



# **POLITECNICO**

## **MILANO 1863**

**Department of Computer Science and Engineering**

---

# **Implementation and Testing**

## **Deliverable Document (ITD)**

---

**SafeStreets**

**- v1.1 -**

*Authors:*

<b>Dlamini, Taras</b>	<b>10451348</b>
<b>Quran, Abdullah</b>	<b>10657807</b>
<b>Abdelaziz, Hesham</b>	<b>10654249</b>

**January 9th, 2020**

<b>1. Introduction</b>	<b>3</b>
1.1 Purpose	3
1.2 Scope	3
1.3 Definitions, acronyms, abbreviations	4
1.4 Revision History	4
1.5 Referenced Document	4
1.6 Document Structure	5
<b>2. Implemented requirements</b>	<b>6</b>
2.1 Mobile application	7
2.1.1 Implemented Requirements	7
2.2 Web application	9
2.2.1 Implemented requirements	9
<b>3. Adopted development frameworks</b>	<b>10</b>
3.1 Adopted programming languages	10
3.1.1 JAVA	10
3.1.2 PHP	12
3.1.3 HTML5	13
3.2 Adopted middle-wares	16
3.3 Additional APIs	17
<b>4. Structure of the source code</b>	<b>17</b>
4.1 Mobile application front-end project structure	17
4.2 Mobile application back-end project structure	18
4.2.1 Java classes	18
4.2.1 PHP Scripts	20
4.3 Web-application front-end project structure	20
4.4 Web-application back-end project structure	23
<b>5. Test Plan</b>	<b>24</b>
5.1 Test scenarios	24
5.1.1 Mobile application test scenarios	24
5.1.2 Web application test scenarios	25
5.2 Test cases	25
5.2.1 Mobile application test cases	25
5.2.2 Web application test cases	32
5.3 Integration strategy	36
<b>6. Installation Instructions</b>	<b>37</b>
6.1 Installing the system	37
6.1.1 Mobile and Web application installation	37
6.2 Troubleshooting	38

6.2 Working on the system	38
6.2.1 Mobile application installation	38
6.2.2 Web application installation	38
<b>7. Effort Spent</b>	<b>39</b>

---

# 1. Introduction

---

## 1.1 Purpose

This Document represents the implementation and testing of SafeStreets as specified in the respective RASD and DD documents. The purpose of this document is to provide a general overview of the implementation and testing activities of the development of a running prototype of SafeStreets .

## 1.2 Scope

SafeStreets is an application trying to provide users with a user friendly and easy way to report traffic violations, such as double parking and car parked in a bike lane. Users would use the application to send authorities images, dates, times, location of violations via Safestreets. SafeStreets must have measures in place to make sure that the images and informations is tamper proof and valid.

In addition, users can see statistics about violations as well as safety of particular areas. Safestreets, would crosscheck data with Municipalities to make give the users Safe notifications if they desire. Furthermore, SafeStreets can give suggestions to Municipalities on how to improve safety in certain unsafe areas.

Finally, Municipalities should be able to use the data they acquire from SafeStreets to issue tickets for particular violations. In this case it is particularly important that the information and images are tamper proof and that the chain of custody of information coming from the user is never broken.

## 1.3 Definitions, acronyms, abbreviations

## 1.4 Revision History

- 9/1/2019 - Initial release

## 1.5 Referenced Document

- Specification document: “SafeStreets Implementation Assignment AY 2019-2020”
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
- Requirements Analysis and Specifications Document: SafeStreets -V1.2-
- Design Document: SafeStreets -V1.2-

## 1.6 Document Structure

Chapter **one** is an introduction of the Implementation document. It describes the document's purpose and scope. It also includes definitions and abbreviations used in this document.

Chapter **two** lists the requirements that were actually implemented in the prototype. Those requirements were explored in detail in the RASD document and their design approach were addressed in the DD document. It also includes a list of the requirements that were not implemented and the justification behind that.

Chapter **three** includes an overview of the adopted development framework, intuition of choosing them, and their pros and cons. In addition, the used middle-wares and the API not mentioned in the DD are discussed.

Chapter **four** presents how the code of the prototype was structured; the front-ends and the backends of both the mobile application and the web application.

Chapter **five** includes the testing scenarios applied on the mobile and the web application and the outputs of these test cases. It also includes the integration strategy

Chapter **six** shows how to install the application and start working on it.

Chapter **seven** shows the effort each member of the group spent to generate this document.

---

## 2. Implemented requirements

---

This chapter presents the functional requirements that were implemented and the ones which were not implemented. Since the implementation of the mobile and the web application is independent, their functional requirements were splitted and discussed separately. As discussed earlier, the functional and nonfunctional requirements of SafeStreets system were explored in depth in the RASD document of this system.

### 2.1 Mobile application

#### 2.1.1 Implemented Requirements

Table 2.1 indicates all the functional requirements related to the mobile application, then it includes the design component responsible for that requirement, and finally, it indicates whether a given requirement was implemented or not.

Table 2.1: Implemented requirements related to the mobile application

Requirement	status of Implementation	Design components
<b>R10:</b> The system must allow users to send a request for data of violation	Yes	<b>Data Manager</b>
<b>R11:</b> The system must be able to anonymize data accessed by users		
<b>R12:</b> The system must be able to send the data of violation to the email the user had registered with.		

<b>R5:</b> The system must be able to store data of violation on the user side	Yes	<b>Database Manager</b>
<b>R6:</b> The system must be able to communicate with its database		
<b>R8:</b> The system must be able to store data of violation in its database		
<b>R1:</b> The system must allow a user to register a new account	Yes	<b>User Authentication Manager</b>
<b>R2:</b> The system must allow a user to login to his/her account		
<b>R3:</b> The system must allow users to take images and to input details of violation	Yes	<b>Violation Manager</b>
<b>R4:</b> The system must allow users to upload images and data of violation to its database		
<b>R9:</b> The system must allow users to view list of violations	No	<b>Statistics Manager</b>
<b>R16:</b> The system must allow users to see a list unsafe areas		
<b>R17:</b> The system must allow users to filter the list of unsafe areas by location, type, time, and frequency		
<b>R18:</b> The system must allow users to send a request to subscribe to notification service	No	<b>Notification Manager</b>
<b>R19:</b> The system must allow users to get safety notifications about the area he is currently in		
<b>R21:</b> The system must be able to notify the user if the safety status of an area has changed		



<b>R7:</b> The system must allow users to use a map service to locate violations	Yes	<b>User Manager</b>
<b>R20:</b> The system must be able to communicate with a map service to locate the areas		

## 2.2 Web application

### 2.2.1 Implemented requirements

Table 2.2 indicates all the functional requirements related to the web application, then it includes the design component responsible for that requirement, and finally, it indicates whether a given requirement was implemented or not.

Table 2.1: Implemented requirements related to the web application

Requirements	status of Implementation	Design components
<b>R13:</b> The system must allow active municipalities to filter data by type, time, date, and location of violation	Yes	<b>Data Manager</b>
<b>R14:</b> The system must allow municipalities to access the data of violation		
<b>R15:</b> The system must be able to communicate with active municipalities to acquire data of accidents		
<b>R22:</b> The system must allow a municipality to activate its account	Yes	<b>Municipality Authentication Manager</b>
<b>R23:</b> The system must allow active municipality to access its account		

<b>R24:</b> The system must allow active municipalities to download generated file of violations	Yes	<b>Data Manager</b>
<b>R25:</b> The system must allow active municipalities to check the suggestions offered by SafeStreets	No	<b>Suggestions Manager</b>

---

## **3. Adopted development frameworks**

---

In this chapter we discuss the frameworks that are used to implement and build the application.

### **3.1 Adopted programming languages**

#### **3.1.1 JAVA**

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. We used Java for the development of the Safestreet mobile app backend.

##### **Advantages**

- Object-Oriented: Allows you to create modular programs and reusable code.
- Platform-Independent: Ability to move easily from one computer system to another
- Distributed: Designed to make distributed computing easy with the networking capability that is inherently integrated into it.
- Secure: The Java language, compiler, interpreter, and runtime environment were each developed with security in mind.
- Allocation: Java has the feature of Stack allocation system. It helps the data to be stored and can be restored easily.
- Multi threaded: The capability for a program to perform several tasks simultaneously within a program.

## **Disadvantages**

- Performance: Significantly slower and more memory-consuming than natively compiled languages such as C or C++.
- Look and feel: The default look and feel of GUI applications written in Java using the Swing toolkit is very different from native applications.
- Single-paradigm language: The addition of static imports in Java 5.0 the procedural paradigm is better accommodated than in earlier versions of Java.

### **3.1.2 PHP**

PHP is a server-side scripting language designed for web development. We used PHP for development of the backends/database access for the Safestreet mobile app and Safestreet web app.

## **Advantages**

- Open source: It is developed and maintained by a large group of developers. This will help in creating a support community and abundant extension libraries.
- Speed: It is relatively fast, since it uses not much system resources.
- Stable: Since it is maintained by many developers, bugs are found and fixed quickly, making it a stable software.
- Powerful library support: You can easily find functional modules you need such a PDF, graph etc.
- Built in database connection modules: You can connect to databases easily using PHP, since many websites are data/ content driven, so we will use database frequently, this will largely reduce the development time of web apps.
- Security: It offers security that can prevent malicious attacks. This can be adjusted for example in the .ini file.

## **Disadvantages**

- Weak type: Implicit conversion may surprise unwary programmers and lead to unexpected bugs. Confusion between arrays and hash tables. This is slow and could be faster. There are often a few ways to accomplish a task. It is not strongly typed. It is interpreted and uses curly braces.
- Poor Error Handling Method: The framework has a bad error handling method. It is not a proper solution for the developers. Therefore, as a qualified PHP developer, you will have to overcome it.
- PHP is unable to handle large number of apps: The technology is helpless to support a bunch of apps. It is highly tough to manage because, it is not competent modular. It already imitates the features of Java language.

## **3.1.3 HTML5**

HTML5 is a software solution stack that defines the properties and behaviors of web page content by implementing a markup based pattern to it. We used HTML5 for the development of the frontend of the Safestreet web application.

## **Advantages**

- HTML5 isn't a proprietary code: you are not required to pay royalties if you decide to use HTML5 for your website.
- The coding with HTML5 is clear and consistent: it is simple, straight-forward, and very easy to read. You can quickly separate the content from the style, making it easier to compose code that is descriptive and clear.
- HTML5 requires less maintenance than other options: it utilizes an open-source programming language that is almost universally known. That means you can find the support you need for troubleshooting online on your own.

- The storage options with HTML5 are more reliable: With HTML5, you have the ability to store user-side data temporarily within a SQL database.
- All compatible browsers collect and use data: when you're using HTML5 from a mobile perspective, you still have the ability to collect useful data, collate it, and then use it to reach your metrics and goals.

## **Disadvantages**

- It requires modern browsers to access it: if you have users trying to access your website through an older browser, then you're not going to be able to reach them. There are media licensing issues which must be considered.
- Multiple device responsiveness can be a headache

### **3.1.4 Javascript**

JavaScript, is a high-level, just-in-time compiled, multi-paradigm programming language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. We used Javascript for the development of the frontend of the Safestreet web application.

## **Advantages**

- Speed: Client-side JavaScript is very fast because it can be run immediately within the client-side browser.
- Simplicity: JavaScript is relatively simple to learn and implement.
- Popularity: JavaScript is used everywhere in the web.
- Interoperability: JavaScript plays nicely with other languages and can be used in a huge variety of applications.
- Server Load: Being client-side reduces the demand on the website server.

## **Disadvantages**

- **Client-Side Security:** Because the code executes on the users' computer, in some cases it can be exploited for malicious purposes.
- **Browser Support:** JavaScript is sometimes interpreted differently by different browsers.

### **3.1.5 XML**

Extensible Markup Language is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. We used XML for the development of the frontend of the Safestreet mobile application.

## **Advantages**

- XML is platform independent and programming language independent, thus it can be used on any system and supports the technology change when that happens.
- XML supports unicode. Unicode is an international encoding standard for use with different languages and scripts. This feature allows XML to transmit any information written in any human language.
- The data stored and transported using XML can be changed at any point of time without affecting the data presentation.
- XML allows validation using DTD and Schema. This validation ensures that the XML document is free from any syntax error.
- XML simplifies data sharing between various systems because of its platform independent nature.

## **Disadvantages**

- XML syntax is verbose and redundant compared to other text-based data transmission formats such as JSON.
- The redundancy in syntax of XML causes higher storage and transportation cost when the volume of data is large.
- XML document is less readable compared to other text-based data transmission formats such as JSON.
- XML doesn't support array.
- XML file sizes are usually very large due to its verbose nature, it is totally dependant on who is writing it.

## **3.2 Adopted middle-wares**

### **3.1.2 Android Studios IDE**

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. We used android studios to develop both the front and backends of the Safestreet mobile app.

## **Advantages**

- Easy to use: as the official and native android development platform the user interface is much simpler to use.
- Great autocomplete features
- Has a drag and drop GUI
- Lighter requirements compared to Eclipse
- Better app testing available compared to Eclipse



## Disadvantages

- Still higher performance requirements than coding without an IDE

### 3.2.2 XAMPP

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. We used Xampp to host the Safestreet server on a local machine which we ran the web app on. Furthermore, it was used to create and access the local database which both the web app and mobile app had access to.

## Advantages

- It keeps developers in mind.
- Easy to install and set up.
- Provides a GUI based control panel for starting and stopping modules/services and updating the config files.

## Disadvantages

- No disadvantages as far we are concerned

## 3.3 Additional APIs

**Google Places** is the only external API that was used in the Safestreet mobile and web apps. It was used in order to provide the users with autofill options when searching locations.

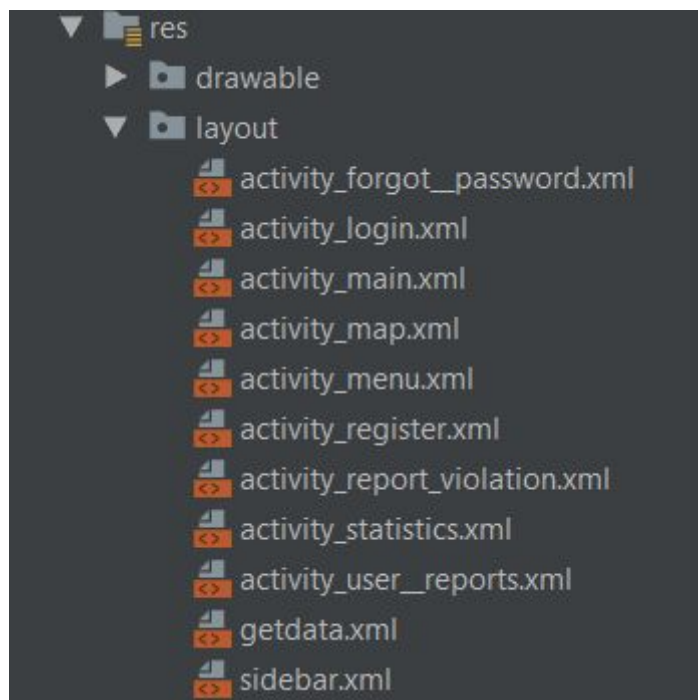
---

## 4. Structure of the source code

---

### 4.1 Mobile application front-end project structure

We used android studios for the development of the front end. The structure of the mobile app is depicted by the images below.



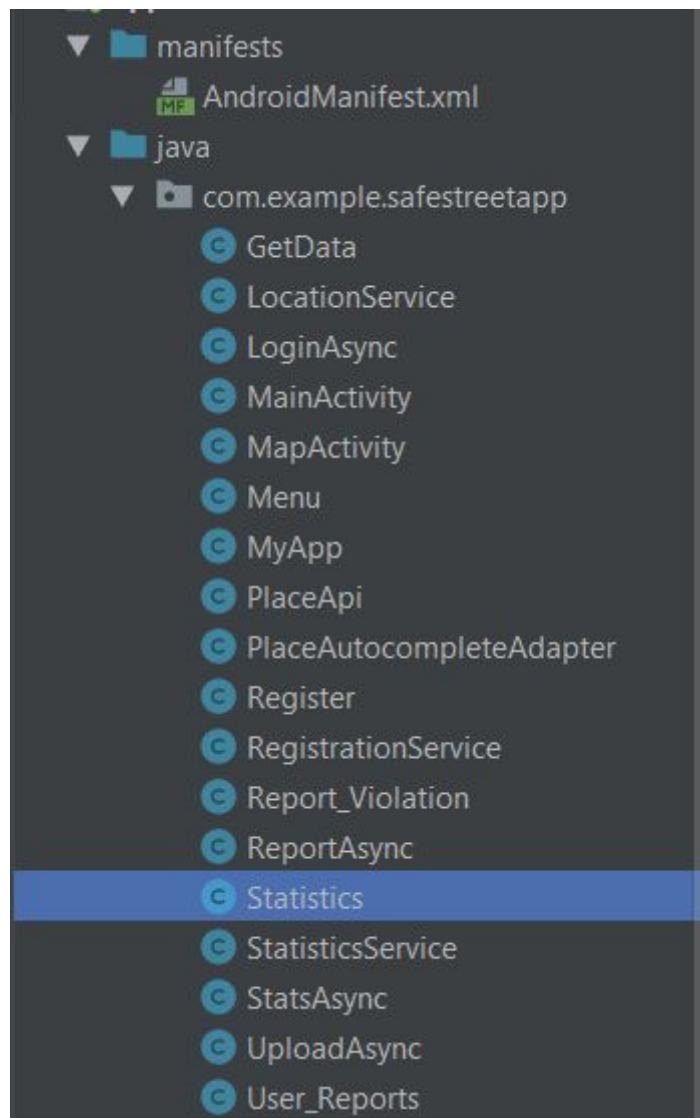
The res folder contains the frontend folders. The drawables folder contains images that we use within the apps user interface. The layout folder contains the layout code for each activity in the mobile application.

## 4.2 Mobile application back-end project structure

The mobile apps backend was handled by both android studios using java classes and PHP scripts for server and database communication.

### 4.2.1 Java classes

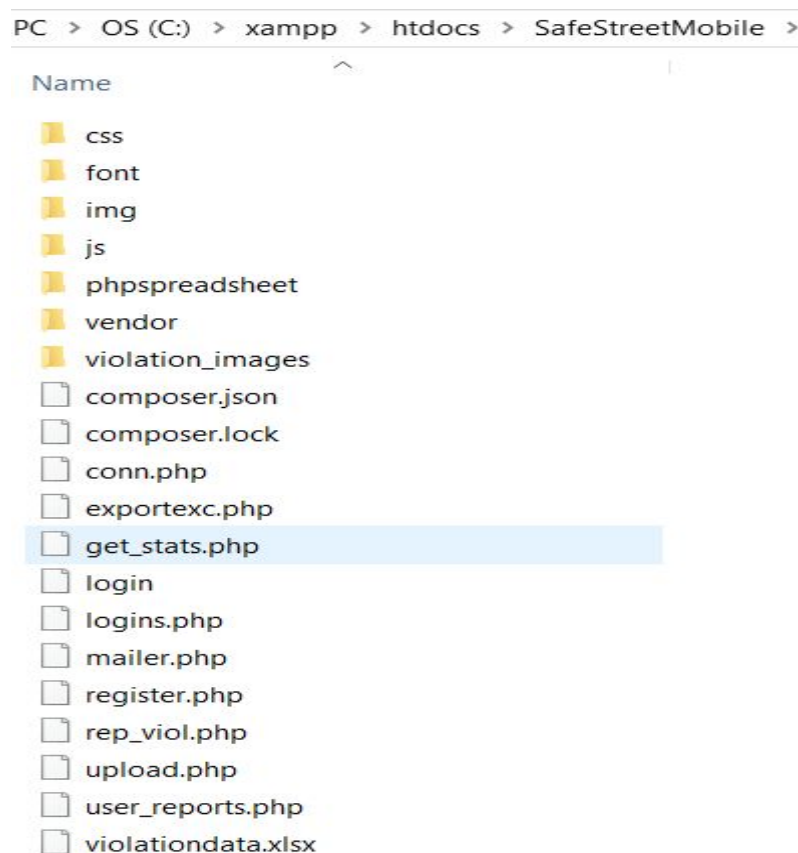
The image below depicts the structure of the java classes used for the backend.



The java folder holds the safestreet package which in turn holds all the java classes used for the backend development of each activity in the res folder as well as numerous AsynTasks used to access the PHP scripts for communication with the server and database.

### 4.2.1 PHP Scripts

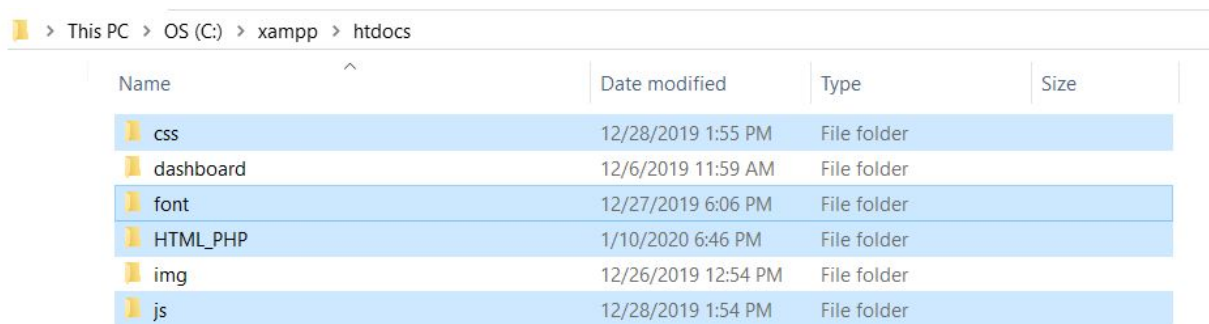
The php scripts are stored on the server side in htdocs and communicate back and forth between the server and database. The image below depicts the PHP scripts structure.



The scripts are stored on the Xampp server inside the SafeStreetMobile folder inside the htdocs folder which is the default server data folder.

## 4.3 Web-application front-end project structure

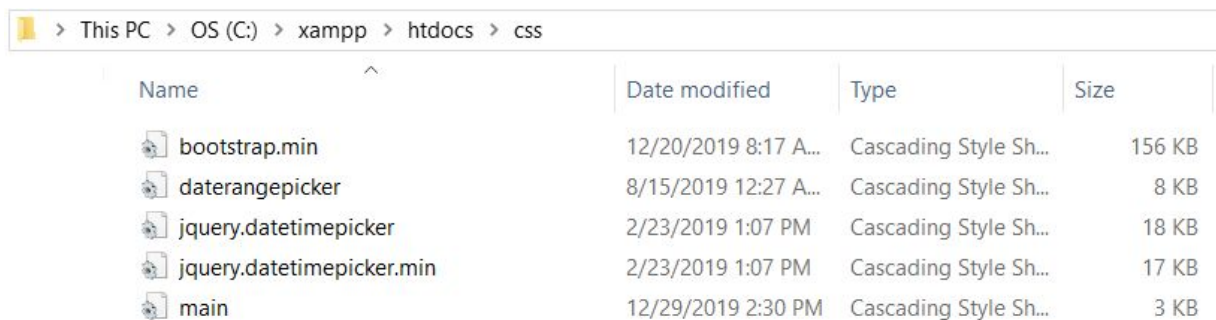
The frontend of the Safestreet app designed using HTML5, CCS, and Javascript and the files are stored on the local server inside the htdocs folder. The images below depict the structure of the frontend.



Name	Date modified	Type	Size
css	12/28/2019 1:55 PM	File folder	
dashboard	12/6/2019 11:59 AM	File folder	
font	12/27/2019 6:06 PM	File folder	
HTML_PHP	1/10/2020 6:46 PM	File folder	
img	12/26/2019 12:54 PM	File folder	
js	12/28/2019 1:54 PM	File folder	

The highlighted folders hold all the frontend folders and their contents and structure are depicted below.

- CSS - cascading stylesheets used to formating and styling part of the front end



Name	Date modified	Type	Size
bootstrap.min	12/20/2019 8:17 A...	Cascading Style Sh...	156 KB
daterangepicker	8/15/2019 12:27 A...	Cascading Style Sh...	8 KB
jquery.datetimepicker	2/23/2019 1:07 PM	Cascading Style Sh...	18 KB
jquery.datetimepicker.min	2/23/2019 1:07 PM	Cascading Style Sh...	17 KB
main	12/29/2019 2:30 PM	Cascading Style Sh...	3 KB

- **Javascript** - scripts used for the cosmetics of the pie chart and datetime pickers.

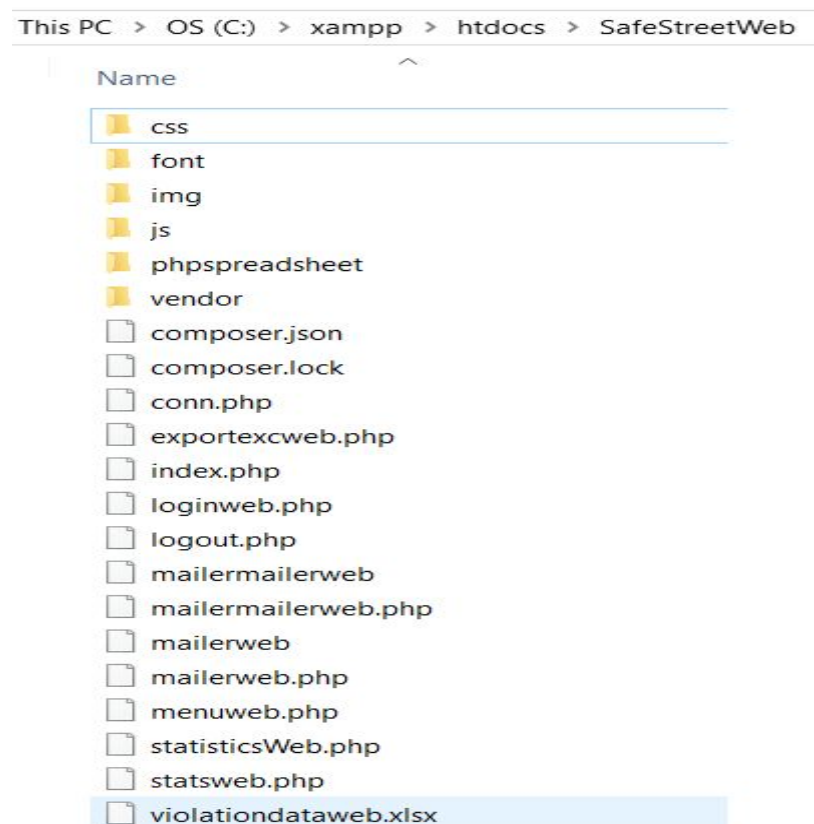
This PC > OS (C:) > xampp > htdocs > js				
Name	Date modified	Type	Size	
bootstrap.min	12/20/2019 8:17 A...	JavaScript File	59 KB	
daterangepicker	8/15/2019 12:27 A...	JavaScript File	65 KB	
jquery.datetimepicker.full.min	2/23/2019 1:07 PM	JavaScript File	60 KB	
jquery.datetimepicker	2/23/2019 1:07 PM	JavaScript File	93 KB	
jquery.min	12/20/2019 8:17 A...	JavaScript File	87 KB	
mdb.min	12/20/2019 8:17 A...	JavaScript File	280 KB	
moment.min	8/15/2019 12:27 A...	JavaScript File	53 KB	
popper.min	12/20/2019 8:17 A...	JavaScript File	21 KB	
toggle	12/26/2019 12:41 ...	JavaScript File	1 KB	

- **HTML5** - used for creating buttons and forms and general structure of each page. The files are php files because they include php scripts in them that use data from the forms input by the user to access the database or server files.

This PC > OS (C:) > xampp > htdocs > SafeStreetWeb	
Name	
css	
font	
img	
js	
phpspreadsheet	
vendor	
composer.json	
composer.lock	
conn.php	
exportexcweb.php	
index.php	
loginweb.php	
logout.php	
mailermailerweb	
mailermailerweb.php	
mailerweb	
mailerweb.php	
menuweb.php	
statisticsWeb.php	
statsweb.php	
violationdataweb.xlsx	

## 4.4 Web-application back-end project structure

The backend of the Safestreet web app uses PHP scripts that are integrated within the HTML5 for ease of use and are therefore in the same folder, inside htdocs inside the SafeStreetWeb folder. The structure is depicted below.



---

## 5. Test Plan

---

### 5.1 Test scenarios

#### 5.1.1 Mobile application test scenarios

Here we include a list of test scenarios to test the correct functionality of the mobile application. The frontend and the backend will be tested. Table 5.1 summarizes the list of test scenarios we will apply on the software.

Table 5.1: Test scenarios for the mobile application

Test ID	Description	Importance
T0	login functionality test	High
T1	registration functionality test	High
T2	reporting violation functionality test	High
T3	access data functionality test	Medium
T4	see statistics functionality test	Medium
T5	My reports functionality test	Low



## 5.1.2 Web application test scenarios

Here we include a list of test scenarios to test the correct functionality of the web application. The frontend and the backend will be tested. Table 5.2 summarizes the list of test scenarios we will apply on the web application.

Table 5.2: Test scenarios for the web application

Test ID	Description	Importance
T6	activation functionality test	High
T7	login functionality test	High
T8	accessing data functionality test	Medium
T9	see statistics functionality test	Medium

## 5.2 Test cases

### 5.2.1 Mobile application test cases

#### Login test cases

- **A.1 Login**

Status: Success

Test ID: To

Prerequisite: existing account

Expected output: login successful

Steps:

1. Launch application
2. Fill the field Email with a valid email
3. Fill the field Password with a valid password

4. Press Login button  
Actual output: Login successful

- **A.2 Login**

Status: fail

Test ID: To

Prerequisite: none

Expected output: login invalid

Steps:

1. Launch application
2. Fill the field Email with a invalid email or password
3. Press Login button

Actual output: Wrong username and password combination

- **A.3 Login**

Status: fail

Test ID: To

Prerequisite: none

Expected output: login invalid

Steps:

1. Launch application
2. Keep the field of email and password empty
3. Press Login button

Actual output: please enter username

- **A.4 Login**

Status: fail

Test ID: To

Prerequisite: none

Expected output: login invalid

Steps:

1. Launch application
2. Fill username and Keep the field of password empty
3. Press Login button

Actual output: please enter password

## **Registration test cases**

- **B.1 Registration**

Status: success

Test ID: T2

Prerequisite: none

Expected output: registration successful

Steps:

1. Launch application
2. Fill the fields of registration with valid inputs (Username, phone number, email, password, confirm password)
3. Press Register button

Actual output: Congratulations! You have registered successfully

- **B.2 Registration**

Status: fail

Test ID: T2

Prerequisite: none

Expected output: red mark at the empty field

Steps:

1. Launch application
2. Fill the fields of registration with valid inputs (Username, phone number, email, password, confirm password)
3. Leave one of the fields empty
3. Press Register button

Actual output: enter the empty field

- **B.3 Registration**

Status: fail

Test ID: T2

Prerequisite: already existed account with same email

Expected output: an account already exist with this email

Steps:

1. Launch application
  2. Fill the fields of registration with valid inputs (Username, phone number, email, password, confirm password)
  3. Press Register button
- Actual output: an account already exist with this email

- **B.4 Registration**

Status: fail

Test ID: T2

Prerequisite: already existed account with same username

Expected output: an account already exist with this username

Steps:

1. Launch application
  2. Fill the fields of registration with valid inputs (Username, phone number, email, password, confirm password)
  3. Press Register button
- Actual output: an account already exist with this username

- **B.5 Registration**

Status: fail

Test ID: T2

Prerequisite: none

Expected output: passwords don't match

Steps:

1. Launch application
  2. Fill the fields of registration with valid inputs (Username, phone number, email, password, confirm password)
  3. Fill different passwords in the password and confirm password fields
  3. Press Register button
- Actual output: passwords don't match

## **Report violation test cases**

- **C.1 Report violation**

Status: success

Test ID: T3

Prerequisite: existing account

Expected output: violation uploaded successfully

Steps:

1. Launch application
2. Login with valid username and password
3. Press Report violation button
4. Allow the app to use camera, location
5. Fill in the required fields correctly (location, date, time) and upload the image from safestreet's directory
6. Press Report violation

Actual output: violation uploaded successfully

- **C.2 Report violation**

Status: fail

Test ID: T3

Prerequisite: existing account

Expected output: please enter the location

Steps:

1. Launch application
2. Login with valid username and password
3. Press Report violation button
4. Allow the app to use camera, location
5. Fill in the required fields correctly (location, date, time) but leave the location field empty
6. Press Report violation

Actual output: please enter the location

- **C.3 Report violation**

Status: fail

Test ID: T3

Prerequisite: existing account

Expected output: please upload image of violation

Steps:

1. Launch application
2. Login with valid username and password
3. Press Report violation button

4. Allow the app to use camera, location
5. Fill in the required fields correctly (location, date, time, image) but leave the image field empty
6. Press Report violation

Actual output: please upload image of violation

- **C.4 Report violation**

Status: fail

Test ID: T3

Prerequisite: existing account

Expected output: sending report failed

Steps:

1. Launch application
2. Login with valid username and password
3. Press Report violation button
4. Allow the app to use camera, location
5. Fill in the required fields correctly (location, date, time, image) but upload image from the gallery directory field empty
6. Press Report violation

Actual output: For security reasons, please upload the image from SafeStreets directory

## **Access data test cases**

- **D.1 access data**

Status: success

Test ID: T3

Prerequisite: existing account

Expected output: data sent to email

Steps:

1. Launch application
2. Login with valid username and password
3. Press access data button

Actual output: the data has been sent to your registered email address

## **see statistics test cases**

- **E1 see statistics**

Status: success

Test ID: T4

Prerequisite: existing account

Expected output: a chart summarizing list of registered violations in the selected street

Steps:

1. Launch application
2. Login with valid username and password
3. Press see statistics button
4. Fill the street name in the location filed
5. Fill in the data and time of the violation (optional)
5. Press see statistics

Actual output: chart summarizing violations according to violation type

- **E2 see statistics**

Status: fail

Test ID: T4

Prerequisite: existing account

Expected output: please enter the location

Steps:

1. Launch application
2. Login with valid username and password
3. Press see statistics button
4. Leave the street name in the location filed
5. Fill in the data and time of the violation (optional)
5. Press see statistics

Actual output: please enter the location

## **My report test cases**

- **F1 my report**

Status: success

Test ID: T5

Prerequisite: existing account

Expected output: list of reported violation by user

Steps:

1. Launch application
2. Login with valid username and password
3. Press my report button

Actual output: table summarizing violations

## **5.2.2 Web application test cases**

### **Activation test cases**

- **G.1 Account Activation**

Status: Success

Test ID: T6

Prerequisite: have a municipality email

Expected output: Activation successful.

Steps:

1. Launch web application
2. Fill the field Email in the “opt in” box with **valid activation** email
3. Press Opt in

Actual output: Activation successful. Email with username and password sent to the email



- **G.2 Account Activation**

Status: fail

Test ID: T6

Prerequisite: none

Expected output: Activation failed.

Steps:

1. Launch web application
2. Fill the field Email in the “opt in” box with **invalid** email
3. Press Opt in

Actual output: You have either already opted in or this email does not belong to the list of official municipalities

- **G.3 Account Activation**

Status: fail

Test ID: T6

Prerequisite: Email already opted

Expected output: Activation failed.

Steps:

1. Launch web application
2. Fill the field Email in the “opt in” box with **valid activation** email
3. Press Opt in

Actual output: You have either already opted in or this email does not belong to the list of official municipalities

## **Login test cases**

- **H.1 Login**

Status: Success

Test ID: T7

Prerequisite: existing account

Expected output: login successful

Steps:

1. Launch application
2. Fill the field Email with a valid email

3. Fill the field Password with a valid password sent through email
4. Press Login button

Actual output: Login successful

- **H.2 Login**

Status: fail

Test ID: T7

Prerequisite: none

Expected output: login invalid

Steps:

1. Launch application
2. Fill the field Email with a invalid email or password
3. Press Login button

Actual output: You have either not opted in or have typed in the incorrect email and password

- **H.3 Login**

Status: fail

Test ID: T7

Prerequisite: none

Expected output: login invalid

Steps:

1. Launch application
2. Keep either the field of email or password empty
3. Press Login button

Actual output: Both email address and password are required

## **Access data test cases**

- **I.1 access data**

Status: success

Test ID: T8

Prerequisite: Account activated

Expected output: The requested file has been sent to your registered email!

Steps:

1. Launch application
2. Login with email and password sent to email
4. Filter data by location, date, and time
3. Press access data button

Actual output: the data has been sent to your registered email address

## **See statistics test cases**

- **J.1 access data**

Status: success

Test ID: T9

Prerequisite: Account activated

Expected output: chart summarizing violations according to violation type

Steps:

1. Launch application
2. Login with email and password sent to email
4. Filter data by location, date, and time
3. Press update chart button

Actual output: chart summarizing violations according to violation type

- **J.1 access data**

Status: fail

Test ID: T9

Prerequisite: Account activated

Expected output: chart summarizing violations according to violation type

Steps:

1. Launch application
2. Login with email and password sent to email
4. Filter data by location, date, and time
3. Press update chart button

Actual output: chart summarizing violations according to violation type

## **5.3 Integration strategy**

Section 5.3 of the DD documents present the description of the integration strategy followed while implementing the system. The strategy addresses the system as a whole while in the implementation we did leave some functional requirements unimplemented, like for example the notification part or the security manager at the application server side. Please refer to the DD document for further details

---

## 6. Installation Instructions

---

### 6.1 Installing the system

#### 6.1.1 Mobile and Web application installation

1. Download and install the latest version of Android Studios
2. Download and install the latest version of Xampp
3. Download and install composer from <https://getcomposer.org/download/>
4. [https://mega.nz/#!oZFkwCxD!we2vxuyycnPxHkor4yIrx3Mwv\\_x64BJ-9IHm-eEH-w](https://mega.nz/#!oZFkwCxD!we2vxuyycnPxHkor4yIrx3Mwv_x64BJ-9IHm-eEH-w) and download the SafeStreet.zip file, password for downloading is SW22019
5. Navigate to C: > Xampp > htdocs and unzip the folder there
6. Open up the Xampp control panel and start Apache and MySQL
7. Open your web browser and navigate to <http://localhost/phpmyadmin/>
8. Create a new Database called “safestreet”, select it and import the safestreet.sql file from the htdocs folder
9. Unzip the Safestreet.zip folder inside htdocs
10. Now open up Android Studios and Click on Import Project, select the SafeStreet Gradle Project inside the SafeStreetApp from htdocs and let it import
11. Now install an emulator and run project to run and test the Mobile app
12. Navigate to <http://localhost/SafeStreetWeb/index.php> to run and test the Web app
13. For Municipality activation use one of the following <https://docs.google.com/spreadsheets/d/17rBztF7xLT7shIKcpb2VC5AxtuezHDIhgJKMAgWJoFI/edit?usp=sharing&fbclid=IwAR29jG-mZpoLb2VsuBh4p2WlT6BiGL6BecQwNuT5PqoSUmR0jNfnxRWI7mM>

## **6.2 Troubleshooting**

- If you have trouble with android studios HAXM error while running the emulator watch <https://www.youtube.com/watch?v=K9QG8VMxpt8> for the solution.
- If your Internal Storage is hidden, make sure you make it visible on you emulator (top right toggle menu) when uploading an image for violations.

## **6.2 Working on the system**

### **6.2.1 Mobile application installation**

You can work on the Mobile application through android studios, by navigating through the Java or XML files according to what you would like to work on. If you would like to edit the scripts for DB or server access open up the php files in the htdocs/SafeStreetMobile folder using any word editor.

### **6.2.2 Web application installation**

You can work on the Web application by opening up the php, css , javascript files found in the htdocs/SafeStreetWeb directory using any word editor.

## 7. Effort Spent

Taras Dlamini		Abdullah Quran		Hesham Abdelaziz	
Task	Hours	Task	Hours	Task	Hours
Mobile app development	45	Layout	3	User UI design	10
Web app development	15	Purpose and scope	2	Mobile app development	20
Structure of the code	3	Implemented Requirements	3	Web app development	10
Adopted frameworks	2	Testing mobile application	7		
		Testing web application	5		
		Mobile app development with external services	15		
		Installation Instruction	3		
Total	65	Total	33	Total	40