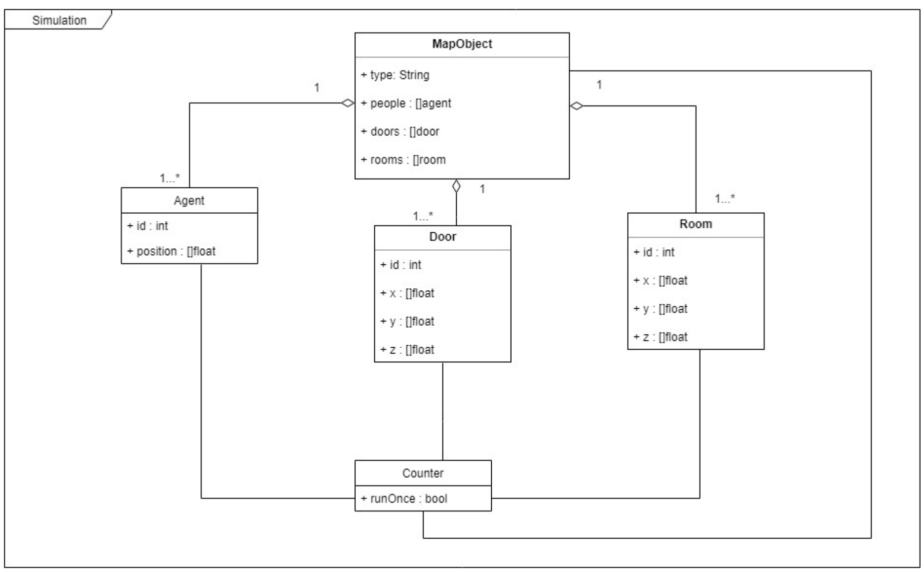
The Database Subsystem will:

- R1. The system will allow for the creation of new database entries.
- R2. The system will allow fir the reading of database data.
- R3. The system will allow database entries to be updated.
- R4. The system will allow for the deletion of database entries.

5.6.3 Actor-System Interaction models

5.7 Simulation Subsystem - S7

5.7.1 Domain Model



5.7.2 Functional Requirements

The Simulation Subsystem will:

- R1. The system will receive data from the communication subsystem.
- R2. The system will build dynamic buildings from the data received.
- R3. The system will display the building constructed from the server data.

- R4. The system will place agents in the building using data received from the other subsystems.
- R5. The system will display the evacuation of agents using data received from the processing subsystem.

6 Trace-ability matrix

UsesCases vs susbsystems	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**
UC1	x						
UC2		x					
UC3			x				
UC4			x				
UC5			x				
UC6				x			
UC7				x			

7 References:

Silva, M. A. Danziger, M. (2015). *The importance of security requirements elicitation and how to do it.* Paper presented at PMI® Global Congress 2015—EMEA, London, England. Newtown Square, PA: Project Management Institute.

Kung, D. C. (2014). *Object-Oriented Software Engineering.* The University of Texas at Arlington. Published by McGraw-Hill Companies, Inc. Avenue of the Americas, New York.