POLITECNICO
MILANO 1863

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

SafeStreets

Design Document

Deliverable:	DD
Title:	Implementation and Testing Document
Authors:	Maldini Pietro , Paone Angelo
Version:	1.0
Date:	12-January-2020
Download page:	https://github.com/pm390/MaldiniPaone
Copyright:	Copyright © 2020, Maldini Pietro, Paone Angelo – All rights reserved

Contents

1 Introduction and Scope

1.1 Purpose

The purpose of this document is describing the development of our prototype of SafeStreets, the choices we made in the development , the software we used for developing and for testing and at the end of the document we present how to install and run the developed software.

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, and the system makes statistics and suggestion 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 Document Structure

This chapter debates about contents and structure of Implementation and Testing Documnet, indeed this document is divided in different sections:

1. Introduction: This chapter rovides a general definition of the developed software;
2. Requirements and Functions: This chapter illustrates the requirements which the software meets and the functions developed to satisfy them;
3. Development Choices: This chapter shows the choices made during the developement, the used frameworks, programming languages and softwares are described here.
4. Structure of the source code:This chapter explains how the source codes are organized.
5. Testing: This chapter explains how the testing was carried on during the developement.
6. Installation Instructions : This chapter shows how the software can be installed and run.

2 Requirements and Functions

2.1 Requirements

In this section we present the requirements debated in the RASD and DD and we analize if they are present in the prototype developed.

- R1) Authorities' location must be known by the system when they are in service: //TODO angelo
- R2) When a Citizen makes a report the position is correctly added with the GPS when is available: //TODO angelo
- R3) The right authorities are notified about violations: //TODO angelo
- R4) Authority must be able to provide the system how the assignment finished, resolved and the type of violation, no intervention needed when arrived, false report: //TODO angelo+"DatabaseAccessFacade provides the functionalities to terminate correctly an assignment"
- R5) The system must make Statistics available when asked: This requirement is developed it costitutes an important part for behaviour analysis so it is available in the prototype in order to show how the system could analyse data. The functionality in the developed software must be improved for an actual release of the software. Now the size of the area in which the statistics are computed is fixed. In an actual release it should dinamically change or it could even be choosen by the client. More useful data may be added to statistics consulting data analysts to increase the value of this functionality.
- R6) Statistics are always updated when an event happens: This requirement is enforced in the developed software with the developing of Report manager and the functionality it provides to communicate with the DataAccessFacade to save reports and modify Assignments.
- R7) For registering a Municipality his/her data must be provided to a System manager who will add those data to the service to sign up him/her: the web application allows the Manager to register a Municipality only if he/she fills all the fields in a form he/she gains access when successfully logging in. When data is not correctly filled the web page doesn't allow the submission of the form. Even if the form is submitted disabling javascript from the browser the Server side controls the validity of the data and in case of invalid data it informs the client with an appropriate error message.
- R8) A visitor must be able to begin sign up process in the SafeStreets App filling a form with his data: //TODO angelo
- R9) When the creation of an account is successful the system must notify the Visitor sending an email to the address provided in the sign up process: The creation of an account is handled by the appropriate servlet depending on the creator of the account. After the creation of an account an email is sent to the email address of the user. The MailManager handles this part of the interaction. This function is important since it is necessary for users which must be added by other users like municipalities , authorities and managers whose passwords are randomly generated by the system. For showing an additional idea we added a link to delete the account if the creation wasn't done by the owner of the email. But for the lack of an actual server with an univoke name to which deploy our prototype the link is only symbolic.
- R10) When GPS is not available the user can input the position from a map: //TODO angelo
- R11)Users to use the full service must be able to login providing the right credentials: This functionality is provided by the access control done in all the servlets which does the controls of the

user session. For the functionalities which are only for a given type of user an additional check on the user type in the session is made to avoid illegal access to the application. If a user accesses the application without permission (not logged in or access functionalities he/she can't access) than an error is returned with a message informing that the access to the functionalities is not allowed.

- R12) The camera of the mobile phone must be accessible to take photos of violations: //TODO angelo
- R13) Suggestions must be available when municipalities request them: We included two parts of this functionality in the prototype. A static part which allows Citizens and authority to send suggestions to a municipality and a dynamic part which takes statistics around the municipality to build the suggestions. Comments are placed where the last part which uses data about accidents to build statistics could be placed in the StatisticsBuilder class.
- R14) The User must be able to select the licence plate between the ones in output from the Licence Plate Recognition algorithm://TODO angelo
- R15) Each Username is unique: This requirement is enforced by the structure of the Database used.

3 Development Choices

3.1 Adopted Programming Languages

3.1.1 Back End

For the developing of the back-end of the we used Java. We choose java for it being cross-platform, so the possibility of running the code on different machines. Even though the back-end prototype is developed in a single machine this choice allows with some modification to deploy different components on different machine and use functionality like Java Remote Method Invocation to communicate. The choice is also driven by the variety of functionality JEE has. JEE has components for database access but also for the creation of Http Servers. Another advantage of using Java is being Object Oriented and allowing Polymorphism and Inheritance allowing the user to create a structure for a general Object and expanding it in different forms. This allowed a really fast developing of the Response Objects which the server returns to the client. Other advantages in the development are the usage of annotations to enforce properties to functions (I.e.@Override) and last but not least automatic garbage collection. Garbage collection is an advantage for developing an application since it removes part of the controls the developer must perform it also adds a considerable drawback of slowing down the application if object are created and than discarded continuously. To remove part of this overhead the use of Object Pooling techique for the most resource consuming Objects is applied, in the project we used an Object Pool for Database Connections.

3.1.2 Front End: Web Application

To develop the front end we used javascript with jquery and leaflet. Jquery allows to write more readable code than javascript and simplifies the developement, also caching mechanism of browsers may allow users to load it from memory if it is already cached. A disadvantage is the difficulty to develop with different components and pages and in most cases the code written is only usable in one or few pages. Being the web applicaiton developed rather small it was possible to develop it without using complex frameworks. Leaflet is a lightweight javascript map library. It provides the functionalities required by the software with a quick loading time and an easy inclusion in the application.

3.1.3 Front End: Mobile Application

3.2 Software used

3.2.1 Back End

The Back End developement was carried on using different softwares:

- Eclipse IDE for Java EE Developers : This IDE provides a rich environment to develop applications in Java. Eclipse has components for managing projects and the dependencies of the project . The IDE also supports JUnit Test Suite and allows to do Coverage testing. It also allows to work with git inside it to manage versioning of the project. We used it also for developing of the web App since it allows editing also of javascript ,html and css files.

- MySQL : MySQL is a relational DBMS easy to set-up, to export and import using MySQL workbench GUI. The GUI allows also simple table creation , table value modifications, stored procedure creation and trigger creation It is also easy to connect from the Java Code .

- Apache Tomcat : Tomcat is an open source implementation of the Java Servlet. This software is used to deploy the Back End. Using Eclipse IDE for the developement after downloading Tomcat the setup is really fast and easy.

3.2.2 Front End

//TODO angelo

3.3 Adopeted API

3.3.1 Back End

In the developement of the Back End we used the API provided by Mysql server to connect to the database , the API of tomcat servlets to develop and deploy servlets and the Java Mail Api to send emails to the users.

3.3.2 FrontEnd

In the developement of the Front End we used the API provided by openStreetMaps in the web Apllication to get the map shown with leaflet and the geolocation API provided by the browser to get the user position.//TODO angelo

4 Structure of the source code

Pietro Maldini

Section	Effort
Introduction	4
Architectural Design	30
User Interface Design	2
Requirements Traceability	5
Implementation, Integration and Test Plan	6
Total	47

Angelo Paone

Section	Effort
Introduction	4
Architectural Design	28
User Interface Design	2
Requirements Traceability	4
Implementation, Integration and Test Plan	5
Total	43

5 Testing

The System is divided in various subsystems:

- Mobile App
- Web Application
- Web Server
- Application Server
- DataBase Connection
- DataBase Server
- External Systems:Google Maps,External DataBases

The subsystems not external will be implemented and tested. They will also be integrated together, and with external systems. Since the Web Server provides user only static data (HTML, CSS, JavaScript) and is deployed to a different device than the web Application, this component can be implemented and tested in any moment during the development but the integration must be done when the full application is implemented. Given our design choice of having a multiple servlet server which communicates with DataBase through a facade, our implementation must have as a starting point the implementation of the DataBase Server and the connection with it.

5.1 DataBase Server and Connection

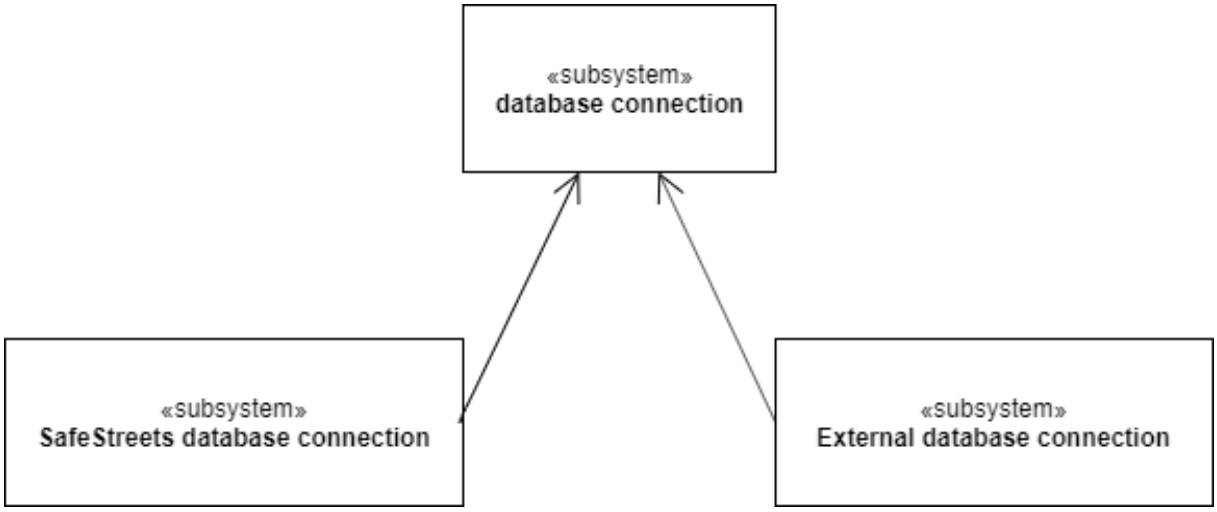

Figure 1: Database Server and Connection Integration

This phase is important so several test will be run to check the correct functioning of the connection and the functions for data access needed by the system. In this phase also the abstract classes for the connection to external DataBases will be implemented and tested with a dummy DataBase to test that the design choice for external DataBase works correctly. Then the DataBase connection is implemented and integrated with the DataBase Server and the External DataBase connection subsystems. The testing phase must cover as much cases as possible and different test approaches must be used to test the functionalities provided by DataBase Connection subsystem. After this initial phase the Application Server must be implemented

5.2 Web Application

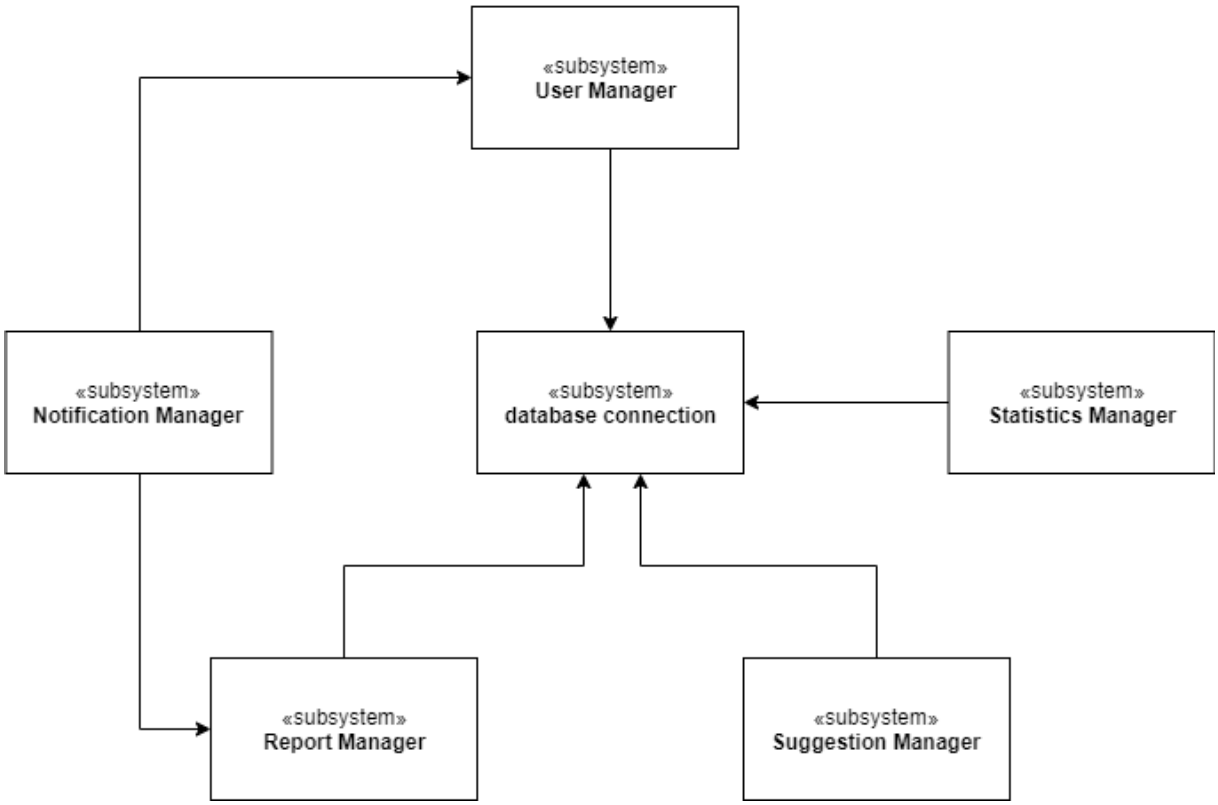

Figure 2: Servlet Integration

The implementation and testing can be performed in parallel for all the managers except for the Notification Manager which requires the implementation of the Report Manager and of the User Manager. All these components must be tested using test stubs to access and test functionalities of the managers, covering all of them and using coverage tests to check if all the checks done on data, and all possible kind of issues (I.e. missing DataBase, null values, exceptions) are handled and there are no controls which aren't reachable for any kind of input. The components are then integrated with the DataBase Connection subsystem

5.3 Mobile App and Web Application

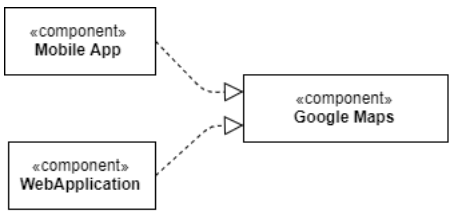

Figure 3: Google Maps Integration

In parallel to the implementation of Application Server the development of the WebApplication and Mobile App can be carried on.

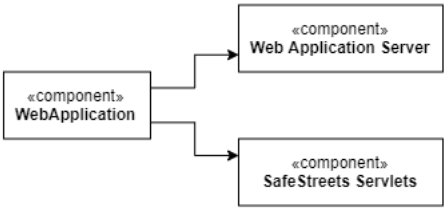

Figure 4: Web Application Integration

After the implementation both needs to be integrated with APIs provided by Google Maps and when Application Server development will be complete they will be integrated and tested together. Web Application also needs to be integrated and tested with Web Server.

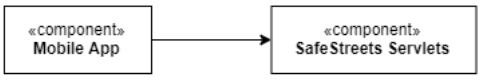

Figure 5: Mobile Applicaiton Integration

5.4 Main feature test plan after all components are integrated

Once all the systems are integrated an additional testing phase will be devoted to check the possible interaction of users with the system. For the main functionalities and interactions in the mobile application testing we will use some flowcharts, and do coverage testing over them and check that no unexpected results occurs with different values when citizens make reports, and when authorities accepts and terminate assignments.

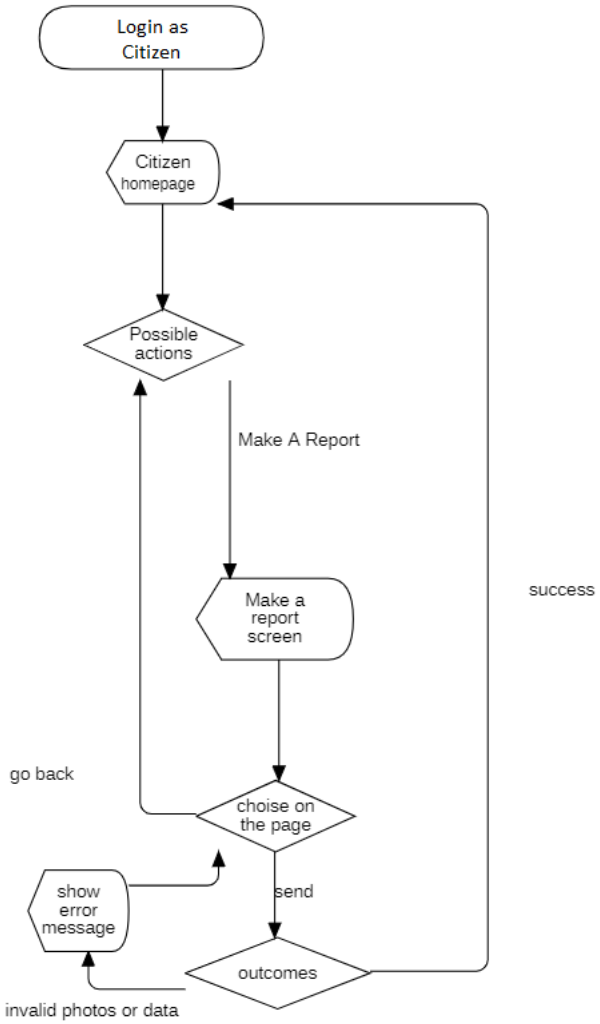

Figure 6: FlowChart: Citizen main features testing

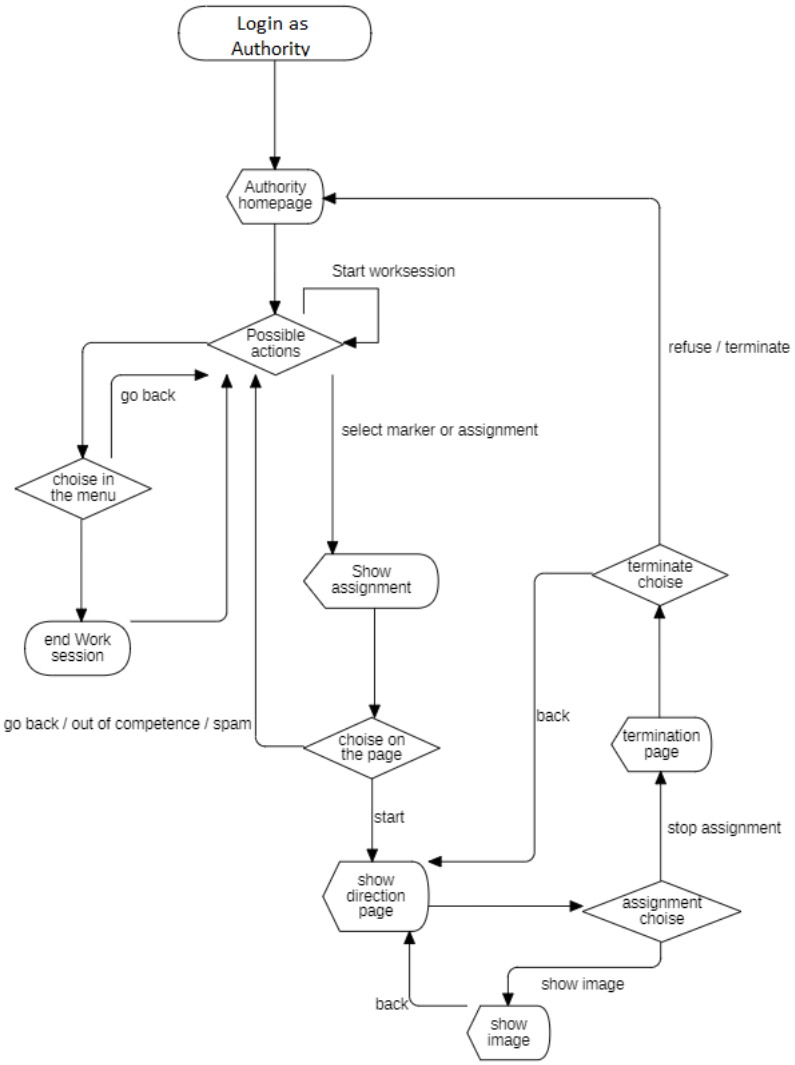

Figure 7: FlowChart: Authority main features testing

6 Installation Instructions

6.1 Used Softwares

- UML Modeling Tools: StarUML, Draw.io
- Document Production: TeXworks
- Collaboration Environment :GitHub.com