Software engineering project

SafeStreets

DD : Design Document.

Authors:
Mattia Micheloni,
Francesco Montanaro,
Amedeo Pachera.

Professor:
M.Rossi.

Version 1.0
08/11/19

Contents

Chapter 1

Introduction

This chapter explains

1.1 Purpose

The purpose of this document is to give a more detailed description of the architectural choises made to perform into the software system all the functionalities described into the RAS document.
Therefore, if the RASD describes all the features that the system must indulge according to the stakeholders needs, the DD describes how it has been thought to convert all the high-level system requirements into the design structure.
It provides guidance and material which is intended to assist technical staff to perform system testing and implementation and, moreover, to facilitate future maintenance and features extension.
List of topics covered by the document:

- High-level architecture.
- Main Components, interfaces and deployment.
- Runtime behavior.
- Design Patterns.
- UI details.
- Mapping between requirements and architecture.
- Implementation, integration and testing plan.

1.2 Scope

Here a short review of SafeStreet scope referred to what just stated in the RASD document.
Safestreet is a software thought to make users able to report traffic violations to authorities. Anyone can report violations simply opening SafeStreet application and taking several pictures of them. For each violation is required at most 6 pictures in which 1 of the involved vehicle license plate.
In order to guarantee to authorities the reliability of data gathered, which means that pictures can not be altered or modified, the latter can be only taken in the application itself.
To make it easier the reporting process all other information are automatically mined by the system, such as time, date, geographical location and license plate; the last one is extracted by the system from the license plate picture.
Finally, user is asked to choose the violation type from a predefined set and confirm the above listed data that will be sent.
If no errors occur, the system stores the violation, otherwise it is refused.
Authorities, in order to access information concerning reports made by users, must log into the SafeStreet website.
They are allowed to discard, modify and validate violations sent by users, in order to take into account only pertinent violations.
Thanks to the data granted by the SafeStreet initiative and the validation performed by authorities, they can even decide to generate traffic tickets from validated violations.
Authorities can also upload information about accidents that occur on the municipality.
The software provides also a service to offer both users and authorities statistics computed on the available data. Obviously, in order to make statistics consistent and reliable, the latter are generated only by information validated by authorities.
Statistics highlight streets or areas with the highest frequency of violations, the vehicles that commit the most ones or the infractions trend that occurs over time.
Crossing data gathered by users and authorities, the system can identify potentially unsafe areas and suggest interventions in order to improve streets safety.

1.3 Definitions, acronyms, abbreviations

This section gives some definitions in 1.3.1, acronyms in 1.3.2 and abbreviation in 1.3.3 which will be use in the document, in order to explain some concept and help the general understanding.

1.3.1 Definitions

1.3.2 Acronyms

- API: Application Programming Interface
- GPS: Global Positioning System
- UI: User interface
- AWS: Amazon Web Server
- OCR: Optical Character Recognition

1.3.3 Abbreviations

- Gn: nth Goal.
- R_n : nth Requirement.

1.4 Revision history

- Version 1.0 : 10/12/2019

1.5 Reference documents

- Design document: “Mandatory Project Assignment AY 2018/2019”.
- Lecture slides of professor M.Rossi and E.di Nitto of Politecnico di Milano.

1.6 Document's overview

The DD document is structured by seven chapters as describes below:

- Chapter 1:
- Chapter 2:
- Chapter 3:
- Chapter 4:
- Chapter 5:

Chapter 2

Architectural design

2.1 Overview: High-level components and their interaction

[Overview]

2.2 Component view

2.3 Deployment view

2.4 Runtime view

2.5 Component interfaces

2.6 Selected architectural style and patterns

2.7 Other design decision

2.7.1 License plate recognition algorithm

Tesseract is a free OCR engine, it can recognize words of approximately every language. It offers a huge variety of API for any type of programming languages and also a C/C++ library to developers who want to build their own application on it. Tesseract is very powerful in recognizing black on white text so it's perfect for license plate recognition, and it has also an API for Android and iOS developers so it's optimized to be run on mobile devices, which make easy to SafeStreets to use it directly on devices instead of running it in the server

Chapter 3

User interface design

Chapter 4

Requirements traceability

Chapter 5

Implementation, integration and test planning

Chapter 6

Effort spent

Mattia Micheloni

TASK	HOURS
Ch.1	1
Ch.2	4
Ch.3	18
Ch.4	12

Francesco Montanaro

TASK	HOURS
Ch.1	4.5
Ch.2	4
Ch.3	13
Ch.4	13

Amedeo Pachera

TASK	HOURS
Ch.1	2
Ch.2	3
Ch.3	13
Ch.4	14

Chapter 7

References