POLITECNICO
MILANO 1863

Software engineering project

SafeStreets

ATD: Acceptance Testing Document.

Authors:
Mattia Micheloni,
Francesco Montanaro,
Amedeo Pachera.

Professor:
M.Rossi.

Version 1.0
19/1/20

Contents

Chapter 1

Introduction

1.1 Scope

The scope of this document is to present the acceptance testing of the Software produced by the team MohamedDlaminiQuran and described in section 1.2.
The document is structured in the following way

- $Chapter_1$: Introduction of the document and the analyzed software
- $Chapter_2$: Description of installation setup and issues.
- $Chapter_3$: Description of the Acceptance testing
- $Chapter_4$: Additional notes on the code

1.2 Project

The project to discuss is SafeStreets from MohamedDlaminiQuran team, link to Github repository:
https://github.com/AbdullahQuran/MohamedDlaminiQuran.
In the team's folder there are all the previous documents (RASD, DD and ITD), so first the RASD was analyzed to understand the project with an high view, then the ITD to proceed with the installation and finally the DD was needed to execute better the testing on the software.

1.3 Definitions, acronyms, abbreviations

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

1.3.1 Acronyms

- UI: User interface
- RASD: Requirement Analysis and Specification Document.
- DD: Design Document.
- UT: Unit Testing.
- ITD: Implementation and testing document.

1.3.2 Definitions

- User: The customer of the application who can upload reports concerning traffic violations and have access to the system's statistics services.
- Municipality: The costumer of the application who can access the information on reports made by the users in order to validate or modify them, and eventually generate tickets. They can also upload on the system information about accidents that occur on the territory and have access to the system's statistics services.
- Violations: The set of all the infractions that occur on the territory which are recognized by the system.
- Metadata: The information extracted by the system from a report such as license plate, geographical position, data, time and traffic violation type.

1.3.3 Abbreviations

- R_n : nth Requirement.

1.4 Revision history

- Version 1.0 : 18/1/2020

1.5 Reference documents

- RASD-SafeStreets: team MohamedDlaminiQuran.
- DD-SafeStreets: team MohamedDlaminiQuran.
- ITD-SafeStreets: team MohamedDlaminiQuran.
- https://github.com/AbdullahQuran/MohamedDlaminiQuran.

Chapter 2

Installation setup

The main issue on the installation phase was the different file system of MacOs with respect to the Windows one, so XAMPP works differently on Macbook, but the problem was solved with MAMP which is a Software with the same functionality.
Another problem was the connectivity with the database due to some preference settings of the server , which was solved talking to the other team.
The last issue was the registration of municipality on the website, which was not clear in the ITD but was solved analyzing better the RASD and talking with the team.
To sum up, the installation setup consisted in XAMPP and MAMP (respectively for windows and macOs operative system) and AndroidStudio to test the application.
A final note: the code is not commented so the understanding of the classes and scripts was a bit tricky, specially for the php files.

Chapter 3

Acceptance testing

This chapter presents the acceptance testing on the software, in section 3.1 there is the list of tested requirements implemented by the other team and in section 3.2 there are some test cases and features extracted by the RASP and the user experience.

3.1 Developed requirements checking

3.1.1 website

- R_{23}: The system must allow active municipality to access its account.
  Test case:log in with email of a verified municipality: municipality.milano@gmail.com. password received via email.
  Result:log in successful.
  Test case:log in with email of a verified municipality: municipality.milano@gmail.com. log in with a wrong password.
  Result:log in refused.
  Test case:log in with a wrong email. log in with a right password.
  Result:log in refused.
  Test case:log in with a wrong email. log in with a wrong email password.
  Result:log in refused.

- R_{22} The system must allow municipality to activate its account:
  Test case:Log in with email of a verified municipality: municipality.milano@gmail.com.
  Result:registration email received.
  Test case:Log in with email of a non-verified municipality: randomemail8217@gmail.com.

Result:registration email not received.

- R_{13} The system must allow active municipalities to filter data by type, time, and location of violation.
- R_{14} The system must allow municipalities to access the data of violation.
- R_{24} The system must allow active municipalities to download generated file of violations. The previous three requirements were strictly correlated, therefore they are tested together.
  Test case:try to filter data by time, date and location. try to get raw data via email.
  Result: succesfull.
  Test case:try to filter data by time, date and a location without violations. try to get raw data via email.
  Result: succesfull.
  Test case:try to filter data by time, location and unfeasible date (date in range [start, end] with start $>$ end).try to get raw data via email.
  Result:Not feasible.

3.1.2 Android application

- R_1: The system must allow a user to register a new account
  Test case: Not complete every field of the form.
  Result: The registration isn't allowed.
  Test case: Mismatch the password confirmation.
  Result: The registration isn't allowed.
  Test case: Complete every field of the form.
  Result: Registration completed and email received.

- R_2: The system must allow a user to login to his/her account.
  Test case: Wrong user and password.
  Result: not log in.
  Test case:Right user but wrong password.
  Result:not log in.
  Test case:Wrong user but wrong password.
  Result:not log in.
  Test case:no user / no password / no user and no password.
  Result:not log in.

Test case:Right user and right password.
Result:Log in executed.

- R_3: The system must allow users to take images and to input details of violation.
  Test case: click the icon to take an image, input the address.
  Result: Ask for camera permission, take the picture, the address is autocompleted and the date-time is taken from the device.
  Test case: click the icon to take an image and don't allow camera permission.
  Result: taking photo isn't allowed.

- R_4: The system must allow users to upload images and data of violation to its database.
  Test case: click upload button after taking picture and inputing data.
  Result:The violation is correctly upload.
  Test case: click upload button without picture or metadata.
  Result:Is not possible to upload the violation.

- R_5: The system must be able to store data of violation on the user side.
  This requirement is not visible from the user, is tested with Junit.
  Test case: Report a violation.
  Result: log message from Junit, a file is created on the device storage to control the reliability of the data.

- R_6: The system must be able to communicate with its database.
  Test case: Register and login user, upload and ask violations, use Junit to log a message when a connection is established.
  Result: In the PhpMyAdmin panel all the uploaded data are present in the database, every response is received and the log message is shown every time.
  Test case: Try the application switching off the internet connection of the device
  Result: Log message received : "Connection with the database failed".

- R_8: The system must be able to store data of violation in its database.
  This requirement is not visible from the user, is tested with Junit.
  Test case: Report a violation.

Result: Log message from Junit, the picture taken is saved by the app on the internal storage.

- R_{10}: The system must allow users to send a request for data of violation
  Test case:Click button "My report".
  Result: The app shows a page with the chronology of the reports.

- R_{11}: The system must be able to anonymize data accessed by users.

- R_{12}: The system must be able to send the data of violation to the email the user had registered with.
  The previous two requirements are tested together.
  Test case: Click Request data.
  Result: Check the email with the received excel file and see that there aren't the license plates on the list.
  Test case: Click Request data (with an invalid email of registration).
  Result:A message is shown : "error on sending file".

3.2 Additional features testing

- Screen rotation on Android app:
  Test cases: rotate screen.
  Result: app doesn't crash or change behavior.

- Filtering statistics on App and Website
  Test cases: try different filter on statistics
  Result: the right statistics are shown.
  Test cases: try with impossible filter (wrong interval date).
  Result: an error message is shown.

- Reliability of data on the application.
  Test case:Change the date-time of the device.
  Result:The app take the wrong date and time in the violation.

- The app layout is not relative and scrollable.
  Test case: Open the app on a small devise (Nexus 5X).

Result: the page "report a violation" is not scrollable so it is impossible to upload a violation.

Chapter 4

Additional notes

The testing process required to inspect part of the software system.
In first hand was very difficult to understand the project structure, due to the high number of classes not well organized.
Moreover there were not comments that could clarify the meaning of the code neither the JavaDoc to explain the methods'behavior or class introductions.