POLITECNICO
MILANO 1863

SafeStreets project

Manuel Pedrozo, Tomás Perez Molina

Implementation and Testing Deliverable

Deliverable:	ITD
Title:	Implementation and Testing Deliverable
Authors:	Manuel Pedrozo, Tomás Perez Molina
Version:	1.0
Date:	January 9, 2020
Download page:	https://github.com/lethanity/PedrozoPerez
Copyright:	Copyright © 2019, Manuel Pedrozo, Tomás Perez Molina – All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose and scope

This is the Implementation and Test Deliverable document, its purpose is to explain the implementation and testing process for the SafeStreets system based on the RASD and DD documents. In the following section the topics to be covered are:

- An overview of the proposed architecture
- Implemented requirements and their motivation
- Adopted development frameworks, along with their advantages and disadvantages.
- The structure of the source code
- Testing
- Installation instructions

2 Implemented requirements

In this section, the requirements specified in chapter 3 of the RASD document are categorized according to their implementation state.

2.1 Included requirements

The following requirements were fully implemented:

[FR1] - The user is able to take pictures from the mobile application to add to a report.

[FR2] - The user is able to fill out a form providing information about a traffic violation, consisting of:

- Type of violation
- License plate
- At least one photo of the scene

[FR3] - The application can determine date, time and location based on information provided by the device when the pictures are taken and adds this metadata to the report.

[FR4] - The user can submit a full report to the system

[FR5] - The user is provided a map where the location of reports are indicated by markers.

[FR6] - The user can filter the reports shown on the map by date, time and type of violation.

[FR7] - The user can search for a specific location on the map by inputting coordinates or an address.

[FR8] - The user can select a report on the map and obtain information about it. This information includes:

- Report ID: unique reference to each report
- Type of violation
- Date and time
- Location: gps coordinates, approximate street name and number

[FR9] - A secured API is provided to obtain and filter report information. The following filters are possible:

- Date and time
- Location
- Type of violation

[SR1] - Only photos taken through the mobile application are submitted, the user cannot upload a picture previously saved on their device.

[SR2] - Special API keys are generated and provided to authorities

[FR10] - A secured API accessible only to authorities, provides the following information about reports:

- Photos

- License plate

[FR13] - The API includes the system's degree of confidence in the report to the provided report information after a query.

[FR14] - Report information provided by the API can be filtered by their degree of confidence.

[FR15] - Users can review photos with a license plate recognition lower than 80% to adjust the confidence in the recognition.

[FR18] - The user can register by inputting his full name and email, and choosing a username and password.

[FR19] - The user has access to view his full account information.

[FR20] - The user can edit his email and full name.

[FR21] - An API key is provided to the user via their profile screen in the mobile application

[FR22] - The user can login to the mobile application by providing their username and password.

2.2 Partially included requirements

Since there was no integration with an external license plate registry, the degree of confidence is based on the license plate recognition only.

[FR12] - A degree of confidence is assigned to each report, where a high confidence report will have:

- License plate recognition of 80%+ confidence.
- Car recognition of 80%+ confidence.
- The license plate and car pair are found in the license plate registry.

If any of these criteria are not met the report is considered low confidence.

2.3 Excluded requirements

As previously mentioned, currently there is no integration with an external license plate registry.

[FR11] - Detected cars and license plates are cross referenced with the license plate registry to ensure the license plate belongs to the car.

The following requirements belong to the advanced functionality, which was not required for the implementation:

[SR3] - End-to-end encryption is provided for the submission of reports.

[FR16] - Information obtained from the ticketing system is used to determine if a report contributed to the issuing of a traffic ticket.

[FR17] - Whether the report contributed to a traffic ticket is included in the information provided by the mobile application and the API.

3 Adopted development frameworks

Since the development time for the application is short, for the most part the team opted for known technologies as to reduce the risk of missing the timeline by encountering unexpected obstacles. This also allowed the team to focus on unknown areas, like the integration with external APIs.

3.1 Back-end server

As stated in section 5.1.2 of the Design Document, the backend is implemented in Kotlin utilizing the Spring boot framework. Kotlin was chosen over Java because its more concise, safer and allows for faster development for those familiar with it. Spring is one of the most popular frameworks for the development of web applications, based on the JVM. It allows for fast development and easy integration with documentation tools that help speed up the work. Because of its maturity and popularity, there is a great amount of documentation on the framework itself. There are a few alternatives, like Play and Grails, which offer similar functionalities, but Spring has been proven to be the most performant of the three.

4 Structure of the source code

4.1 Back-end server

The source code is organized following the Gradle standard.

The application resides in the /back directory, where Gradle configuration files can be found.

- /src
  - ◇ /main contains the source code files.
  - ◇ /java/com/openalpr/jni contains the OpenALPR java files.
  - ◇ /kotlin/se2/SafeStreets/back contains the kotlin files, organized according to Spring boot framework.
    - /controller contains Spring controller classes.
    - /model contains application models and DTOs.
    - /repository contains Spring repository classes.
    - /security contains Spring security configuration and filters.
    - /service contains Spring service classes.
  - ◇ /resources contains resources used by the application, like Spring properties files.
- /test contains the source code files for testing.
  - ◇ /kotlin/se2/SafeStreets/back contains unit test classes.
  - ◇ /resources contains files used for testing, like images.

5 Testing

6 Installation

6.1 Back-end server and database

The recommended way of installing the application is through Docker and the provided docker-compose file. This will set up both the backend server and the mongodb database.

Pre-requisites:

- Have Docker installed in your computer, can be downloaded from: https://www.docker.com/
- Have an internet connection for installation.

Installation steps:

- Make sure Docker is running.
- Open the terminal/command line at the directory containing the provided docker-compose.yml file.
- Type and run the following command: docker-compose up.
- Wait for Docker to finish the setup.
- To check if everything is ok, open a browser and navigate to http://localhost:8080/auth/ping. The text “pong” should be displayed on your screen.

7 Effort Spent

7.1 Manuel Pedrozo

Task	Hours

Table 1: Effort spent by Manuel Pedrozo

7.2 Tomás Perez Molina

Task	Hours

Table 2: Effort spent by Tomás Perez Molina

8 References

- “SafeStreets Mandatory Project Assignment”
- “Implementation Assignment”

