POLITECNICO DI MILANO 1863

SOFTWARE ENGINEERING 2 PROJECT

Design Document (DD)

SafeStreets

Version 1.0

Authors
Tiberio Galbiati
Saeid Rezaei

Supervisor
Dr. Matteo Rossi

Download page: https://github.com/TiberioG/GalbiatiRezaei.git

November 19, 2019

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

This is the Requirement Analysis and Specification Document (RASD) of SafeStreet application. Goals of this document are to completely describe the system in terms of functional and non-functional requirements, analyze the real needs of the customer in order to model the system, show the constraints and the limit of the software and indicate the typical use cases that will occur after the release. This document is addressed to the developers who have to implement the requirements and could be used as a contractual basis.

1.1.1 Description of the given problem

SafeStreets is a crowd-sourced application that intends to provide users with the possibility to notify authorities when traffic violations occur, and in particular parking violations. The application allows users to send to authorities pictures of violations, including their date, time and position. Examples of violations are: vehicles parked in the middle of bike lanes, in places reserved for people with disabilities, on foothpats, double parking etc.

SafeStreets stores the information provided by users, completing it with suitable meta-data every time it recieves a picture. In paricular it is able to read automatically the license plate of a vehicle and store it without asking the user to type it. Also it stores the type of the violation which is input by the user from a provided list. Lastly it stores the name of the street where the violation occurred, rereiving it automatically from the geographical position where the user took the picture. Then the application allows both end users and authorities to mine the information crowd-souced. Two visualizations are offfered: the first is an interactive map where are high-lighted with a gradient color the streets with the highest frequency of violations. The second is a list of the vehicles that committed the most violations (available only to authority users).

In addition the app offers a service that creates automatically traffic tickets which can be approved and sent to citizens by the local police. This is done using the data crowd-soucred by the users. The application guarantees that every picture used to generate a ticket has't been altered. In addition, the information about issued tickets is used to build statistics. Two kind of statistics are offered: a list of people who received the highest number of tickets and some trends of the issued tickets over time and the ratio of approved tickets over the violations reported.

1.2 Scope

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

- Heatmap : A heatmap is a graphical representation of data that uses a system of color-coding to represent different values
- Enduser : a regular citizen which will use the app
- Authority user : someone who's working for an authority like police, municipality etc.
- Geocoding : the process of converting addresses (like a street address) into geographic coordinates (latitude and longitude)
- Reverse geocoding: the process of converting geographic coordinates into a human-readable address

1.3.2 Acronyms

- ALPR : Automated Licence Plate Recognition
- GUI : Graphical User Interface
- GDPR : EU General Data Protection Regulation
- API : Application Programming Interface

1.3.3 Abbreviations

1.4 Revision history

This is the first released version 10/11/2019.

1.5 Reference Documents

References

[1] OpenALPR Technology Inc. , OpenALPR documentation http://doc.openalpr.com

[2] MongoDB Inc, The MongoDB 4.2 Manual https://docs.mongodb.com/manual/

[3] Node.js Foundation, Node.js v13.1.0 Documentation https://nodejs.org/api/

[4] StrongLoop, IBM, and other expressjs.com contributors, Express.js website http://expressjs.com

[5] GOOGLE inc, Google Maps Platform Documentation | Geocoding https://developers.google.com/maps/documentation/geocoding/start

[6] GOOGLE inc, Google Maps Platform Documentation | Heatmap https://developers.google.com/maps/documentation/javascript/heatmaplayer

1.6 Document Structure

This document is divided in five parts.

1. **Intrduction**
2. **Architectural Deisgn**
3. **User Interface Design**
4. **Requirements Traceability**
5. **Implementation, Integration and Test Plan**
6. **Effort spent** contains the tables where we reported for each group member the hour spent working on the project

2 Architectural Design

2.1 Overview

The general architecture of our system has three tiers. We have a mobile app running on mobile devices, smartphones or tablets with ios or Android. then we have a server

the kind of architecture is distributed logic as explainde in the slides.

2.2 Component view

2.2.1 Mobile App

The mobile application is the c

2.3 Report Violation

This component is responsible of showing the widget to take the camera.

2.4 Form screen

This component is needed after the picture has been taken to show the form where the user will select the kind of violation.

2.5 Deployment view

In Figure **??** is shown the Deployment diagram.

The deployment consist of three tiers. The first tier consist is **Mobile device** the user will use, which can be a smartphone or a tablet using as operating system either iOS or Android. The exection environment is the built Flutter app.

The second tier is the **Application Server**. It is supposed to be a dedicated server running a linux distribution specific for server use. As an example of OS we choose Centos 7. Other distros can be used like Red Hat Enterprise Linux, Debian, OpenSUSE. As execution enviorment we install Node.js which is an open-source JavaScript runtime environment that executes JavaScript code outside of a browser. Inside Node.js we use the web application framework Express.js which is designed for building web applications and APIs.

The third tier is the **DB Server**. It consists in another server where we run the DB system MongoDB. We choose to run the database in a separate server and not in the same as the ApplicationServer in order to increase scalability. MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schema.

2.6 Runtime view

2.7 Selected Architectural styles and patterns

2.8 Other design decisions

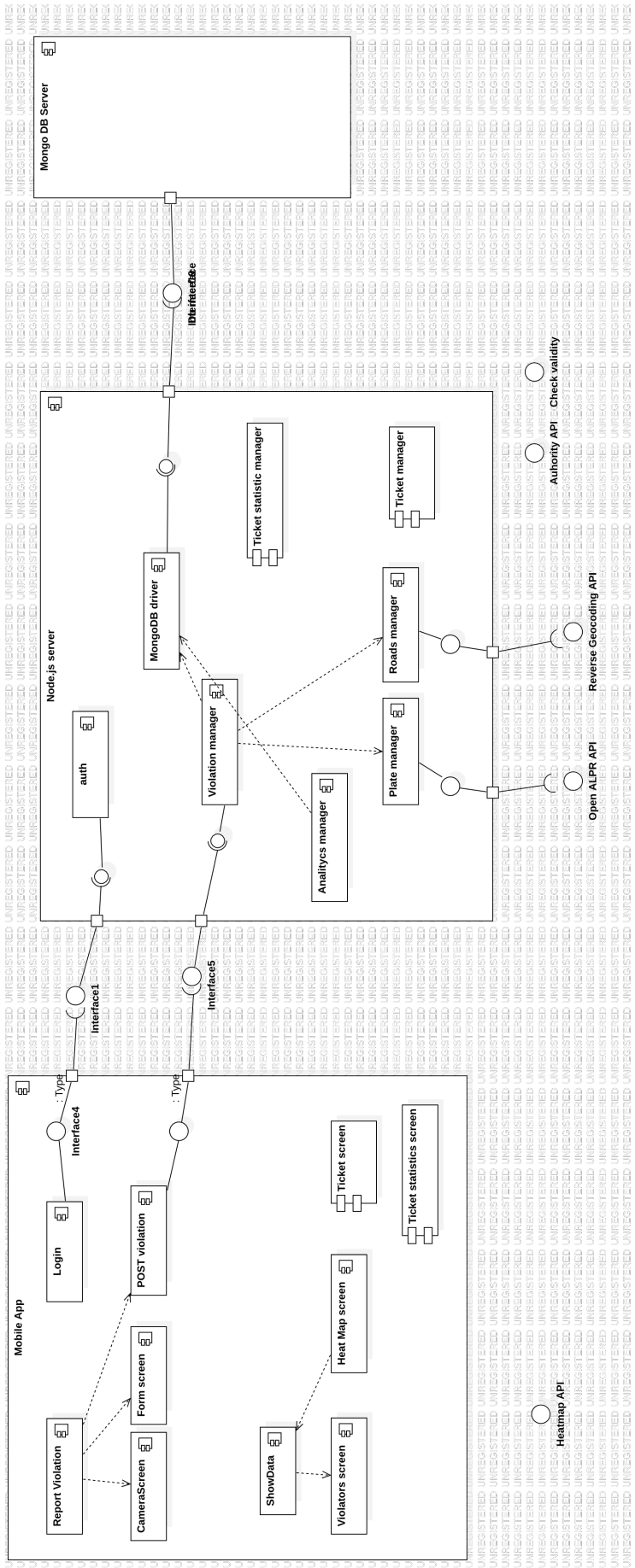

Figure 1: Component diagram

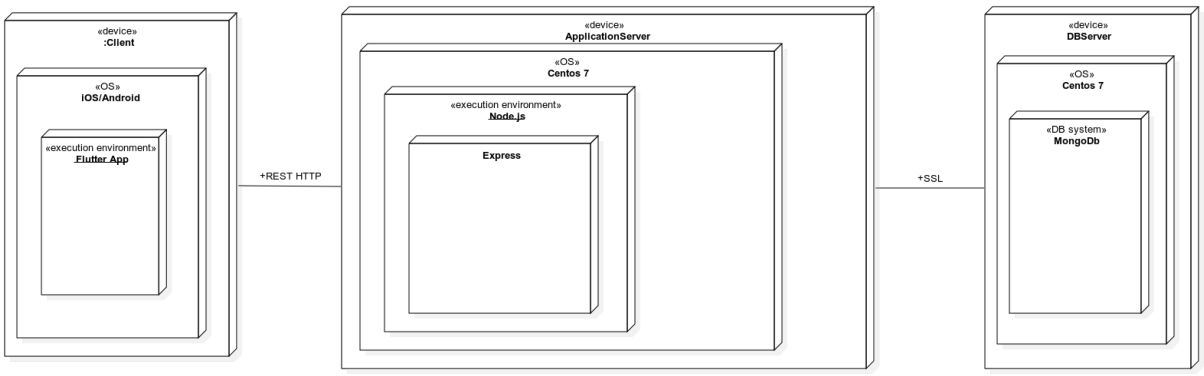

Figure 2: Deplo diagram

3 User Interface Design

uiiiiii

4 Requirements Traceability

5 Implementation, Integration and Test Plan

iiiii

6 Effort Spent

Tiberio	
Task	**Time**
Structure of document	1h
Component diagrams and study of REST	2h
Component diag	1h 30 min
Meeting design	1h 30 min
Total	**39 h**

Saeid	
Task	**Time**
Meeting design	1h 30 min
Total	**36 h 30 min**