POLITECNICO
MILANO 1863

Computer science and engineering
Software engineering 2 - Project 2019/2020

SafeStreets

Requirement Analysis and Specification Document

Deliverable:	DD
Title:	Design Document
Authors:	Maldini Pietro , Paone Angelo
Version:	1.0
Date:	9-December-2019
Download page:	https://github.com/pm390/MaldiniPaone
Copyright:	Copyright © 2019, Maldini Pietro, Paone Angelo – All rights reserved

Contents

Table of Contents **3**

1 Introduction **4**
1.1 Purpose 4
1.2 Scope 4
1.2.1 Description of the given problem 4
1.3 Definitions, Acronyms, Abbreviations 4
1.3.1 Definitions 4
1.3.2 Acronyms 5
1.3.3 Abbreviations 5
1.4 Revision History 5
1.5 Reference Documents 5
1.6 Document Structure 5

2 Architectural Design **7**
2.1 Overview: High-level component and their interaction 7
2.2 Component view 8
2.3 Deployment view 11
2.4 Runtime view 12
2.5 Component interfaces 12
2.6 Selected architectural styles and patterns 12
2.7 Other design decisions 13

3 User Interface Design **14**
3.1 UX diagram 14

4 Requirements Traceability **18**

5 Implementation, Integration and Test Plan **19**

6 Effort Spent **21**

7 References **22**
7.1 lorem ipsum 22

1 Introduction

1.1 Purpose

The purpose of this document is going more in the technical details than the RASD concerning SafeStreets application.
The Design Document gives more details about the design giving guidelines over the overall architecture of the system. This document aims to identify the core design choices for developing the system:

- The high level architecture
- The components and their interfaces
- The design patterns
- The Interaction between the components
- Planning for implementation, integration and testing of the system

A mapping of the requirements to the architecture's components is also given//TODO chiudere questa frase

1.2 Scope

1.2.1 Description of the given problem

SafeStreets is a crowd-sourced application whose intention is to notify the authorities when traffic violations occur. Citizens, thanks to the system, will be able to send information about violations to the authorities who will take actions against them. In this way, the service provided by the authorities can be improved because they will receive notifications through the app. The sources of notifications are the Citizens who take photos of violations and send them to the authorities through the application. The information provided by users are integrated with other suitable information and are stored by the service. The system also runs an algorithm to read the license plate of the vehicle in the photos. All collected data can be seen by Citizens and authorities to find which streets are the safest. Users can have different levels of visibility: authorities must be able to know the license plates of vehicles in the photos, while normal users can only see data in the form of statistics. Moreover, data are sent to the municipal district so that important information can be extracted through statistics in order to make decisions to improve the safety of the area. Finally, the system will have to be easy to use, reliable and highly scalable to fit perfectly with the mutable context in which it will be used.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Violation: parking violations which can be notified by Citizens to authorities
- Report: Notification sent by Citizens to the system
- Mapping System: external software that provides maps and directions to reach the position of a violation
- Licence plate Recognition Algorithm: calculation process that identifies the alphanumeric number on license plate
- Spam: a series of messages that are undesired
- App: application software

- Blocked: means that the account is banned for a given period
- Metadata: data about a violation. Position , date, time and the username of Citizen.
- Assignment : Work Request for authorities generated upon the receiving of a notification made by Citizens.

1.3.2 Acronyms

- RASD: Requirement Analysis and Specification Document.
- DD: Design Document
- API: Application Programming Interface
- GPS: Global positioning system
- HTTP: HyperText Transfer Protocol
- HTTPS: HyperText Transfer Protocol over Secure Socket Layers
- UML: Unified Modeling Language
- JSON: JavaScript Object Notation
- UI: User Iterface
- SQL:Structured Query Language

1.3.3 Abbreviations

Rn : n-th requirement.

1.4 Revision History

- DDv1.0 delivered on 9/12/2019

1.5 Reference Documents

- RASD versione 1
- Specification Document: “Assignments AA 2019-2020.pdf”.
- IEEE Standard for Information Technology—Systems Design—Software Design Descriptions

1.6 Document Structure

This chapter debates about contents and structure of DD, indeed this document is divided in seven different sections:

1. The Introduction provides a general appearance of the systems defining which are the goals to reach, describing the problem.
2. The Architectural Design
3. The User Interface Design
4. The Requirements Traceability

5. The implementation, integration and test plan
6. The Effort Spent
7. The Reference

2 Architectural Design

2.1 Overview: High-level component and their interaction

The SafeStreets application is a distributed application with three logic software layers. The Presentation layer wich manages the interaction of the user with the system and is responsible of maintaining the GUI, the Application layer which handles the logic of the application and its functionalities the last layer is the Data access layer which manages the accesses to the database and allows the separation of concerns between business logic and data. This so called three-tier architecture is thought to be divided on three different hardware layers that represents a group of machines so that every logic layer has a dedicated hardware. This architecture makes it possible to guarantee scalability and flexibility of the system and also lighten the computational load of the server splitting it in two different nodes. This architecture improves also security of data since users can't access directly data layer but must communicate with application layer that will retrieve only the necessary data. Citizens accesses the system using their mobile phone app , the app communicates with the application layer to send reports and get statistics , authorities can access the application in the same wway as citizens but they can also receive assignments, see reports and also take on assignments and finish them. The server of the application layer sends push notifications in asynchronous way to authorities to warn them about violations in the area. Municipality and System manager communicates with application layer to add new municipality and authorities to the system but also to retrieve statistics and to read suggestions for improving safety on streets. The Application layer communicates with data access layer synchronously to retrieve information and asynchronously to store information about violations. This kind of architecture allows the server nodes to be replicated to improve the system scalability. Replicating nodes adds the need for a new component in the architecture , the load balancer, this component is responsible for distributing the working load among the replicated nodes, this approach also increases the fault tolerance of the service since a fault in a node doesn't affect the service availability. A fault in the system may increase the work load for the other nodes, the number of replications should be decided considering also this possibility to avoid the creation of a bottleneck due to a fault of the server. To assure security of data managed by the system the server has two firewalls to check packages exchanged with users. Both filters packages incoming and going out of the application logic one towards user devices and the other towards the data access layer.

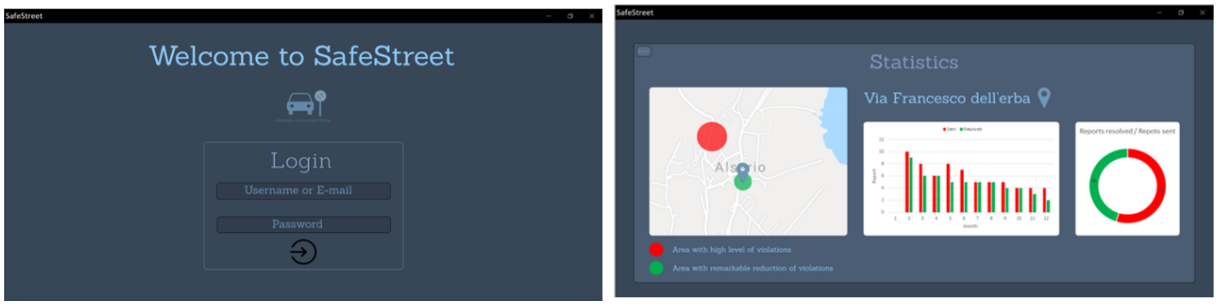

Figure 1: PlaceHolderTODO immagine coi firewall

Citizen and authorities are provided of mobile applications to access the functionalities offered by safestreets. Municipalities and system managers communicates with safestreets using a web application. The Application layer is divided in two parts, one which sends static data (HTML,CSS,JAVASCRIPT) to web application and another part which communicates with database and provides users dynamic contents. Our system uses only few caching capabilities because lot of data is dynamic and chages continuosly. The only informations we cache are informations of the violations when an authority takes the corresponding assignment ,citizens have in cache the list of reports they have done and can choose how long that list is kept in memory. Safestreets uses a relational database to store data about users and violations , the technology used in municipalities' databases depends on the municipality, a different

interface may be used to communicate with each of them. For scaling purpouse a non relational database could be used in later releases to collect data from different databases with different structures and then uses mining techniques to find useful information and pattern in recurrent violations and information useful for building statistics may be queried on a regular basis and stored in a location which is faster to access for our system , reducing the number of queries to be done to external databases.

2.2 Component view

In this section we present the components of SafeStreets we start from a generic point of view showing the main parts which composes our system and then we divide the system in smaller parts to analyse the behaviour of the single components and their subcomponents.

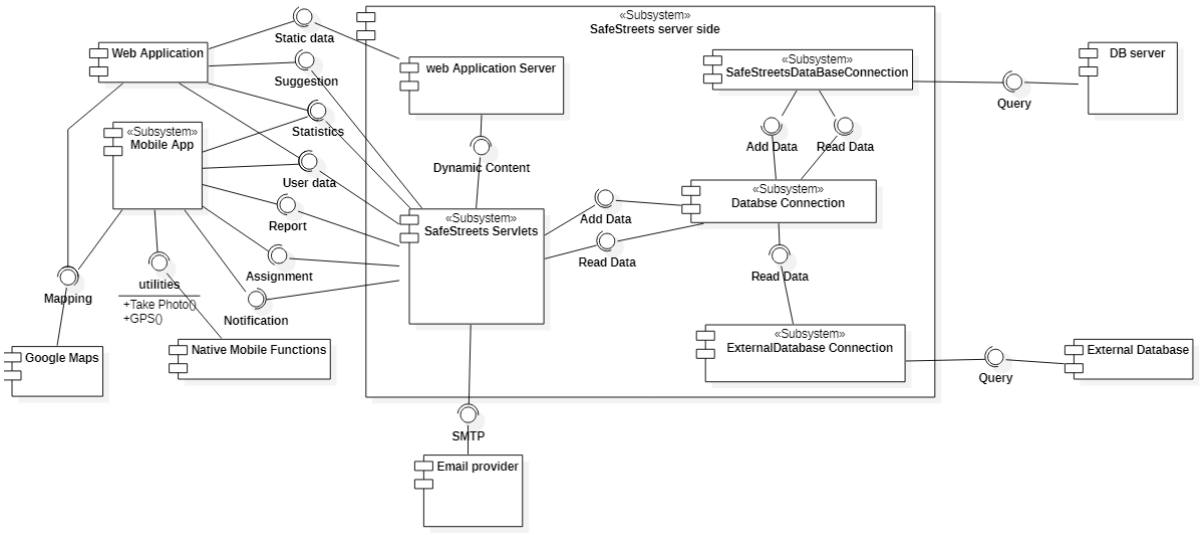

Figure 2: PlaceHolder TODO generic component diagram

In this diagram we represent a high level logic view on Subsystems and components. We show the interaction between the server and mobile app,web application, email provider and databases. The server is composed of five SubSystems and components:

- SafeStreets Servlets: this contains all the Servlets which allows the user to communicate with the server.To retrieve and save data this component communciates with the Database Connection subsystem. All response to the user are sent in JSON format.

- Web Application Server: this component provides the web application the static data of the application (HTML,CSS,JAVASCRIPT), this part is separated from SafeStreets Servlets for the sake of separation of concerns, dynamic content and static contents are different so different components should get request to retrieve them.

- DataBase Connection: this component acts as a facade which separates servlets from data access logic. This component communicates with SafeStreets servlets providing them the functionalities to access data. It communicates to Connection subsystems which communicates with SafeStreets Database and with External Databases to get data which is required by users.

- SafeStreesDataBaseConnection: this component communicates with SafeStrees database executing query to update and read data from it and provide data to DataBase connection component.

- ExternalDatabaseConnection: this component communicates with External databases executing query to read data from it and provide them to DataBase connection component.

SafeStreets Servlets communicates using SMTP protocol to send email to users who creates new account or forgets credentials. It also communicates with Mobile App and Web application providing them Access to their acoount data, the possibility to make reports for Citizens , to take assignments for the Authorities, to read suggestions for Municipalities and to see statistics for every type of users and visitors. The server also communicates with databases to retrieve and save data about violations, users and accidents. Mobile application and web application both communicates with Google maps API to get maps to show to users. Mobile app also needs to communicate with functionalities given by the device such as taking Photos and retrieving position using GPS.

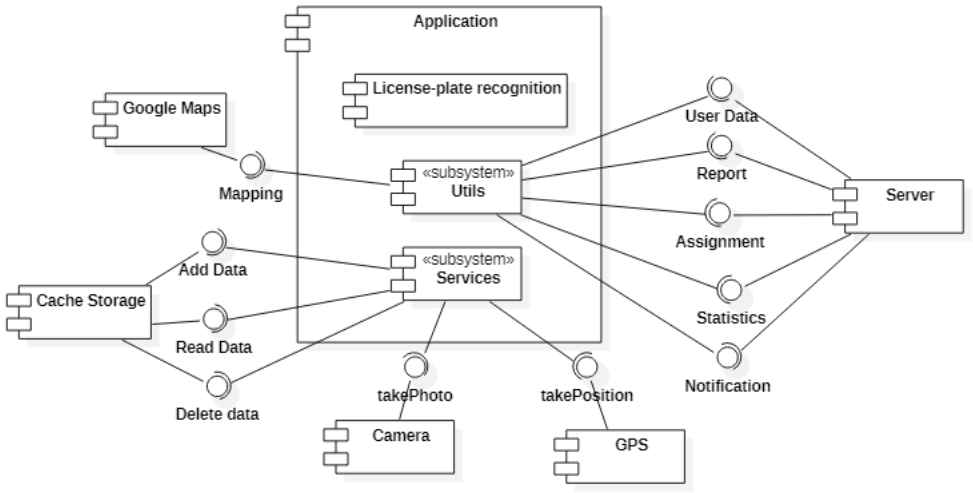

Figure 3: PlaceHolder TODO mobile app component diagram

In this image we show how Mobile Apllication Subsystem can be seen in more detail. There are two main components :

- Utils: this component takes care of the communication of the application with SafeStreets Server and Google Maps mapping API.

- Services: this component communicates with Mobile phone's services which are needed for our application. Those components are GPS which is used to retrieve user position, the phone camera user to take photos of the violations and cache storage used to save locally some useful informations both for Citizens and Authorities.

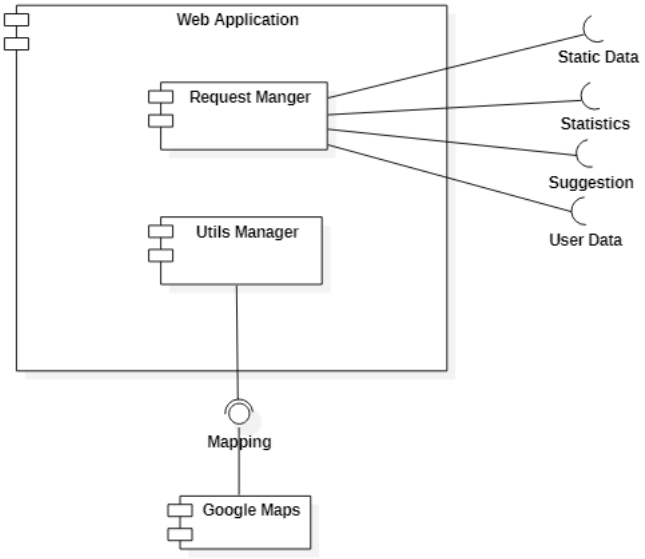

Figure 4: PlaceHolder TODO web application component diagram

Web Application works in a similar way to the mobile app. It must retrieve static data from webServer differently from mobile app.

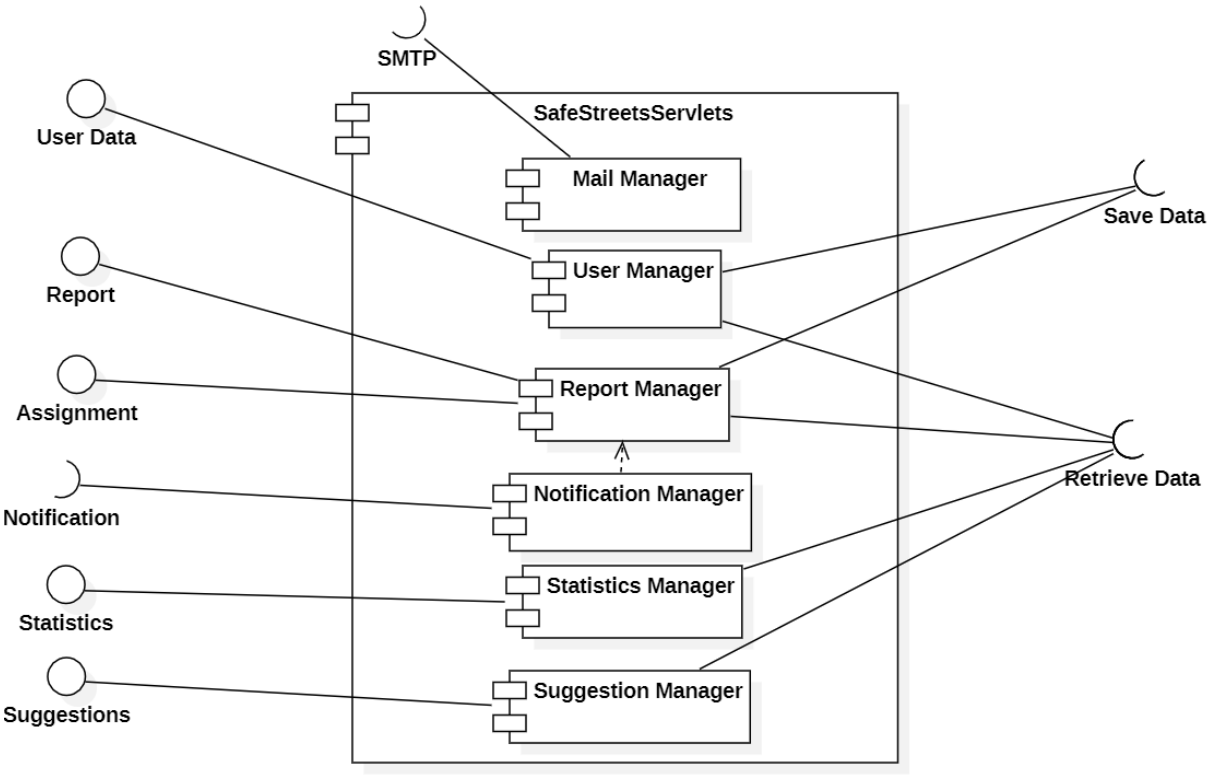

Figure 5: PlaceHolder TODO Servlets compent show functionalities to the users

SafeStreets Servlets component is composed of 5 Managers:

- Mail Manager: this component allows the system to send email to Users when they create account or they modify their credentials. This component is associated with user manager which informs it when an account creation or modification is successful.
- User Manager: this component allows users to create handle their account informations.They can create a new account, modify their data and Login. To allow thees functionalities this component must be able to both save data and retrieve data from database.
- Report Manager: this component allows citizens to create reports and manage them, and also allows Authorities to take assignments and terminate them. It communicates to the notification manager when new Assignments are created.
- Notification Manager:This component sends notificiations to authorities about new violations. This component requires that the Mobile application opens a websocket to communicate.
- Statistics Manager: This component retrieves data to make statistics requested by Users and visitors.
- Sugggestion Manager: This component retrieves suggestion requested by Municipalities.

All components takes data and saves data using interfaces exposed by DataBase Connection componet

2.3 Deployment view

In the following image the deployment diagram for SafeStreets shows the distribution of the system components and the different deployment nodes. We used a different color to represent External Database servers to show that the node is different, that's because the system should not implement these node but should only communicate with it. Google Maps API and Email Provider are not shown in this diagram to simplify it and because their interaction with our system are described in component view.

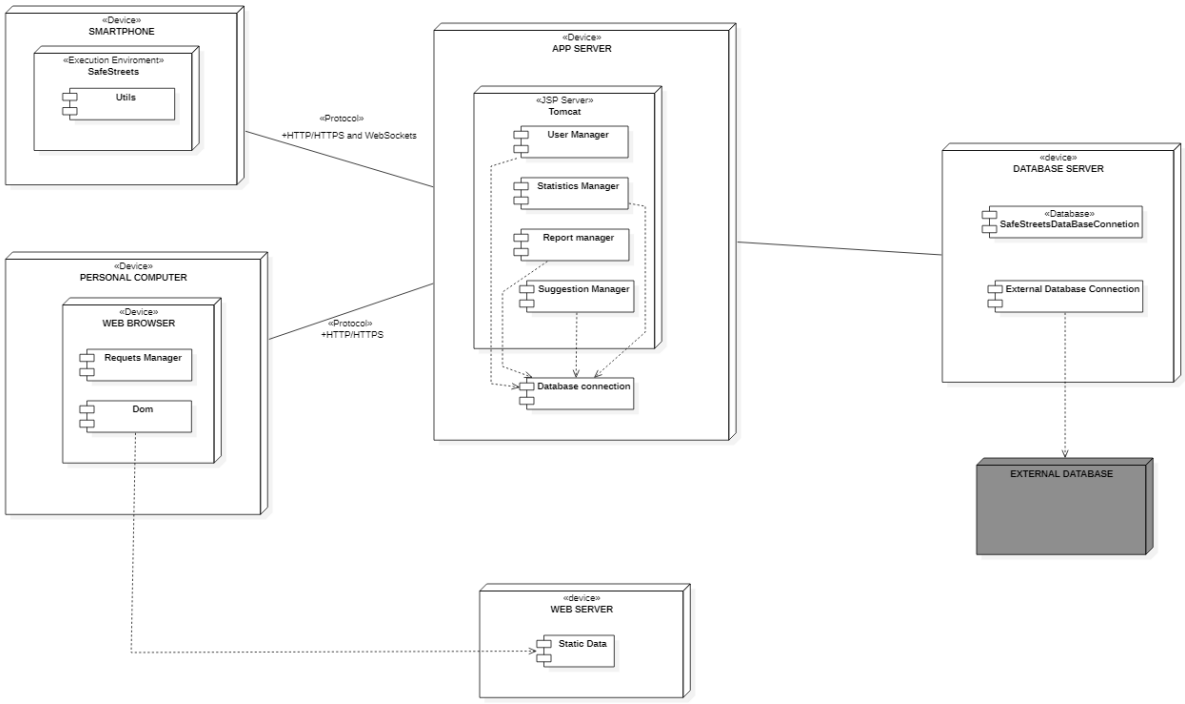

Figure 6: PlaceHolder TODO Deployment diagram

This Diagram is divided in three sections to show clearly the separation of the different layers of the Application:

- Presentation : this layer containts the presentation logic. Users to access data must be provided with mobile application for Citizens and Authorities, with web application accessible by a web browser for municipalities and system managers. Mobile application must be available for both Android and iOS but not just newer versions to make the application the more accessible as possible, for the same reason web application should be compatible with major desktop web Browsers : Google Chrome, Mozzilla Firefox, Safari and Microsoft Edge(at the time this document is being written). Communication happens using HTTP and HTTPS for both devices while it also happens through WebSockets for mobile application push notifications
- BusinessLogic: this layer is divided in 2 different nodes types. Web Server communicates with webapplication and sends it static data needed to render static components(HTML,CSS,JS). App Server instead allows users of both type of application to access dynamic data acting as an intermediary to separate presentation and data but also to control access to data.
- Data Access: This layer excecutes a relational DBMS and provides functionalities to the Business logic to access data requested by users. We show in this layer also the external databases which are another important source of data for SafeStrees. regarding this component at an initial phase the server will access using the provided interfaces those external databases. When the application will grow a different approach must be implemented to preserve performances by reducing requests to external systems. To reduce this burden a non relational Database internal to safeStreets may be used to store information from external databases using those as sources to create datamarts and use technologies optimized for Big Data. Using this approach will add a recurring activity to be performed which is mining this non relational Database to build suggestions for municipalities.

2.4 Runtime view

To communicate the various internal parts of the S2B use HTTP and HTTPS protocol. No specific communication interface must be implemented.

2.5 Component interfaces

2.6 Selected architectural styles and patterns

Multiple Servlet server
We choose to use a server which provides multiple servlets to create different access points to the server for different requests. This choice allows us to create separation of concerns SafeStreets Server which is composed of components with few functionalities and a reduced amount of interactions between them. This approach allows the implementation and testing of components in parallel. The only components which must be developed in advance and tested are Database and database access components. Once those components are completed and their functionalities are tested it is possibile to develop different functionalities of the server in parallel, the components must be tested and then integrated with database access components. This approach allows also to make the system scalable , the different functionalities may be separated in different harware and a routing device may be used to send the request to the right machine which can answer to the request. The system can also be expanded in a really simple way since to add a new functionality a new independent servelt may be added to handle it and then a new manager to handle the request and send it correctly to the database connection components in some cases an already existing component may be expanded to handle it. Since different functions of the server are separated in case the system needs to be checked because some functionalities aren't working in the correct way it is possible to pinpoint the parts of the system which aren't correctly working and fix them. For this approach to work it is really important that the Database access components are working correctly since

a bug in that component will affect the overall application making later application maintanance more costly, it is really important that this part is tested in all its functionalities and documentation must be as clear as possible to allow future maintenance and expansions the more simple as possible. TODO check grammar

Three Tier Client-Server

The separation of the system in three different layers allows to makes the system more flexible and reusable. Together with the multiple Servlet approach it allows to add functionalities and modifing them without affecing the entire application. This also allows the separation of the presentation of data to the user from data access using the application layer to block access to informations that a user should not access(I.e. a citizen should not access photos of violations or assignment lists)

2.7 Other design decisions

Thin client

A thin client is characterized by the fact that it's only function is to communicate with server to retrieve data and concentrates on presentation of data to the user.

This design choice allows us to separate the business logic from the presentation. However our Mobile application has some business logic in it so it is not purely a thin client. This choice is made to reduce drastically draining computational resources from server. The business logic in the App is the Licence plate recognition algorithm which checks if at least one photo provided by the user contains a valid licence plate, doing such a control server side would have needed to send all the photos to the server to check them and in case of failure notify the application of it and ask the user to re-take photos and send them again to the server, considering also that those kind of algorithm can take lot of time to be executed. To clarify this statement this example may help to understand our choice: an execution time of a millisecond on a mobile phone may be seen as a bit of lag by the user but if the algorithm runs on the server that millisecond of delay which must be mutiplied by the amount of users sending photos and the number of photos sent by them can result in a delay of seconds in the performances of application server Nowadays it is important to consider a lot those kind of delay since users can access any kind of application and user experience while using an application is crutial, a delay of just a couple of seconds can make a user uninstall completely an app and also discourage other people from using it.

3 User Interface Design

3.1 UX diagram

The mockups of the application were already exposed in the RASD document.
We present UX diagrams showing the flow all type of users interacts with the system. There are repetitions in diagrams to show how mobile application users have lot of common interaction but also web app users have lot of interactions in common. Every user when application starts up can login or ask to retrieve his/her lost login credentials. All user can also access statistics and modify their account's data and credentials. The differences are in some screen available only for specific type of users. Citizen can create new account and make reports, authorities can take assignments and terminate them, municipality can register authorities but can also see suggestions , system manager and municipalities can register new municipalities.

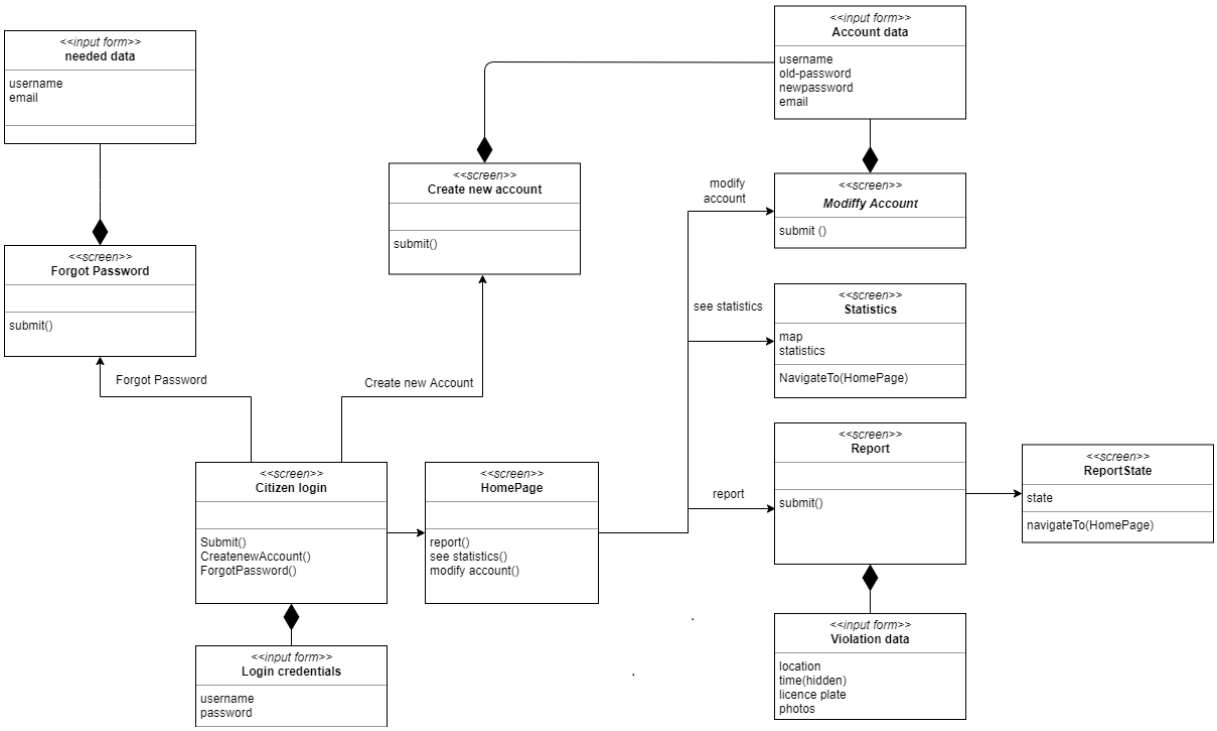

Figure 7: TODO

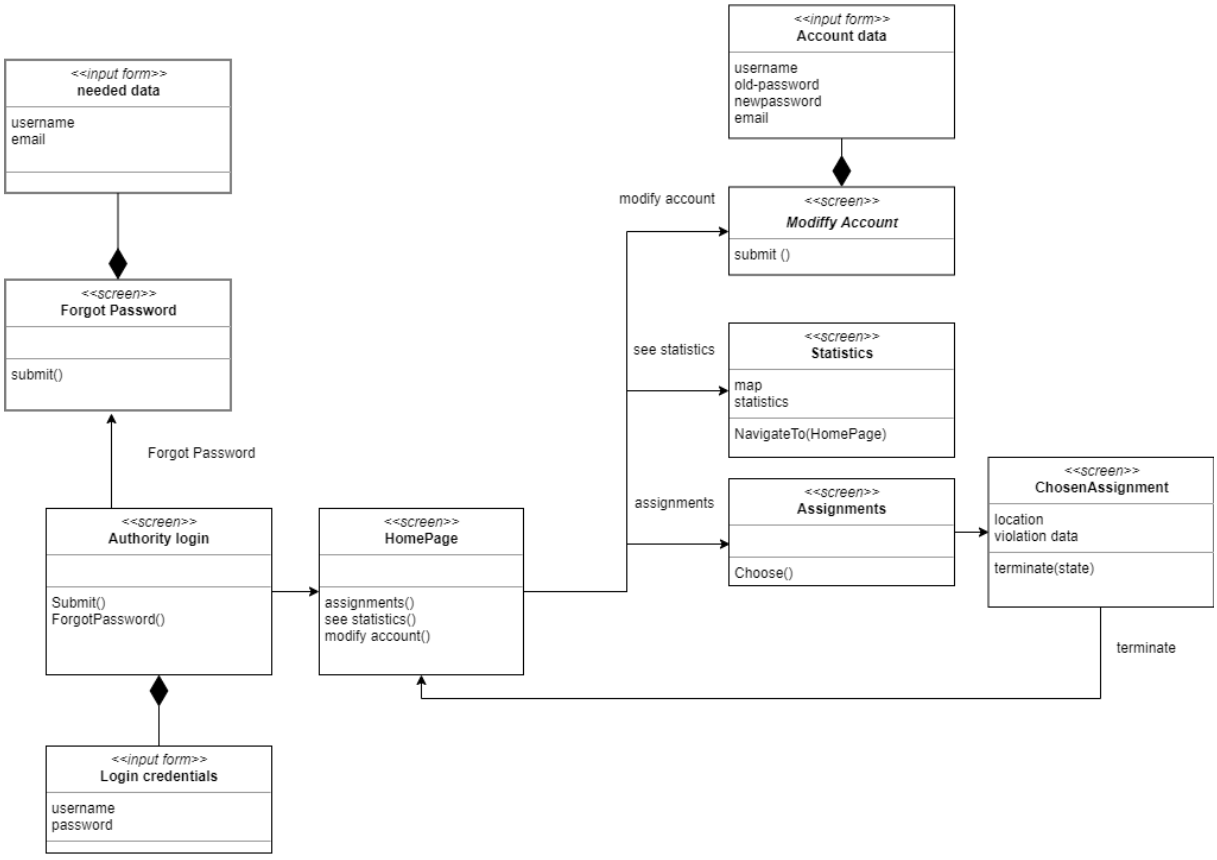

Figure 8: TODO

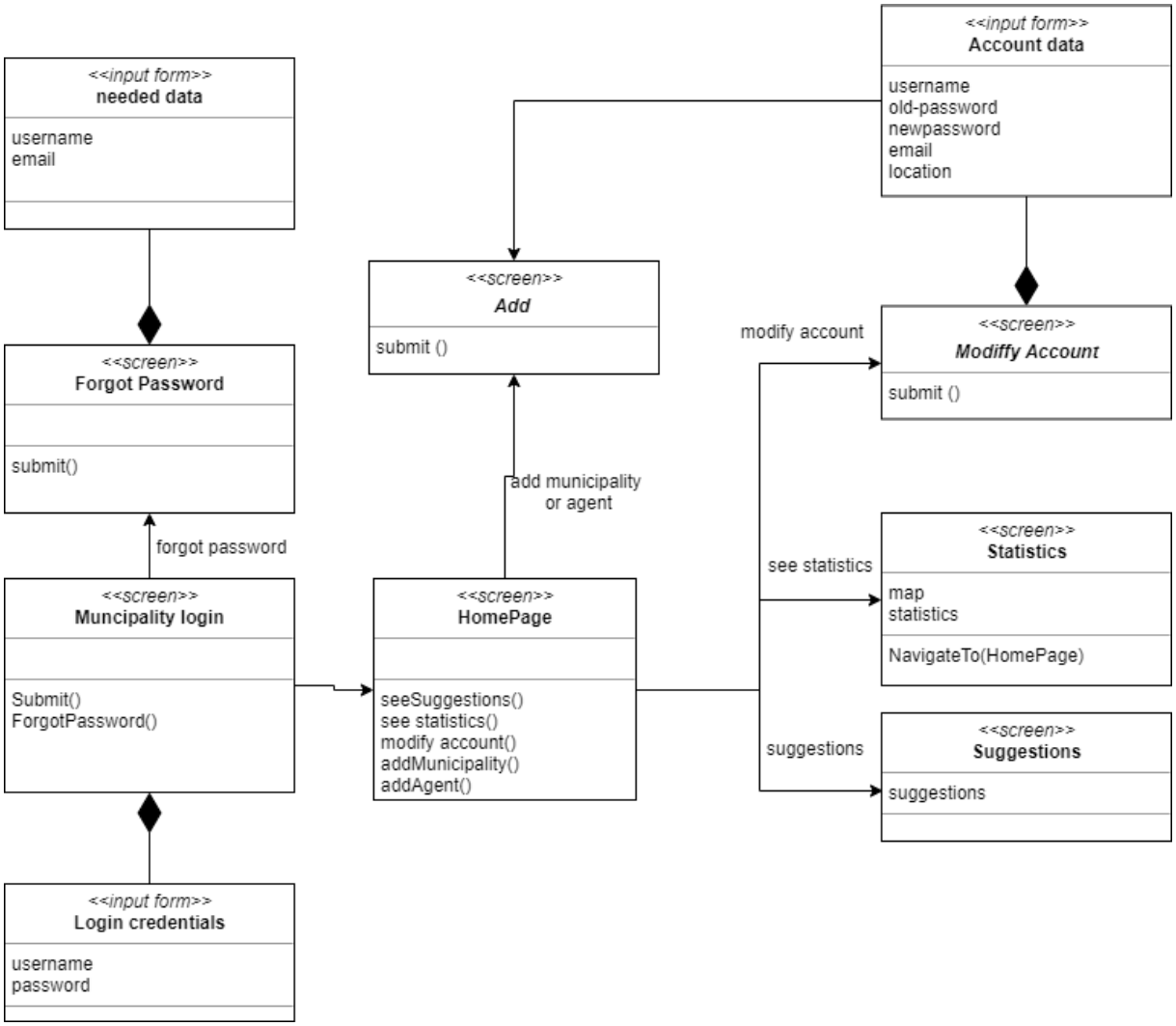

Figure 9: TODO

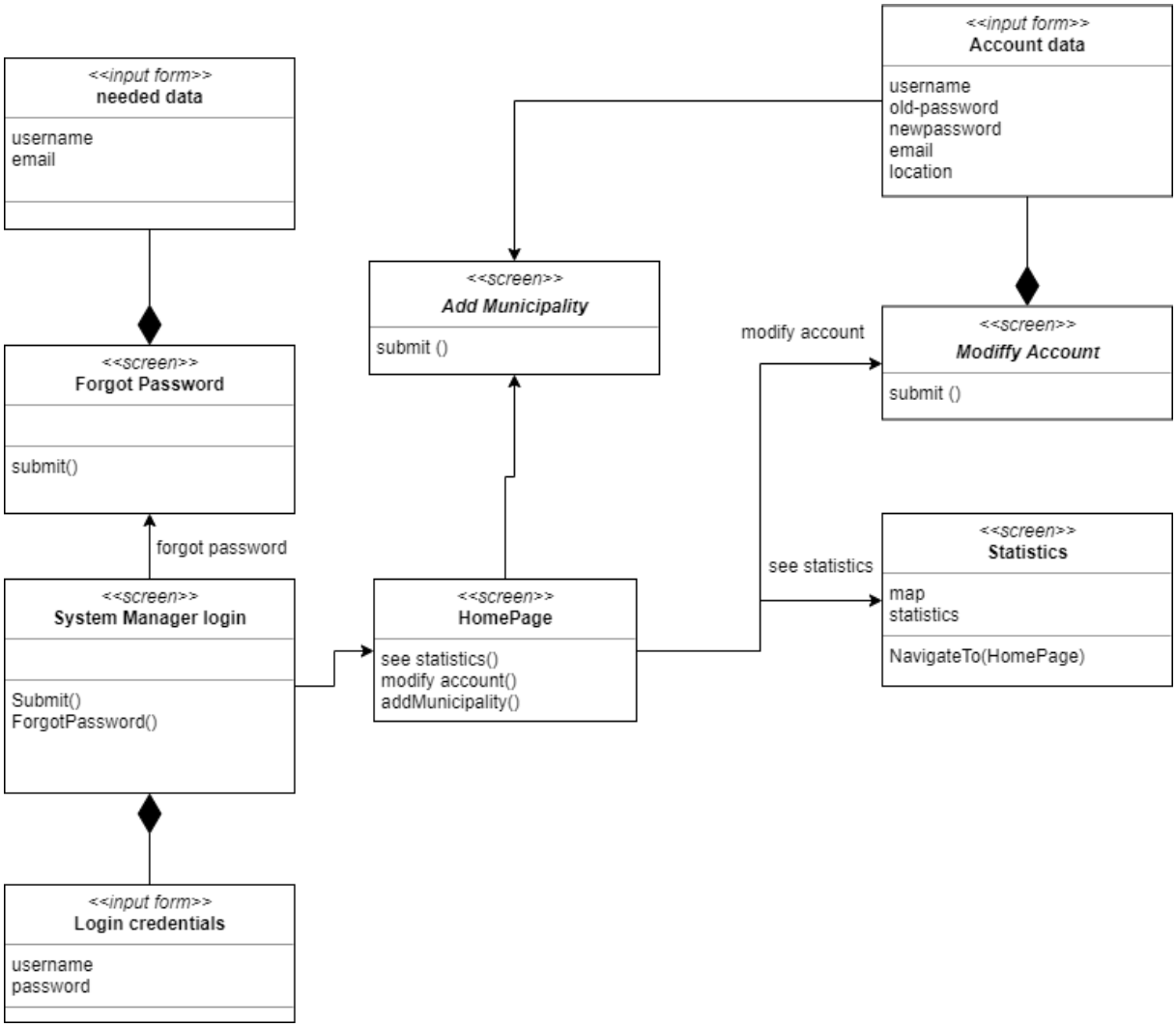

Figure 10: TODO

4 Requirements Traceability

The DD is thought to guarantee that the components of the system are able to enforce the requirements presented in the RASD.
In this Chapter we show which components communicates with each other to enforce the fulfillment of the requirements. A small description of the interactions is here to show how the communication should be performed.

- R1) Authorities' location must be known by the system when they are in service : Authority Position Servlet gets data about authorities' location and through User Manager, the application sends data to the server everytime the authority moves from the previous locationn of at least more than 200 meters TODO think about this 200.
- R2) Data relative to the violation sent by the user must be clear : Application checks if at least one photo contains a licence plate clearly visible and recognizable by a Licence Plate Recognizing Algorithm, other usefull information like position are gotten from mobile phone's system, if not possible the user can insert position data but the inserted position validity will be checked. Time is added server side. TODO think if the requirement is well written
- R3) The right authorities are notified about violations : Notification Manager gets Authority position from database. The position is updated by User Manager which communicates with the DataBase Connection package trough DataAccess Facade.
- R4) Authority must be able to provide the system how the assignment finished: resolved, no intervention needed when arrived, false report: Application after an authority accepted an assignment takes him/her to a screen where the assignment can be terminated. This Screen allows user to communicate with the Assignment Servlet which notifies the Report Manager.
- R5) The system must make data available when asked: Data Access Facade allows all manager classes in the server package to access data which can be showed to users. User may query data making calls to the servlets
- R6)Data and statistics are always updated when an event happens: In Database Connection package Data Collector is responsible for updating data about violations in the database but also to combine data from Safestreets database and municipalities' databases to build statistics
- R7) System Manager must fill correctly the form with Municipality's data : web app allows user to insert all the needed data and doesn't allow to send data if all the needed data are not correctly inserted. For further security also Authority and municipality registration Servlet checks if all needed data are inserted.
- R8) A visitor must be able to begin sign up process in the SafeStreets App: //TODO
- R9) When the creation is successful the system must notify the Visitor ://TODO
- R10) A citizen must be able to correctly report violations details ://TODO think about R2 is needed?
- R11) Authority must be able to correctly finish their assignments and specify how the violation was resolved and the correctness or non-correctness of the report://TODO think about R4 is needed?

5 Implementation, Integration and Test Plan

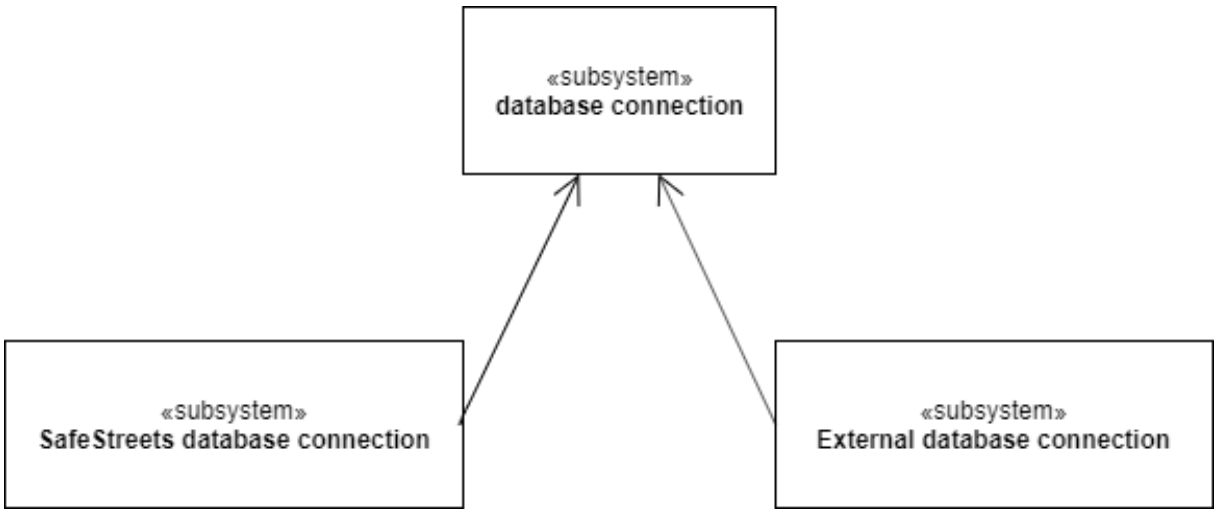

Figure 11: PlaceHolder TODO DB test diagram

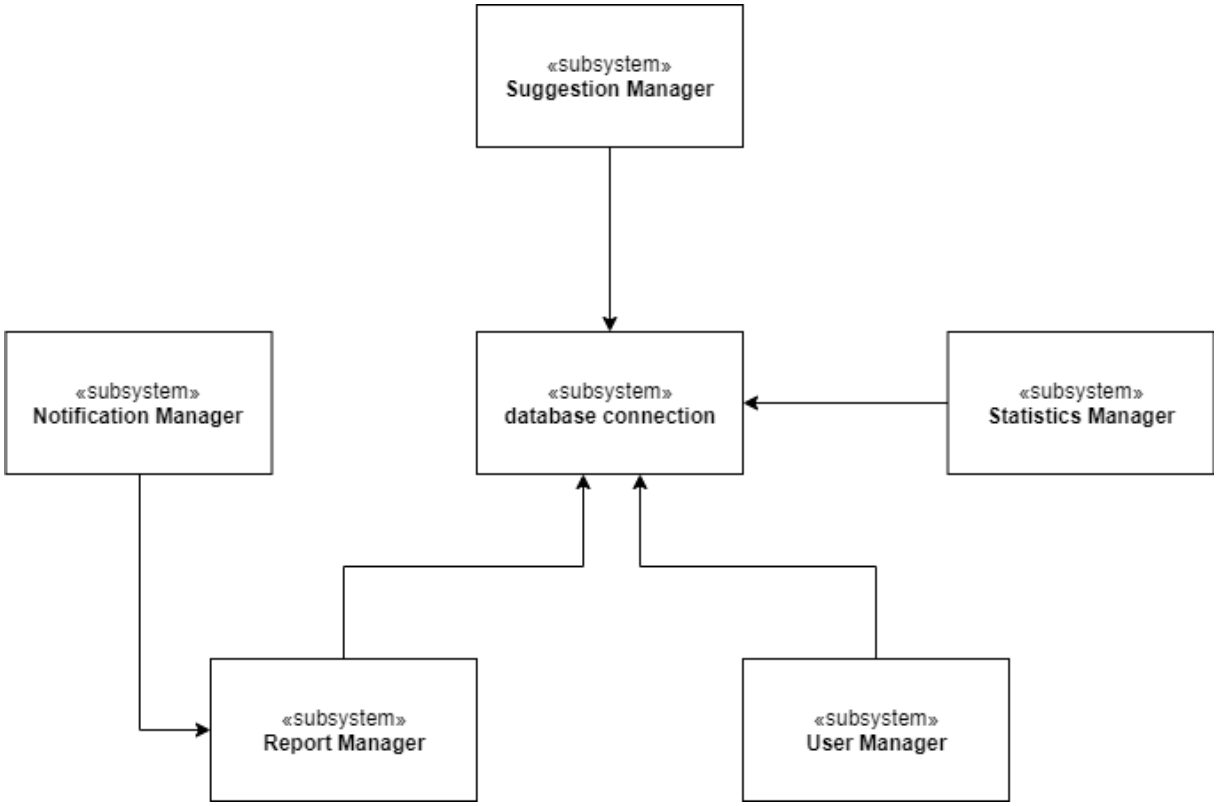

Figure 12: PlaceHolder TODO servlet test

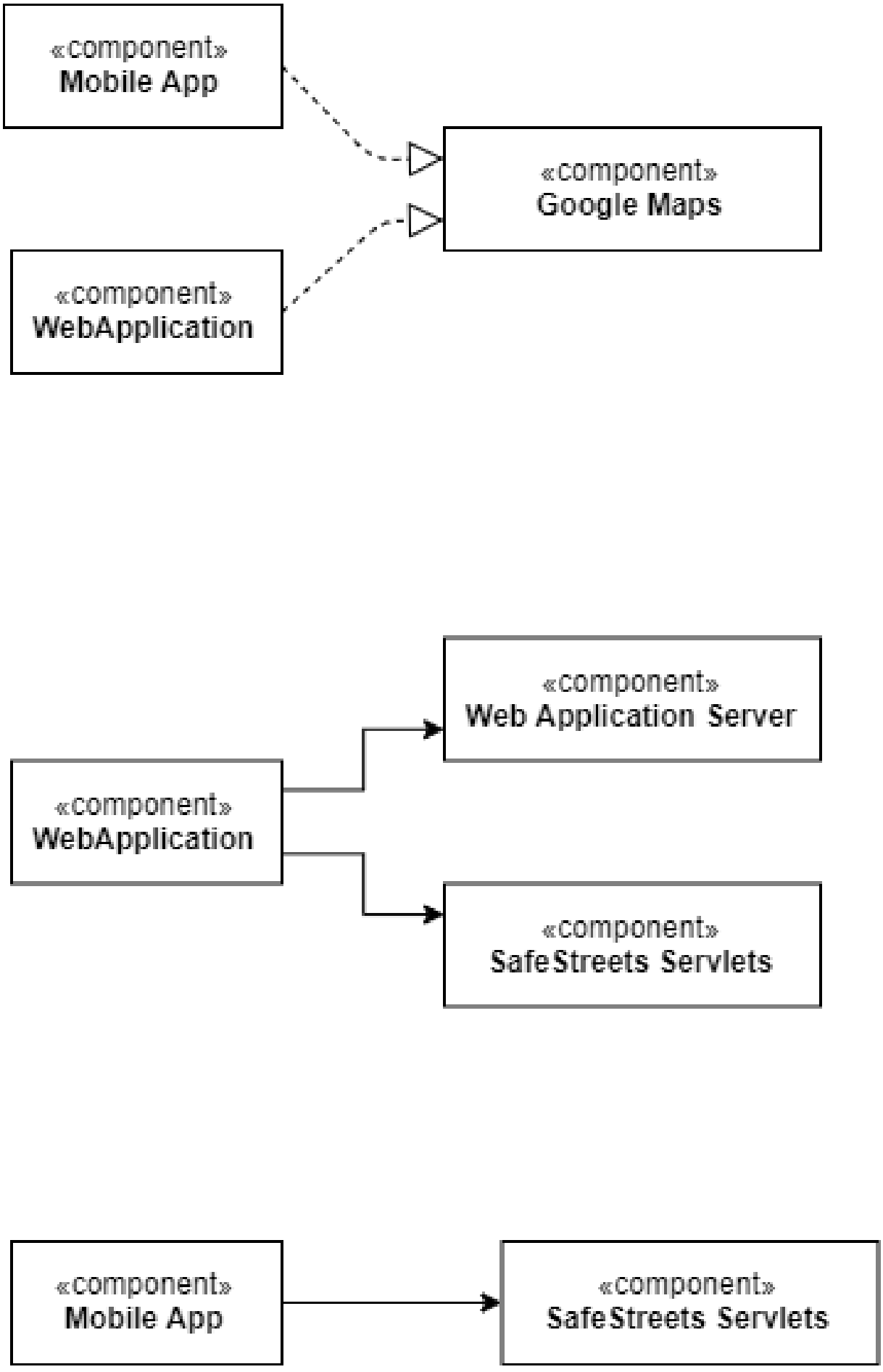

Figure 13: PlaceHolder TODO separate diagrams

6 Effort Spent

Maldini Pietro	
Section	Effort
Discussion about project, organization and project text analysis	5
Introduction	3
Overall Description	7
Specific Requirement's	9
Scenarios	3
UML Modelling	10
Alloy Modelling	9
Total	46 h

Paone Angelo	
Section	Effort
Discussion about project, organization and project text analysis	5
Introduction	2
Overall Description	6
Specific Requirement's	7
Scenarios	5
UML Modelling	14
Alloy Modelling	6
Total	45 h

7 References

7.1 lorem ipsum