Design Document
Safestreets

Luca Massini   Daniele Nicolò

release date to be defined

Contents

1 Introduction

1.1 Purpose

This document is necessary to describe the architecture of the system from several points of view. This document has the scope of giving more details (especially to developers) about the system to be in terms of system architecture, in terms of software implementation and in terms of integration and testing. This document follows a top down approach. In fact as the reader will proceed toward the end of the document there will be an increasingly detailed description of all the design aspect concerning the system.

1.2 Scope

Safestreets is a mobile application that allows private users to inform authorities about parking and traffic violations. A user must take a picture of the violation, describe it and insert the place where it occurred. An image recognition algorithm is run in the back end on the sent picture. A user has also the possibility to see the safety of the streets and the parking areas and to check the most reported streets and vehicles.

The system offers also the possibility to receive suggestions in order to improve the streets safety. This feature is available only for municipality accounts. These suggestions are generated by the system using an algorithm that retrieves the information from the users' reports and also from the data about accidents given by the municipality.

1.3 Definitions,Acronyms and Abbreviations

1.3.1 Definitions

TODO

1.3.2 Acronyms

TODO

1.3.3 Abbreviations

TODO

1.4 Revision History

TODO

1.5 Reference Documents

TODO

1.6   Document Structure

The following sections are structured as below:

- **Architectural Design:** In this section there will be the description of the system from the architectural point of view. This means to show the components and their interactions with each other and to explain the design patterns choices and the architectural styles. The architecture will be shown both from a software point of view and also from a physical point of view.

- **User Interface Design:** In this section there is an explanation,in terms of user experience (UX), of the user interfaces already showed in the RASD mockups.

- **Requirements Traceability:** Here we describe how the requirements already explained in the RASD match with the design choices done in this document.

- **Implementation, Integration and Test plan:** Here there is the description of how we will implement all the components, how we will integrate them together and finally how we will test both the single components and also the integrated system.

- **Effort spent:** Here there is the division of the work hours of each member of the group and the description of the tasks completed and related time spent.

2 Architectural Design

2.1 Overview

In the figure below it is shown the architecture of the entire system. This description has to be intended not as an overview of the entire system. Details will be shown later in the document.
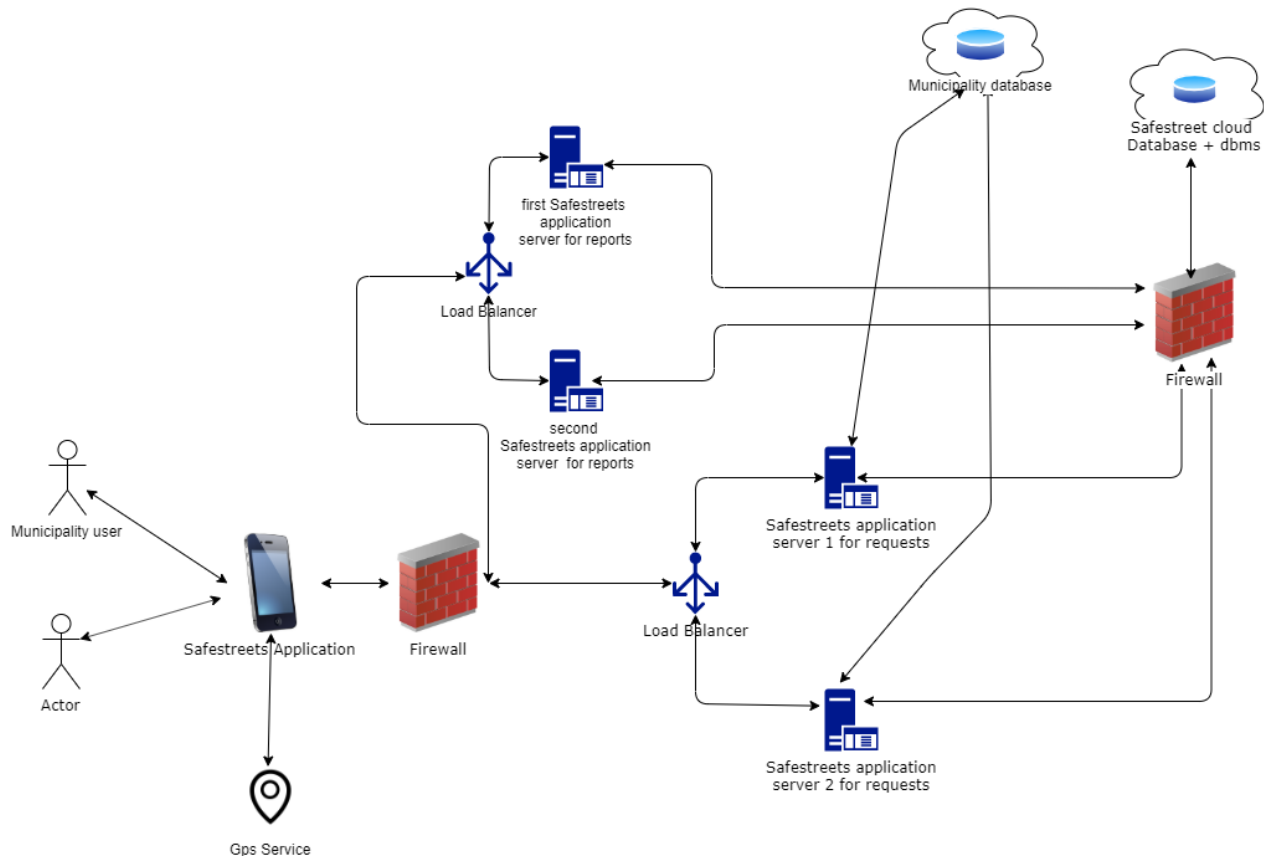

Figure 1: Architectural overview

From the diagram above we can immediately distinguish 3 layers that make up the entire architecture. Servers manage the logic one whereas databases manage the data layer and finally the client smartphone is where the presentation layer takes place.

The architecture has 2 clusters of application servers and two "type" of servers which deal with different tasks. One type of server handles the request by the user and the other one manages the reports by the private users. Every cluster contains 2 servers of the same typology. Every server contained in one cluster access the system database whereas instead only the two servers dedicated to requests access the external one (which is the external database of the municipality). Servers which resides in different clusters don't communicate with each other to guarantee the service. Also servers of the same cluster don't communicate with each other obviously. The workload balancing between servers of the same cluster will have to work thanks to load balancers.

The client in order to receive the service will speak directly and only to the firewall placed at the entrance of the system. The first firewall (the one near the SafeStreets application in the diagram) is the element that will rout the user message to the right cluster based on the type of request. In this case that firewall has not only a security function (obviously the main function of a firewall is that) but it also behaves as a gateway which routes packets. All the clusters are placed inside the DMZ (between two firewalls) for security reasons. They must be accessible from outside the internal network. The second firewall divides clusters from the system database. The database will have to be accessible only from the clusters for security reasons and because of this fact they are outside of the DMZ.

2.2 Component view

Here there is a description of the software components that will be implemented in the specific hardware. The diagram is not horizontal in order to make it more readable.
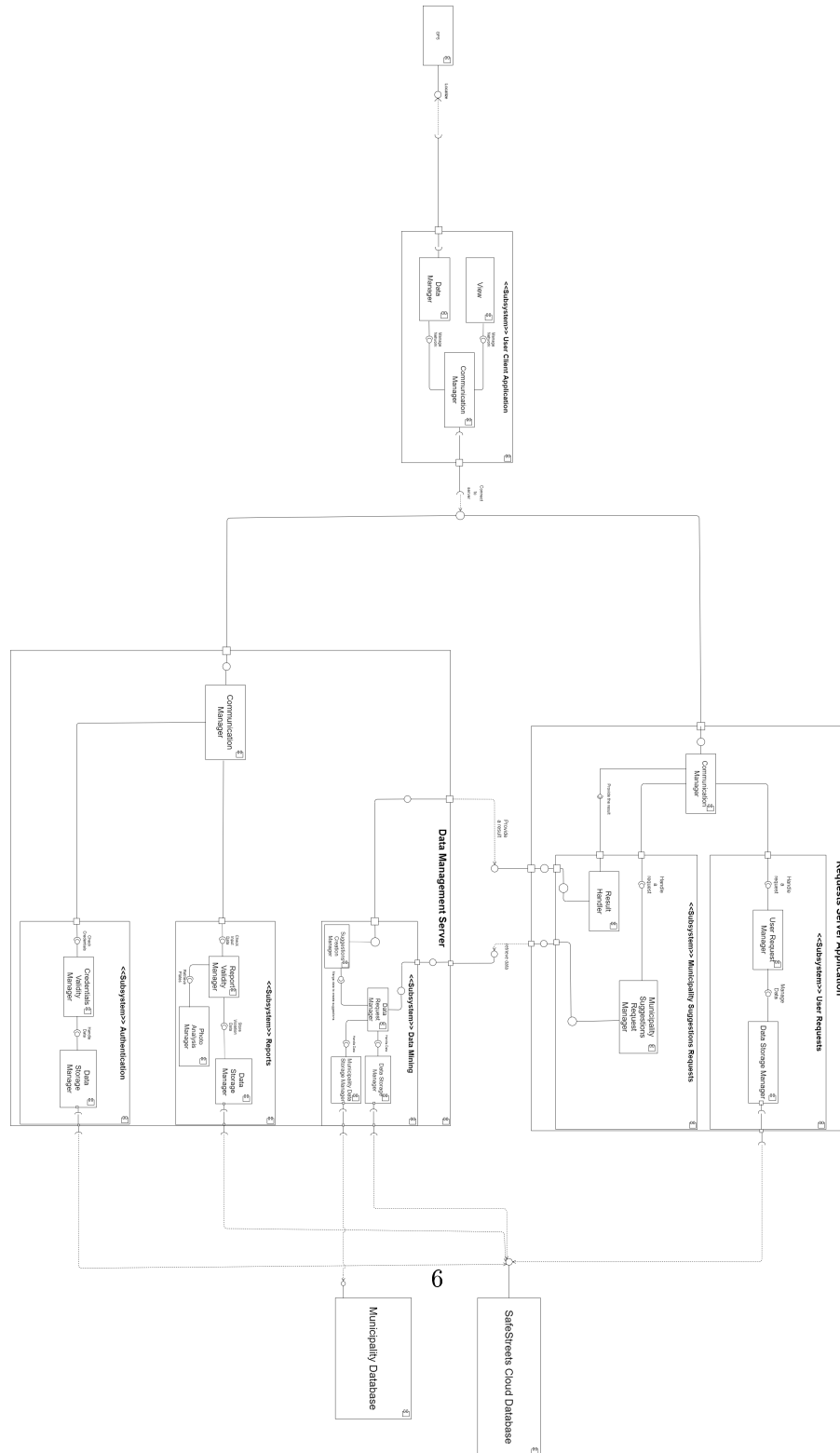
Figure 2: Component diagram

With this diagram our purpose is to show the internal software architecture of SafeStreets.
The main components are the user client application, the requests server application and the data management server. The external interfaces are SafeStreets cloud database, the municipality database and the GPS interface.
These main components contain smaller modular components that communicate with each other and with the external interfaces to complete different functions. Now we will describe the internal components and the external interfaces in detail.

User Client Application The User Client Application is situated in the user's smartphone. It contains the Data Manager, the View and the Communication Manager:

- **Data Manager**
  This component is responsible of taking the user's position in the world using the GPS service offered by his smartphone. Those data will be sent to the server in case he selects a service that uses the GPS localization.

- **View**
  This component is the view part of the MVC pattern. It takes data provided by the controller situated in the Server and shows them to the user with its interface.

- **Communication Manager**
  The Communication Manager is responsible for the network. It manages all the communication from the client to the server and vice versa.

Requests Server Application   This component is situated on one of the two servers and contains a Communication Manager, the User Requests component and the Municipality Suggestions Requests one:

- **Communication Manager**
  It has the same function of the one situated on the client. It dispatches and receives message from the client application.

- **User Requests**
  This component manages all the requests coming from the private users and the municipality, except for the suggestions requests. It contains a User Requests Manager, that elaborates the requests and asks for the necessary data to the Data Storage Manager. The User Requests component contains also the Data Storage Manager that retrieves the requested data querying the SafeStreets external database.

- **Municipality Suggestions Requests**
  This component deals with the suggestions requests made by the munic-
  ipality. It contains the Municipality Suggestions Requests Manager that
  receives the request from the Communication Manager and asks for the
  needed data to the Data Mining component, situated in the other server in
  the Data Management Server Application. When the suggestion is ready,
  the Data Mining component send it back to the Municipality Suggestions
  Request component, especially to the Result Handler, that dispatches it
  to the Communication Manager that sends it to the Client Application.

Data Management Server Application This component is situated on the
other server compared to the Requests Server Application. It contains the Data
Mining Component, the Reports Component and the Authentication one:

- **Data Mining**
  This component is responsible for the retrieval of the information needed
  to provide suggestions to the municipality. It receives from the Municipal-
  ity Suggestions Requests Component a request for suggestions. It contains
  the Data Requests Manager that handle the received request and asks for
  data to other two components, also situated in the Data Mining Compo-
  nent. They are the Data Storage Manager that provides the data about
  the violation and the streets that SafeStreets has in its cloud database,
  and the other is the Municipality Data Storage Manager that provides
  the data, again about the violations and the streets, but now querying
  the Municipality external database. When all those data arrive to the
  Data Requests Manager, it dispatches them to the Suggestions Creation
  Manager, that runs a data mining algorithm provided by an Oracle API in
  order to merge the received data and retrieve information useful to provide
  suggestions to the Municipality. Finally the result is sent to the Result
  Handler, situated in the Municipality Suggestions Requests Component,
  that proceeds to give it to the User Client Application.

- **Reports**
  This component is responsible for the violation reports. It receives the
  reports from the Communication Manager and it contains a Report Va-
  lidity Manager that checks that the violation data inserted by the user are
  correct. Then it has the Photo Analysis Manager that retrieves the license
  plate of the violator from the photograph using an algorithm provided by
  an external API that scans the photo. Finally there is a Data Storage
  Manager that takes all the information about the violation (license plate,
  type of violation, description, position, date and hour) and stores them in
  the SafeStreets database.

- **Authentication**
  The Authentication component manages all the operation about login,
  sign up and logout. It contains a Credentials Validity Manager that checks
  the user's input (both in login and sign up) and checks if he is logged

in whenever he wants to logout. Then it contains also a Data Storage
Manager that asks for the account data that has to be checked by the
Credential Validity Manager and stores the user's account data in case of
sign up.

External Interfaces   Some of the components already described have to com-
municate with external interfaces in order to guarantee specific functionalities.
These interfaces are:

- **SafeStreets cloud database**
  This component is shared with both Servers. In each of the two servers
  there is a component that is the only one that communicates with this
  database and it's the Data Storage Manager, that can both query to re-
  trieve data and add new information into the database.

- **Municipality Database**
  This component communicates with the Data Management Server Appli-
  cation, in particular with the Municipality Data Storage Manager, that
  can only query to obtain information in order to create suggestions for the
  Municipality. This database is in read-only mode.

- **GPS**
  This external component is necessary for the user's automatic localization.
  It communicates with the User Client Application, in particular with the
  Data Manager that is the one specialized in retrieving the position in-
  formation (city and street) from the GPS service offered by the user's
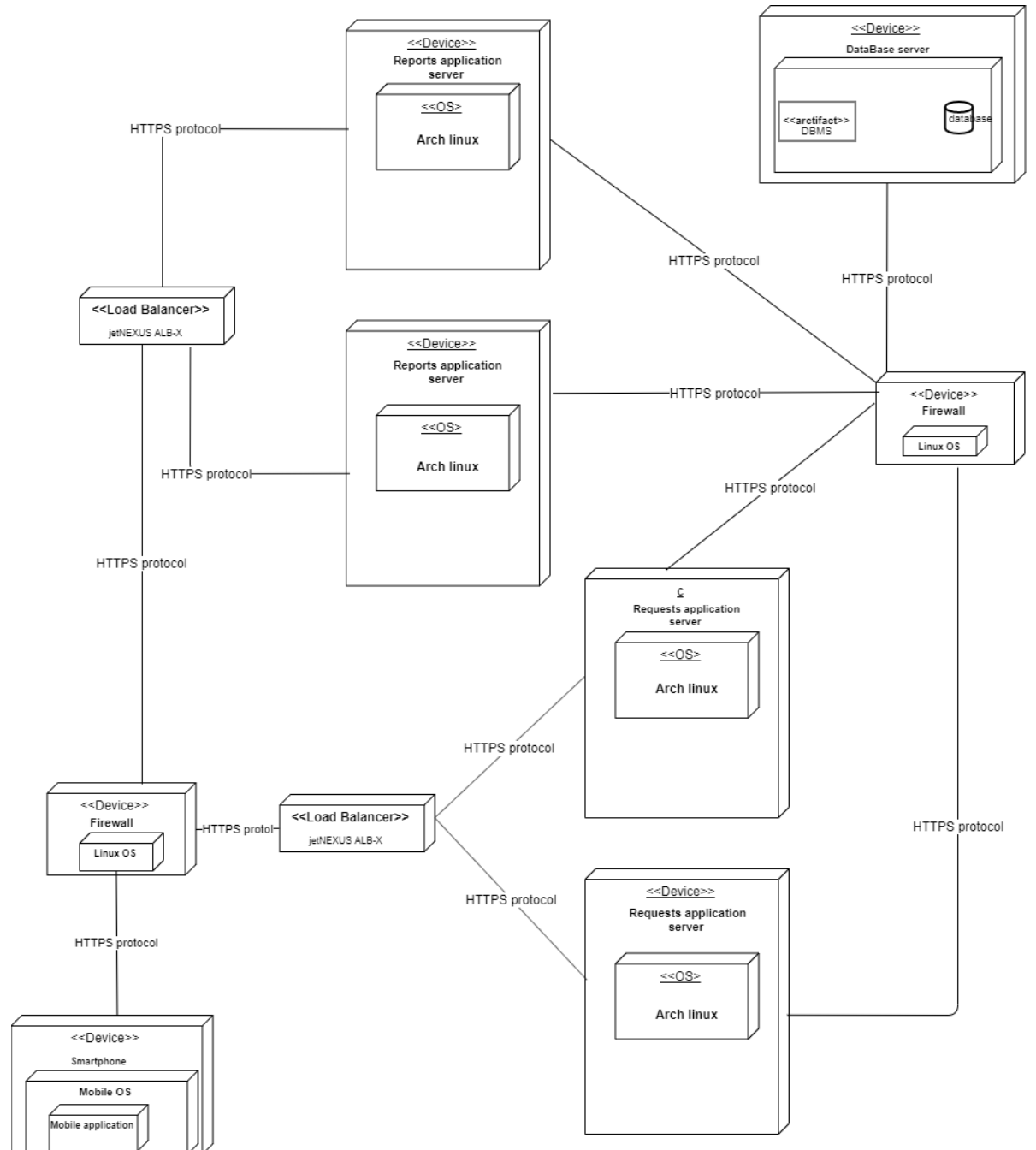 smartphone.

2.3 Deployment view

Figure 3: Deployment diagram

This diagram is very similar to the one in the overview. It express some aspects which have been already explained and some others are explained only here. In fact we can see all the operating systems for each server and also the protocol used in the communication between the devices. Here you can find a brief description of what the single components do:

- **Firewall:** Firewalls are well known for sure by the majority of the people but we think that it is important to describe their operation the same. Firewalls are placed between the internal network and the external ones.They are used to filter the incoming/outcoming packet traffic. Through a set of rules they decide if a packet can pass or (unique alternative) be blocked. This can be very useful to guarantee a better system safety.

- **Load Balancer:** Load Balancers are used to improve the workloads across multiple computing resources. Thanks to load balancer the load can be balanced in an optimal way.This can make the system more performing and can also improve the fault tolerance of the system because of redundancy.

- **Servers:** In this architecture we can find two clusters, each one containing 2 application servers. The servers are specialized with no overlap of functionalities. One is specialized in the managing of the reports by the private users. It must update the database with the new incoming data only after having done all the checks about the validity and correctness of the report.
  The other server deals with the managing of the requests by the private users or municipalities. It is in charge of check the correctness and validity of the request,and then it must run all the data mining algorithms necessary to retrieve the result and finally send the results to the right private user.

- **Databases:** The databases are cloud databases managed thanks to an API that will be provided by the provider of the service. We have only one cloud database used to store all the Safestreets data. Another cloud database is the external one of the municipality. Safestreets has only a special view of the part of the municipality database concerning the accidents. All these cloud database use SQL.

2.4 Runtime View

Sign Up Runtime View

With this sequence diagram we explain which components come into play and how they interact when a user signs up. This process is the same both for private users and municipality.
When the user pushes the sign-up button, the View sends the sign up request

to the Client Communication Manager that provides to dispatch it, together with the user's submitted personal data, to the Server Communication Manager, through the use of the Load Balancer in order to manage the load and assign it to one of our two servers. Then the Server Communication Manager is responsible to send all the data to the Credentials Validity Manager that checks the validity of the inserted data, controlling if the password complies our restrictions. Then it sends to the Data Storage Manager a request to query the SafeStreets Cloud Database to discover if there is already or not an existing account associated with the username inserted by the user. After having received a response from the Database, the Data Storage Manager sends the response to the Credentials Validity Manager that checks it. In case there is already an existing account, the Credentials Validity Manager sends an error message to the Server Communication Manager that propagates it to the Load Balancer and then to the Client Communication Manager informing it about a malformed request or an already-in-use account. The Communication Manager sends the message to the View that shows it to the user.

In case all the verifications are ok the process is almost the same, except for the fact that the exchanged messages tell that everything has gone well.
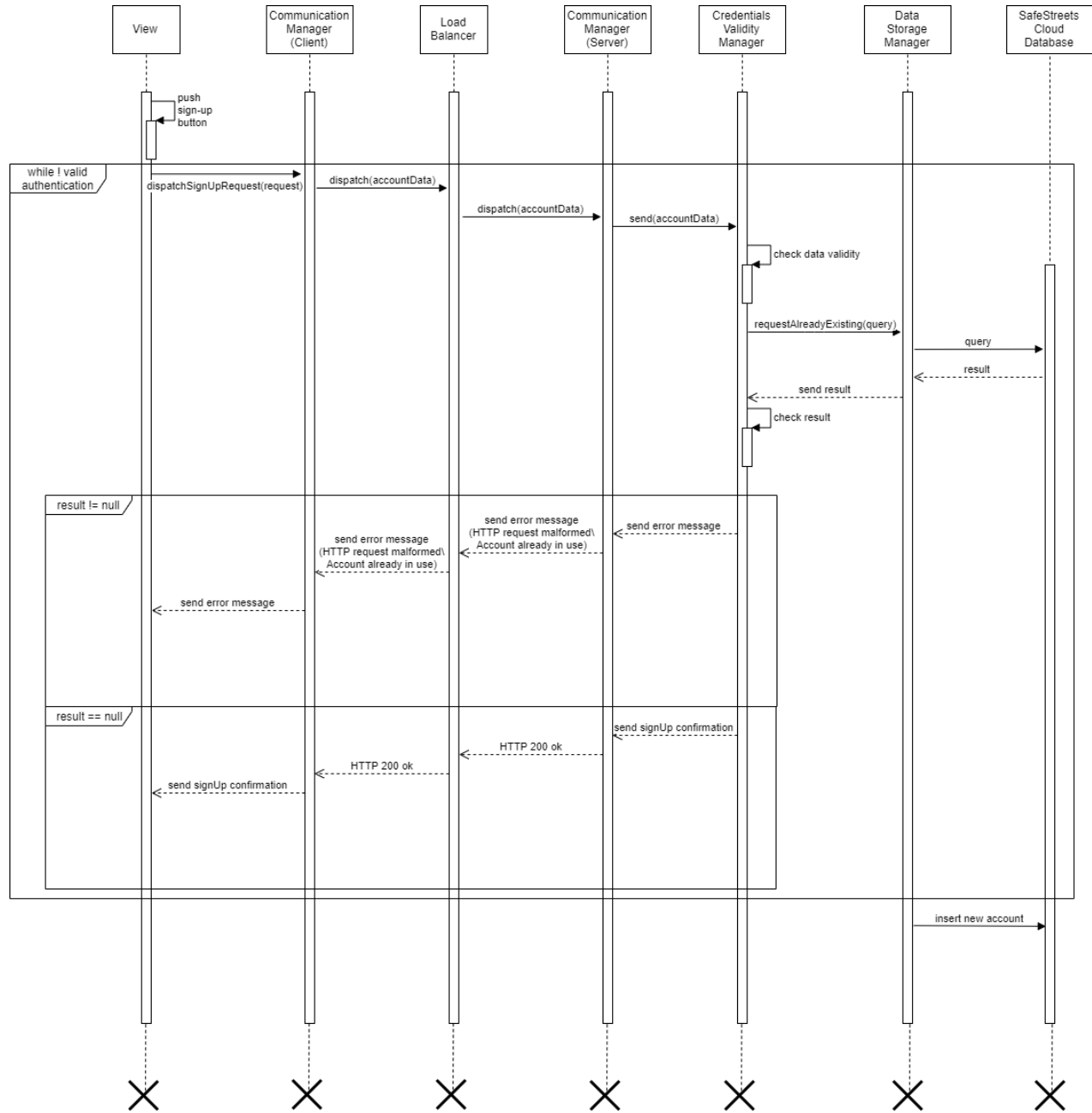
Figure 4: Deployment diagram

Login Runtime View

In this sequence diagram we explain which are the components involved and how they communicate with each other.
When the user pushes the login button, the view propagates the login request to the Client Communication Manager that is responsible to dispatch it to the Server Communication Manager, also thanks to the use of the Load Balancer that has the same function explained for the signing up.
Then the Server Communication Manager sends the request with the credentials to the Credentials Validity Manager, that elaborates a query to check if the credentials submitted by the user match an existing account in the database. The query is sent to the Data Storage Manager that dispatches it to the SafeStreets Cloud Database. When the Data Storage Manager receives an answer from the database, he sends the result to the Credentials Validity Manager that checks if it's valid or null.
If it's null it dispatches an error message to the Server Communication Manager that sends it to the client (through the Load Balancer) as a HTTP malformed request or wrong credentials.
Finally, when the Client Communication Manager receives the error message, it sends it to the View that displays it to the user.
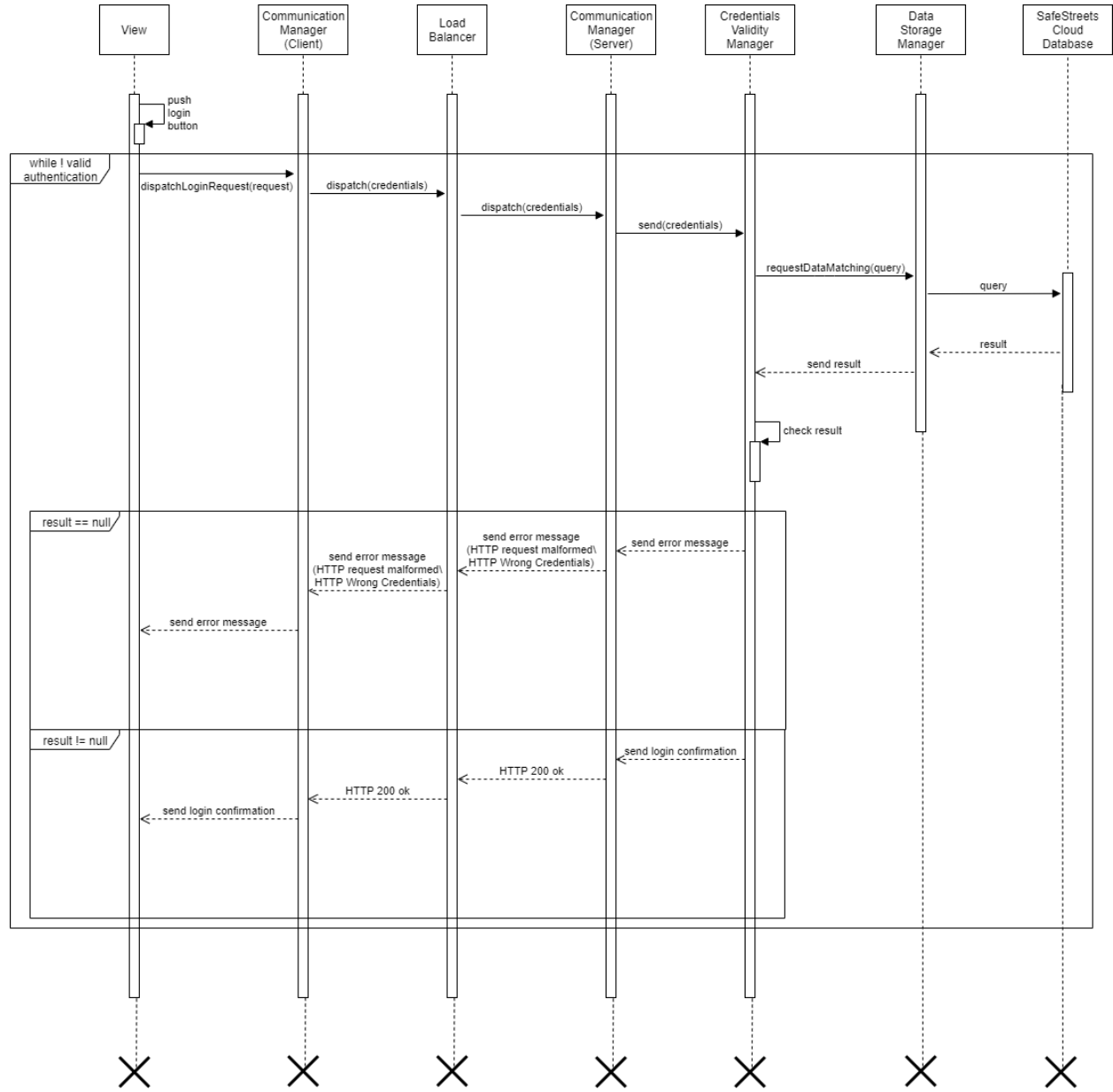
Figure 5: Deployment diagram

Report Runtime View

In this sequence diagram we explain the process of a Violation Report.
When the user submits all the violation information by pushing the report
button, the View sends the report to the Client Communication Manager. It
dispatches the report to the Load Balancer that sends it to the Server Com-
munication Manager. It sends the report to the Report Validity Manager that
checks the correctness of the report. To do this it checks if the violation type
belongs to one of the categories suggested by the municipality. After that, if
the report isn't correct it sends back to the Server Communication Manager an
error message. The Server Communication Manager will propagate it to the
client (through the Load Balancer) as invalid input. Then, the Client Com-
munication Manager sends the error message to the View, that shows it to the
user, that can reformulate the report and redo the same process in order to do
a new submission.
If the report is correct, the Report Validity Manager sends the confirmation
back to the Server Communication Manager that, with the same process of the
error communication, dispatches it to the client as HTTP 200 OK in order to
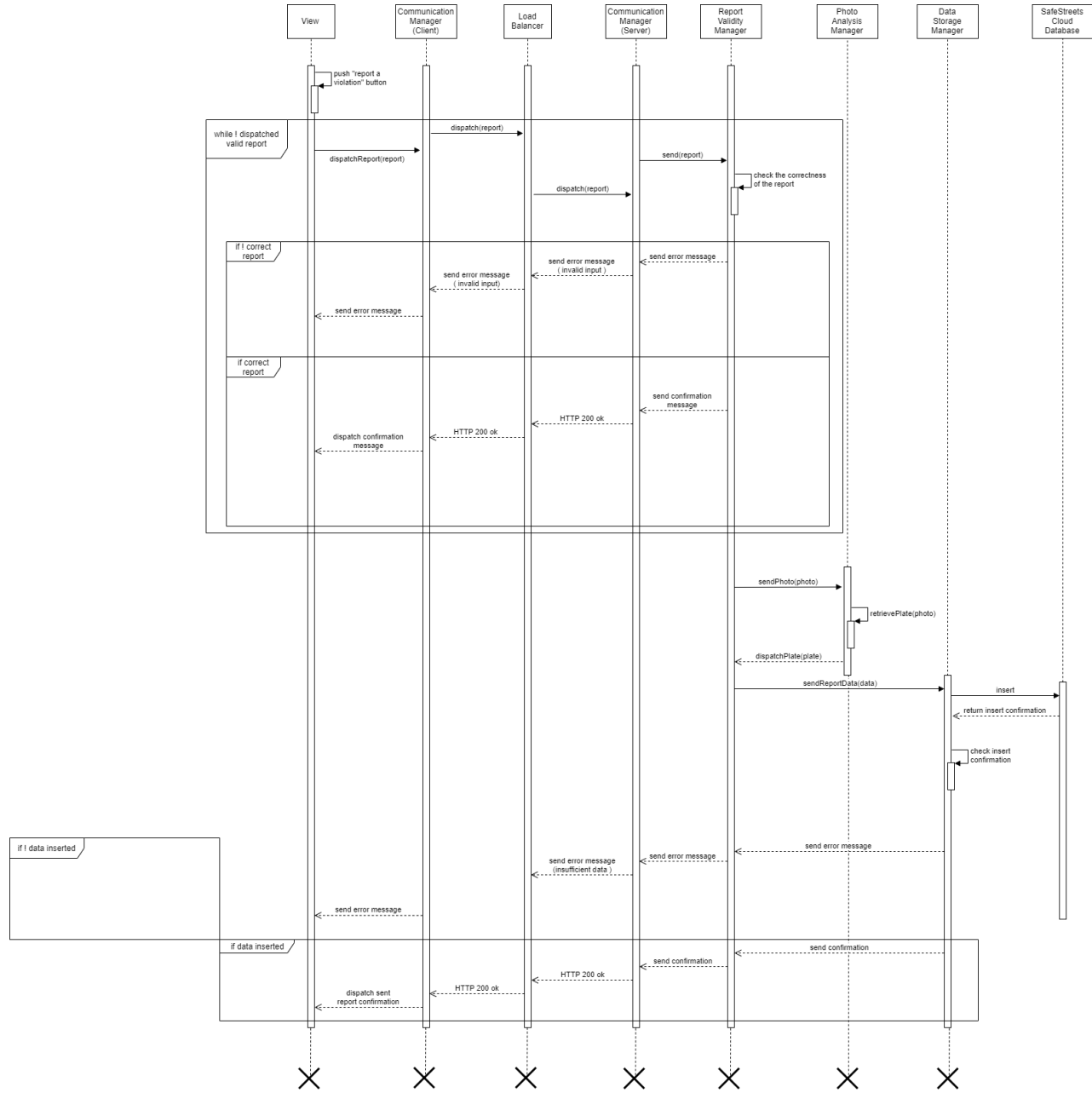finally make the View display the confirmation to the user.

Figure 6: Deployment diagram

17

2.5 Component interfaces

2.6 Selected architectural styles and patterns

We have selected an 4 tiers architecture. The entire system is a client-server application. The smartphone of the user implements only the presentation layer. This decision is due to the fact that we want the mobile application to be lightweight. We want the app to perform discreetly even with non-modern hardware phones. We want an application that limits as little as possible the hardware in a phone in terms of performance. The system data are stored in a database whereas the systems uses also another database which is the one of the municipality. The application logic is divided in two specialized clusters of application servers as already explained. We have chosen this to make the two interactions with the user (a user request or a user report) independent as much as possible.
The data layer is implemented in the databases of the system.
The decision of having different layers is to make the system as maintainable as possible. Every layer will offer only a simple interface to receive data and another one to send data to another layer and nothing more. In this way the implementation details are hidden. This means that when the system is modified it cannot happen that the modification of a layer causes a chain of successive modifications in other layers caused by the latter.

The system can be seen also as an example of MVC pattern:

- **Model:** The model is the component which store all the application data and check their integrity and so eventually modify the data structures to keep it. It is implemented in the databases (both the external and the internal one)

- **View:** The view is the software component that deals with the presentation layer. It only show data and information to the user. This component is implemented in the user device (as already explained before)

- **Controller:** The controller is the software component that contains and manages the application logic. In this system this logic layer must be implemented in the application servers.
  In our case, in particular, it is divided in two subcomponents implemented in different devices. One part concerns the managing of the requests and the other one concerns with the reports of the user. The first its implemented in the servers for the request and the latter in the servers for the reports.

2.7 Other design decisions

Here it is possible to find another design decision which are not about patterns and design styles which are the following:

- **Use of cloud databases:** We have chosen cloud databases to store the systems data because in this way it is possible to decrease or increase the resources according to the load of requests or reports. In this way it is possible to balance the storage space according to the load automatically. A solution based on proprietary databases would have been probably more efficient in terms of performances but surely more critical in terms of allocation or deallocation of resources. Furthermore, another important aspect of this choice is the fact that the physical maintenance of cloud database is entrusted to third parties (the owners of the physical devices which are the providers of the cloud service).

- **Use of clusters:** The use of clusters has been decide in order to try to make the system possibly more performing but above all, more fault tolerate. We have discarded the idea of having a single server instead of a cluster precisely because of this aspect. If the single server is down, then, the entire service will surely be inaccessible for the user.
The use of clusters is excellent as it makes the system more scalable. In fact, it is also easy to add additional servers possibly in the future.

- **Use of load balancers:** The use of load balancers has been decided to make the system more performing. In fact the throughput can be maximized and the response time minimized considerably.

- **Use of relational databases:** For this system we have chosen relational databases. On of the reason about this decision is the fact that the data that the system manages and stores is well defined and structured. We know that sometimes operations like "join" between different tables can affect the system and makes it slower. However, we think that this aspect will not cause relevant problems in our system. In fact the performance constraints are large enough to not cause problems in our opinion. Furthermore, relational databases have been used for several years and they are used also nowadays for a lot of applications. Because of this, we have a strong theoretical background about them. This is not always true for not relational databases.

3   User interface design

Here there will be shown the user interfaces with the help of UX diagrams. In particular there is shown the flow that the user will follow to navigate inside the application. Some aspects concerning the user interfaces design have been already treated in the "Requirement Analysis and Specification Document",if the reader wants to understand in a better and more detailed way, he/she is strongly suggested to refer also to it. In this document it is described only the flow that the user must follow to use the application. UX diagrams:
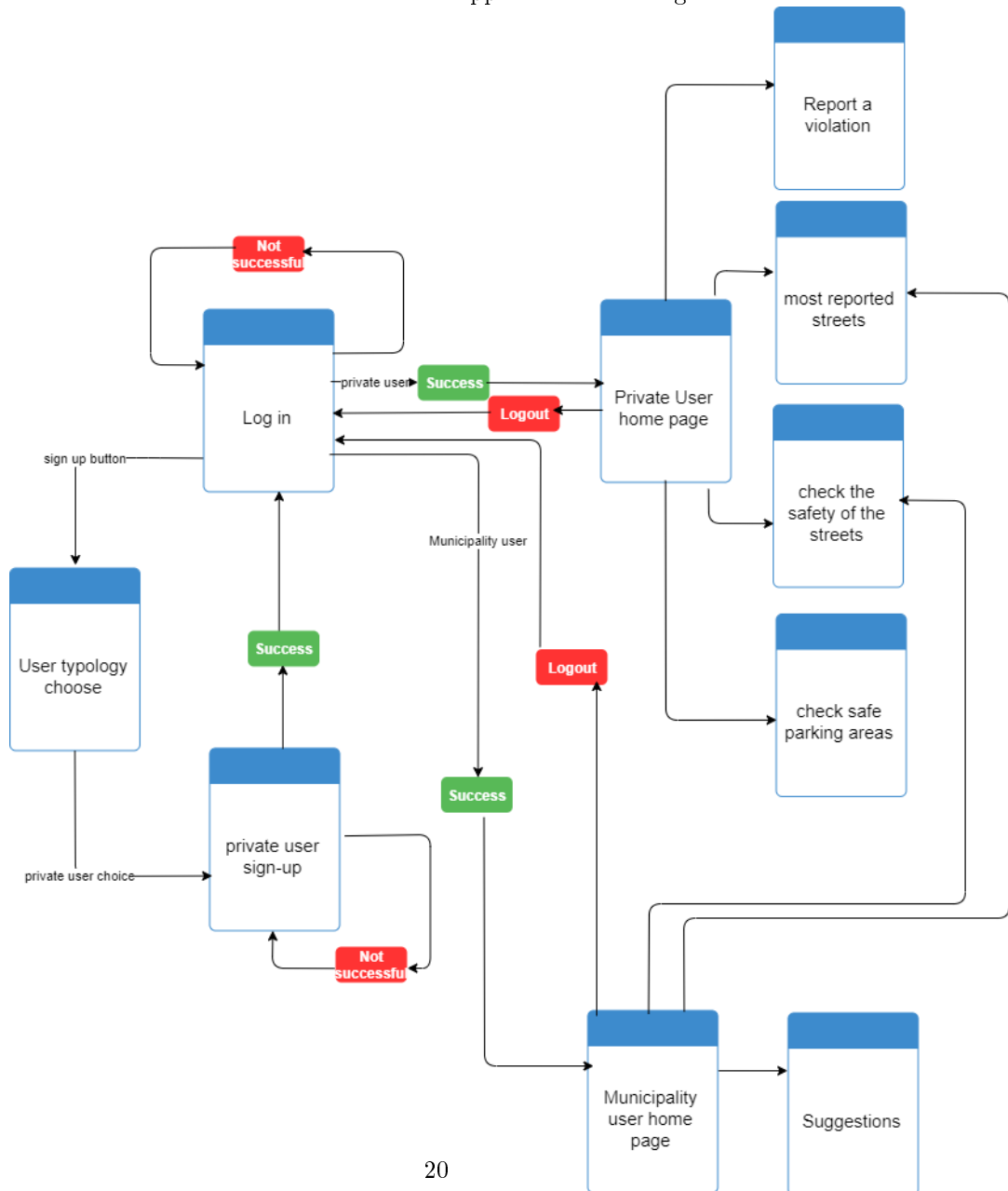
Figure 7: UX diagram

4   Requirements Traceability

Here it is possible to find an explanation about how the requirements in the rasd map to the to the design elements that can be found in this document.

Component	Requirements (RASD)

View	
	• [R1] ::The system must allow the user to take a picture.
	• [R2] :The system must allow the user to indicate the street where the violation occurred.
	• [R3] :The system must allow the user to indicate the hour and the date when the violation occurred.
	• [R4] :The system must allow the user to insert a brief description of the violation.
	• [R5] : The system can localize his/her position automatically as alternative to the manual insertion.
	• [R6] :The system must allow the user to finally submit the report.
	• [R9] : The pictures must be taken only through the app camera and not imported from the device gallery in order to take photos in real time.
	• [R14 :]The system must allow the user to insert the city in which the violation occurred.
	• [R15] :The system must allow the user to insert a number representing the radius within which the system must provide the condition of the streets.
	• [R23] : The system must allow the user to choose between municipality account or private user one.
	• [R27]:The system must allow the user to recover his/her password if he/she has forgotten it.
	• [R28] : The system must allow the municipality users and the private ones to access the corresponding Home Page after the log-in.
	• [R29] :The system must allow the user to logout from the platform in every moment by the exit button placed in the Home Page.

User requests manager	• boh
Municipality Suggestions Requests manager	• boh
Data Mining manager	• [R17] : : The system must run a data mining algorithm to update the information (contained in the database) about the violations and the streets safety. • [R19] : The system must retrieve the information about violations from the users' reports. • [R20] : The system must use both data received by the users and by the municipality. • [R21] :The system must run a data mining algorithm to update the database of the system. It must update the suggestion for the municipality.
Report Validity manager	• [R11] :The system must check if the reported street exists in reality. • [R12] : The system must allow the user to insert the street around which he/she wants to check the safety.

Credentials validity manager Manager	[R8] : The chosen username must identify the private user uniquely. [R24] :The system must allow the municipality to sign-up if and only if they insert a valid ID code. [R25] : The system must allow the user (both private user and the municipality) to sign-up if and only if he/she inserts a username that is not already used. [R26]: The system must allow the user (both private user and the municipality) to sign-up if and only if he/she inserts a password that complies the security policy. [R27]:The system must allow the user to recover his/her password if he/she has forgotten it. [R29] :The system must allow the user to log-out from the platform in every moment by the exit button placed in the Home Page. [R28] : The system must allow the municipality users and the private ones to access the corresponding Home Page after the log-in. [R30]:The system must check that the chosen username (chosen during the sign-up phase) must be unique.
Data manager	[R5] :The system can localize his/her position automatically as alternative to the manual insertion. [R13] :The system must be able to retrieve users' position data from the GPS service of their smartphones.

Photo Analysis Manager	• [*R7*] : The system must run an algorithm to retrieve the license plate from the violation picture.
Data Storage manager	• [*R10*] : The system must store the report if it is correct. • [*R18*]: The system must retrieve the information from its database • [*R22*]: The system must retrieve municipality suggestions from its database.
Result Handler	• [*R16*]: The system must report the safest streets in a map form. These streets must be highlighted on the map.