Politecnico di Milano

SAFESTREETS - DD

Samuele Meta - Stiven Metaj

Supervisor: Matteo Rossi

Department of Computer Science and Engineering

December 6, 2019

Contents

1 Introduction **3**
1.1 Purpose . . . 3
1.2 Scope . . . 3
1.3 Definitions, Acronyms, Abbreviations . . . 3
1.3.1 Definitions . . . 4
1.3.2 Acronyms . . . 4
1.3.3 Abbreviations . . . 5
1.4 Revision History . . . 5
1.5 Reference Documents . . . 5
1.6 Document Structure . . . 5

2 Architectural Design **6**
2.1 Overview . . . 6
2.2 Component View . . . 6
2.3 Deployment View . . . 8
2.4 Runtime View . . . 8
2.5 Component Interfaces . . . 8
2.5.1 REST API . . . 8
2.6 Selected Architectural Styles and Patterns . . . 8
2.6.1 Multi-tier Architecture . . . 8
2.6.2 Use of RESTful guidelines . . . 8
2.7 Other Design Decisions . . . 8
2.7.1 Thin Client . . . 8
2.7.2 MVC . . . 8
2.7.3 Firewalls . . . 8

3 User Interface Design **9**
3.1 User eXperience Diagrams . . . 9

4 Requirements Traceability **10**

5 Implementation, Integration and Test Plan 11

5.1 Overview 11

5.2 Component Integration 11

5.3 Something on Testing? 11

6 Effort Spent 12

1 Introduction

1.1 Purpose

The following Design Document (DD) is aimed to provide an overview of the *SafeStreets* application, explaining how to satisfy the project requirements declared in the RASD and stating the successive refinements made together with the Stakeholders according to their needs. The document is mainly intended to be used by developers teams as a guidance in the development process, by testing teams to write automated testing and to avoid structural degradation of the system in case of maintenance or future extension. Indeed, its purpose is to provide a functional description of the main architectural components, their interfaces and their interactions, along with the design patterns and algorithms to be implemented.

1.2 Scope

As explained in the RASD document, *SafeStreets* is a crowd-sourced mobile application that allows Citizens to notify Authorities about traffic violations, with particular emphasis on parking contraventions. Citizens are able to manage their reports, visualizing the whole history and modifying or removing them. On the other hand, Authorities can access a complete overview of the incoming reports and choose if to accept or reject them.

1.3 Definitions, Acronyms, Abbreviations

In this section follow definitions, acronyms (including their meaning) and abbreviations used in this document.

1.3.1 Definitions

- *Client*: a desktop computer or workstation that is capable of obtaining information and applications from a Server.
- *Server*: a computer or computer program which manages access to a centralized resource or service in a network.
- *Firewall*: a part of a computer system or network which is designed to block unauthorized access while permitting outward communication.
- *Port*: an endpoint of communication in an operating system.
- *Design Pattern*: reusable software solution to a commonly occurring problem within a given context of software design.

1.3.2 Acronyms

DBMS	DataBase Management System
HTTPS	Hyper Text Transfer Protocol over SSL
API	Application Programming Interface
REST	REpresentational State Transfer
MVC	Model View Controller
OS	Operative System
UI	User Interface
UX	User Experience
URL	Uniform Resource Locator
RASD	Requirements Analysis and Specification Document
SPA	Single Page Application
ERD	Entity-Relationship Diagram
SSL	Secure Sockets Layer
DDoS	Distributed Denial of Service

Table 1.1: *Acronyms*

1.3.3 Abbreviations

- [**R.n**]: n-th Requirement in the RASD document

1.4 Revision History

[TBD]

1.5 Reference Documents

- Specifications document “SafeStreets. Mandatory project assignment”
- SafeStreets RASD Document
- IEEE Standard 1016-2009: IEEE Standard on Software Design Descriptions
- UML documentation: `https://www.uml-diagrams.org`

1.6 Document Structure

The rest of the document is organized as follows:

- **Architectural Design**: details the System’s architecture by defining the main components and the relationships between them as well as specifying the hardware needed for the System deployment. It will also be focused on design choices and architectural styles, patterns and paradigms.
- **User Interface Design**: provides further details on the UI defined in the RASD document through the use of UX modeling.
- **Requirements Traceability**: shows the relations between the requirements from the RASD and the design choices of the DD and how they are satisfied by the latter.
- **Implementation, Integration and Test Plan**: provides a roadmapping of the implementation and integration process of all components and explains how the integration will be tested.
- **Effort Spent**: describes how the work has been split between the members of the team and how long did the DD take to be completed.

2 Architectural Design

2.1 Overview

2.2 Component View

Web Client, Mobile Client

This component represents the Client machines that access to the API of the Business Logic of the System. They do not have any notable functionality to be outlined, due to the fact that they are implemented as thin clients as explained below. The Web Clients, accessing trough a browser, need the Presentation component in order to display the web pages of the application. This layer provides to the it the structure of the User Interface without accessing data and application logic. On the other hand, the Mobile Client embeds the Network Manager, the Presentation Component and the Data Manager.

User Manager

This Manager includes all the operations that affect the user-related data. It exposes methods to change account credentials and preferencies. Furthermore, it manages the data stored in the DB through the interaction with the interface of the Model Interface.

Authentication Manager

This Manager handles all the methods related to the authentication task and access to the System. Specifically, it deals with both the User registration and login, making use of the Model Interface to interact with the DB. It handles credentials verification and constraints, guaranteeing the creation of consistent accounts. Moreover, it also uses the Mail Service Interface to communicate with the Mailing Service and send registration emails.

Report Manager

This Manager allows the Citizen to create reports and send them to the platform. It interacts with the Model Interface in order to store them permanently and to retrieve the list of the previous reports. On the other hand, it allows to the Authorities to have a complete overview of the incoming reports and gives the possibility to accept or refuse them.

Statistics Manager

This Manager handles the requests of Guests, Citizens and Authorities to obtain insights about data through statistics. It is responsible to aggregate the information, eventually interacting with the Municipality Interface, and return it to the User. It also invokes the Authentication Manager to check the role of the User, in order to return only the allowed statistics.

External Services Interfaces

Some of the components in the System are also dedicated to communicating with external services through specific interfaces. These interactions are bilateral and essential to guarantee the application's functionalities. These components are both on the Client side and on the Server side. In particular the interfaces needed by the System are:

- *Mail Service Interface*: is responsible of the interaction with the mail service. It is used to send an email confirmation to the User, at the request of the Authentication Manager, during the registration phase.
- *Map Service Interface*: is responsible of providing, at the request of the Statistics Manager, a visual representation of raw data whenever these have a spatial dimension. It can also be invoked by the Report Manager in the eventuality that the Citizen is not able to remember the name of the street and needs a map as an helper.
- *Push Notification Interface*: interacts with the Push Notification Service and is responsible to notify to the Citizen messages of interest, such as the outcome of the profile verification or the acceptance of a report.

- *Municipality Interface*: is responsible of providing, at the request of the Statistics Manager, additional data in possession of the Municipality, in order to join them with local statistics and provide to the User a complete overview.

Model Interface

The Model Interface includes two sub-components. The Data component provides the set of Classes corresponding to the tables contained in the Database. The Storage Interface provides the methods for querying the Database.

Data Base

This component represents the DBMS, which provides the interfaces to retrieve and store data. In the data base, for each user, credentials and application data are safely and securely stored.

2.3 Deployment View

2.4 Runtime View

2.5 Component Interfaces

2.5.1 REST API

2.6 Selected Architectural Styles and Patterns

2.6.1 Multi-tier Architecture

2.6.2 Use of RESTful guidelines

2.7 Other Design Decisions

2.7.1 Thin Client

2.7.2 MVC

2.7.3 Firewalls

3 User Interface Design

3.1 User eXperience Diagrams

4 Requirements Traceability

5 Implementation, Integration and Test Plan

5.1 Overview

5.2 Component Integration

5.3 Something on Testing?

6 Effort Spent

The effort spent from each member of the team to build the DD can be summarized with the following tables: