



POLITECNICO

MILANO 1863

Department of Computer Science and Engineering

Requirement Analysis and Specification Document (RASD)

SafeStreet
- v1.1 -

Authors:

Dlamini, Taras	10451348
Quran, Abdullah	10657807
Abdelaziz, Hesham	10654249

November 10th, 2019

Table of Content

1.Introduction	3
1.1 Purpose	3
1.1.1 General Purpose	3
1.1.2 Goals	3
1.2 Scope	4
1.2.1 Description of Problem	4
1.2.2 World and Shared Phenomena	4
1.3 Definitions, acronyms, abbreviations	5
1.3.1 Definitions	5
1.3.2 Acronyms	7
1.3.3 Abbreviations	7
1.4 Revision History	8
1.5 Referenced Documents	8
1.6 Document Structure	8
2. Overall Description	9
2.1 Product Perspective	9
2.2 Product Function	13
2.3 User Characteristics	14
2.4 Assumptions, Dependencies, Constraints	14
2.4.1 Domain Assumptions	14
3. Specific Requirements	15
3.1 External Interface Requirements	15
3.1.1 Regular User Interfaces	15
3.1.2 Municipality Interfaces	17
3.1.3 Hardware Interfaces	22
3.1.4 Software Interfaces	22
3.2 Functional Requirements	22
3.2.1 Regular User	22

3.2.2 Municipalities	30
3.2.3 Requirements	35
3.2.4 Traceability Matrix.....	38
3.3 Performance Requirements	41
3.4 Design Constraints	42
3.4.1 Standard Compliance	42
3.4.2 Hardware Limitations	42
3.5 Software system Attributes	42
3.5.1 Reliability	42
3.5.2 Availability	43
3.5.3 Security	43
3.5.4 Maintainability	43
3.5.5 Compatibility	43
4. Alloy.....	44
4.1 Formalizing Essential Requirements.....	44
4.1.1 Purpose of the model.....	44
4.1.2 Alloy Model.....	44
4.1.3 Generated Worlds.....	46
4.2 Formalization of Goal five.....	47
4.1.1 Purpose of the model.....	47
4.1.2 Alloy Model.....	48
4.1.3 Generated Worlds.....	49
5. Effort Spent.....	51

1. Introduction

1.1 Purpose

1.1.1 General Purpose

In the modern era, the availability and use of smart devices is at an alltime high, this has lead to the development of a large number of applications to help people, companies and governments make their daily routines easier and more efficient. The traffic management sector is no exception.

Safestreet is a company that is trying to help, citizens and municipalities, make cities safer and more organized places in terms of traffic accidents and violations. Safestreet uses their application to allow the general public to report traffic violations in terms of type of accident (i.e. double parking), location, and time and date. Furthermore, users can see statistics about the general safety of areas. Safestreet uses their website to allow municipalities to view raw data about violations reported by users (useful for identifying repeat offenders, ticketing, etc..). In addition, municipalities receive feedback from Safestreet about what can be done to improve the safety of identified problem areas in the city.

1.1.2 Goals

[G1]: users can input data of violation to SafeStreet

[G2]: Users can access anonymize data of violations

[G3]: Active municipalities can access data of violations

[G4]: Users can see customized statistics and list of unsafe areas

[G5]: Users can opt to have notifications about the safety of the area they are in

[G6]: Active municipalities can see SafeStreets' suggestions about unsafe areas in their perimeter

1.2 Scope

1.2.1 Description of the problem

SafeStreet 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 Safestreet. Safestreet 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. Safestreet, would crosscheck data with Municipalities to make give the users Safe notifications if they desire. Furthermore, Safestreet 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 Safestreet 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.

1.2.2 World and shared phenomena

Understanding the World and Shared phenomena allow to better define the scope, goals, and product of the system. The entities involved and influence the real world events are reported in Table 1.1 as World Phenomena. interaction is part of a relationship between entities in the real world and Trackme environment are reported in Table 1.1 as shared phenomena.

Table 1.1

Phenomenon	World/Shared
The occurrence of parking violations	World
The occurrence of an accident	World
Users willing to report parking violations through SafeStreets	World
Municipalities willing to use and have SafeStreets' data	World
Municipalities willing to use SafeStreets' data for ticketing	World
Parking violation details (Image, Type, Location, Time) reported through SafeStreets	Shared
Accident details (Type, Location, Time) offered by municipality	Shared
Suggestion about an area generated according to violations and accidents information known by the system	Shared
Status of an area depending on the accident information	Shared
Statistics viewed by users of the application	Shared

1.3 Definitions, acronyms, abbreviations

1.3.1 Definition

Area: In our system we define an area as a street.

Vehicle: Any mode of transportation that has a registered license plate

Parking violation: parking violation in our system is defined as a violation where a vehicle has broken a road rule, out of one of the five predefined categories of parking violations.

Five Categories of Parking Violation:

1. Double parking: Vehicle parking in two separate parking spots.
2. Parking in a bike lane: Vehicle parking in a designated bike lane.
3. Parking in a no parking area : Vehicle parking in a no parking area, i.e. in front of a store.
4. Parking on the Sidewalk : Vehicle parking on the sidewalk where people walk.
5. Parking in a handicapped zone without an appropriate permit.

Severity: how severe the accident is.

Accidents: An accident is defined as a collision between a vehicle and another piece of property (another vehicle, street sign, bicycles, etc...) and is also defined in the scope of five predefined “accident” categories

Five Categories of Accidents:

1. Vehicle rollover: the vehicle rolls over on its side one or multiple times
severity : 10
2. Single vehicle accident: only one vehicle is involved, usually a crash with a street sign, lamp post, etc...
severity : 3
3. Rear-end collision: a vehicle crashes into a vehicle in front of it
severity : 5
4. Side-impact collision: a vehicle crashes into the side of another vehicle
severity : 7
5. Head-on collision: vehicles crash into each other from the front
severity : 8

Safety Index: number of accidents * (sum of severity of accidents/number of accidents)

Safe area: An Area where which is a 25 or less on our safety index, which is calculated by number of accidents and types of accidents.

Unsafe area: An Area where which is a 26 or above on our safety index, which is calculated by number of accidents and types of accidents.

Verified violation: A violation that has been analysed by Safestreet and been approved as a valid violation.

Chain of custody: The order in which the data has been passed down by, from entity to entity.

1.3.2 acronyms

- RASD: Requirement Analysis and Specification Document
- API: Application Programming Interface
- GPS: Global Positioning System
- UI: User Interface
- TP: Third Party
- MTTR: Mean Time To Failure, the length of time a device, system, etc... is expected to last in operation

1.3.3 abbreviations

- [Gn]: n-goal.
- [Dn]: n-domain assumption.
- [Rn]: n-functional requirement.
- [UCn]: n-use case.

1.4 Revision History

- 10/11/2019 - Initial release

1.5 Referenced Document

- Specification document: “SafeStreets Mandatory Project Assignment AY 2019-2020”
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications

1.6 Document Structure

The document is divided into four parts, each part is devoted to approach each one of the steps required to apply requirements engineering techniques.

Chapter **one** gives introduce the problem in question and describes the purpose of the application of SafeStreets. It also summarizes the goals to be achieved of the software application and describes the world and shared phenomenon.

Chapter **two** presents the overall description of the project. The product perspective includes details on the user characteristics and the assumed domain and constraints.

Chapter **three** describes the external interface requirements, including: user interfaces, hardware interfaces, software interfaces and communication interfaces. The functional requirements are explained using use case and sequence diagrams while The non functional requirements are explained using performance requirements, software system attributes, and design constraints..

Chapter **four** includes the formalization of the most crucial functional requirements using alloy and the discussion of its purpose. Also it include some of the worlds generated in compliance with the requirements.

2. Overall Description

2.1 Product Perspective

Previously we defined the scope of the system, here we will expand on that definition with the use of a class diagram which shows the interaction of the main components of the system as well as state diagrams that help describe the main functions of the system in more detail.

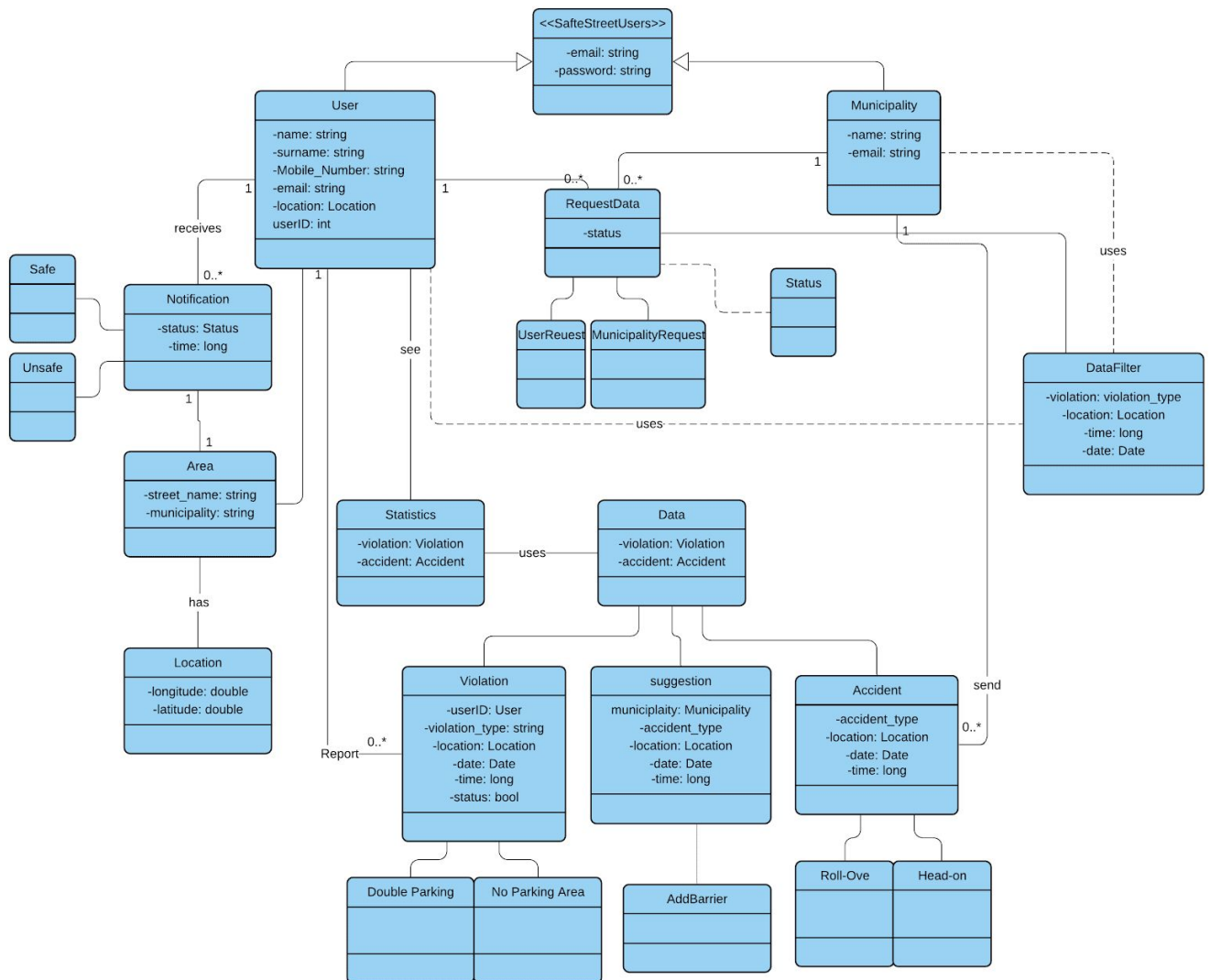


Figure 2.1: SafeStreets' class diagram

Figure 2.1 represents the roughly sketched class diagram which gives an overall description of the system main domain.

Statechart diagrams

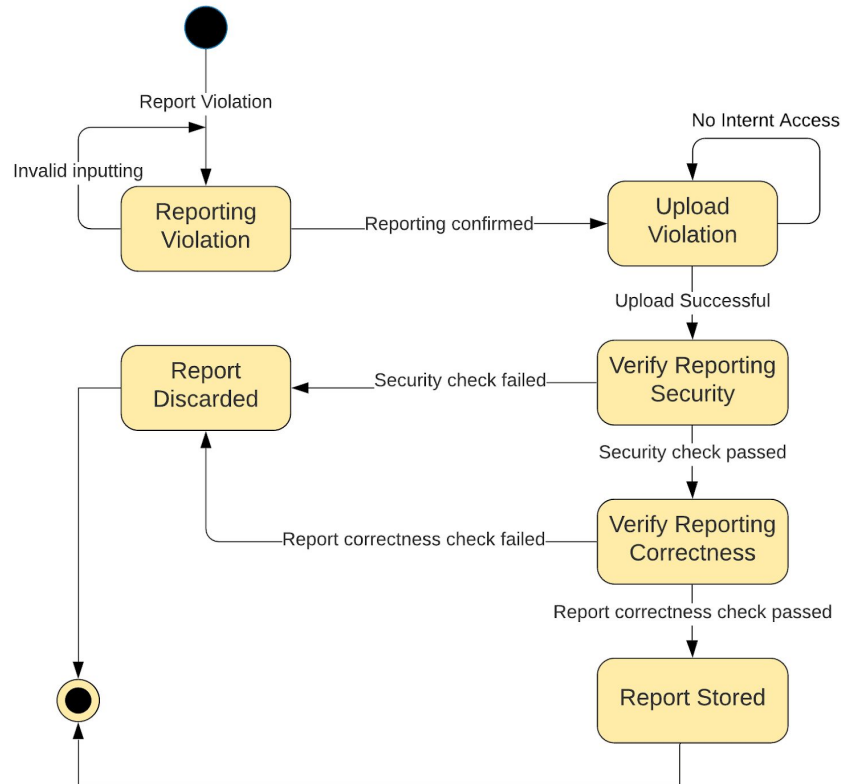
**Figure 2.2: Statechart diagram of reporting a violation**

Figure 2.2 is the representation of the action of reporting a violation and its phases, the user starts of by reporting a violation by filling in all the required fields, if successfully done, he moves on to uploading the violation with the image showing it. Once he has got internet access the upload succeeds and the application checks that the image has not been tampered with. If this fails the report is discarded, otherwise it moves on to verifying that the report and image are clear and that it is indeed a violation. If this check fails again the report is discarded, otherwise the report is stored in the database.

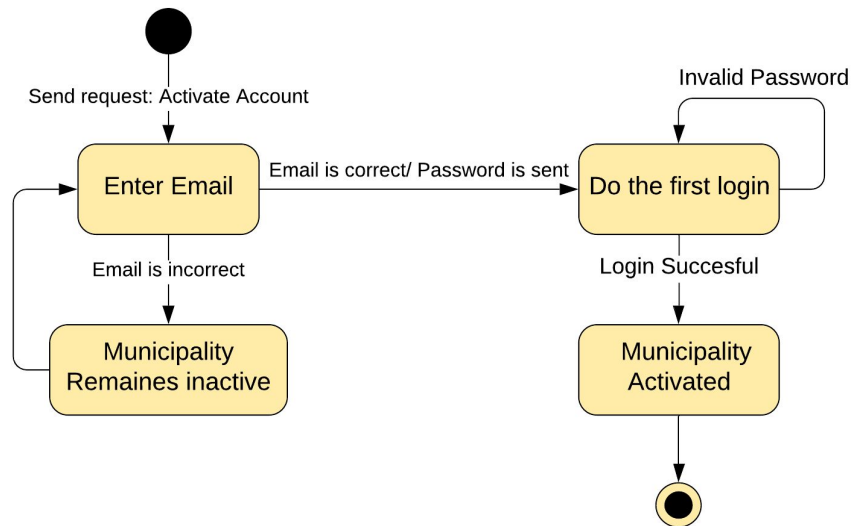


Figure 2.3: Statechart diagram of Activating a municipality

Figure 2.3 is the representation of the activation of a Municipality on the website. The first step is for the municipality to enter their email into the activation form on the home page, this email is cross checked with a list of official municipality emails provided by the government. If the given email is on the list the municipality is sent its login details, otherwise it remains an inactive municipality. Once the Municipality first logs in with the correct given login details, they are activated.

Figure 2.4 is the representation of the notification system of the Safestreet app. Once the notifications have been enabled, the user can see the notification about the safety of the area he/she is in. As soon as new data is received by Safestreet, it is processed and then the safety of an area is redefined and sent as a notification to the user.

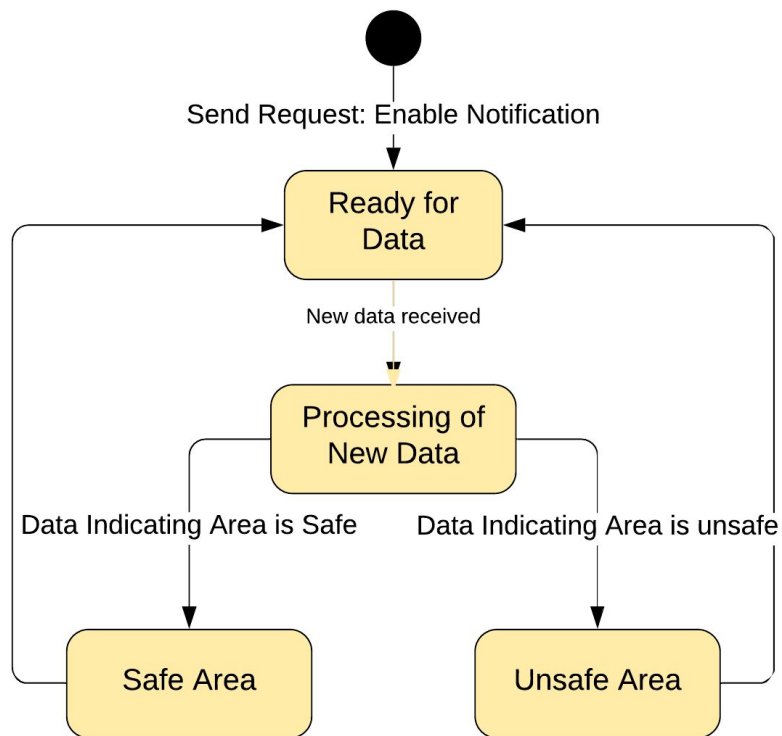


Figure 2.4: Statechart diagram of notification service

2.2 Product Function

The Safestreet system is composed of two main components. The mobile application used by regular users for violation reporting, viewing safety statistics, and receiving raw data for mining purposes. The website used by the municipalities to receive raw data from Safestreet, receive suggestions from Safe Street about how to improve the safety of city based on statics about violations, and finally to send Safestreet data about accidents and ticketed violations. The app and web pages are used to help Safestreet build statistics about the cities it is active in and thus help prevent violations, increase safety, and help municipalities by making their job easier and more efficient when it comes to the traffic management aspect.

Mobile App

The application is designed in order to allow registered users to report violations by taking images of the violation through the app, upload the image, filling in the location, time and date and send it to the Safestreet server for analysis. Once analyzed and determined to be a valid violation, using a license plate detection algorithm, it is stored in the database for statistical purposes. In addition, the regular user is able to receive full anonymized raw data for mining purposes, as well as, view already mined statistics about the safety of an area, at specific times and dates.

Website

The website is designed for municipalities that wish to opt in to the Safestreet service to be able to do so. Safestreet has a list of emails of Municipalities, given to them by the government. Now if a municipality wants to use the Safestreet service they simply fill in that official email into the activation form and receive an email with their login details, if their email matches the one on the list Safestreet has. Once logged in the Municipality is able to receive filtered raw data about violations, which they can use for ticketing purposes, to find repeat offenders, etc... Furthermore, the Municipality is able to send Safestreet info about ticketed violations and accidents, as well as see suggestions from Safestreet about how they can improve specific areas and their safety around the city.

2.3 User Characteristics

The stakeholders of the SafeStreets systems are:

Regular Users:

- Can register and use the SafeStreet application to report parking violations
- Can have access to anonymize samples of parking violations
- Can view statistics and list of unsafe areas SafeStreets offers
- Can opt to have notification about the safety status of their location

Municipalities:

- Can activate their interface with the SafeStreet system
- Can have full access of the data of parking violations
- Can see SafeStreets Suggestions about unsafe areas in their perimeter
- Can send SafeStreets data of accidents occurred in their perimeter

2.4 Assumptions, Dependencies, Constraints

2.4.1 Domain assumptions

Domain assumptions list the assumption that are expected to be in the external environment. This is helpful in defining the domain of interest which the software machine is supposed to operate on. Below the list of the assumption we assume to take place in the real world.

[D1]: The system is authorized to store data of violation in its database

[D2]: User's mobile has enough space to store images of violation

[D3]: Users had allowed SafeStreets application to use storage, camera, internet service, and location

[D4]: SafeStreets is using a secure communication protocol

[D5]: The details entered by users about location is valid

[D6]: The system is authorized to store and analyze the data of accidents received from municipalities

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 Regular User Interfaces

Below are mockups of the user interface of the app for regular users representing the most important functionalities that different users will access.

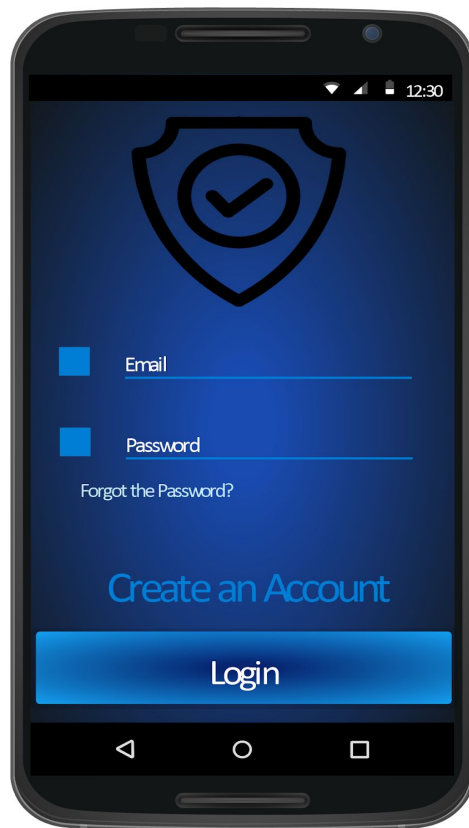


Figure 3.1 - Mockup: Login form

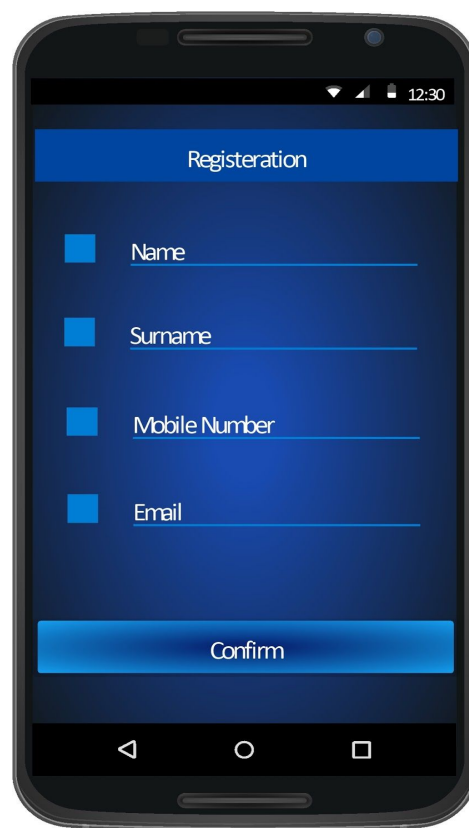


Figure 3.1 - Mockup: Registration form for regular users

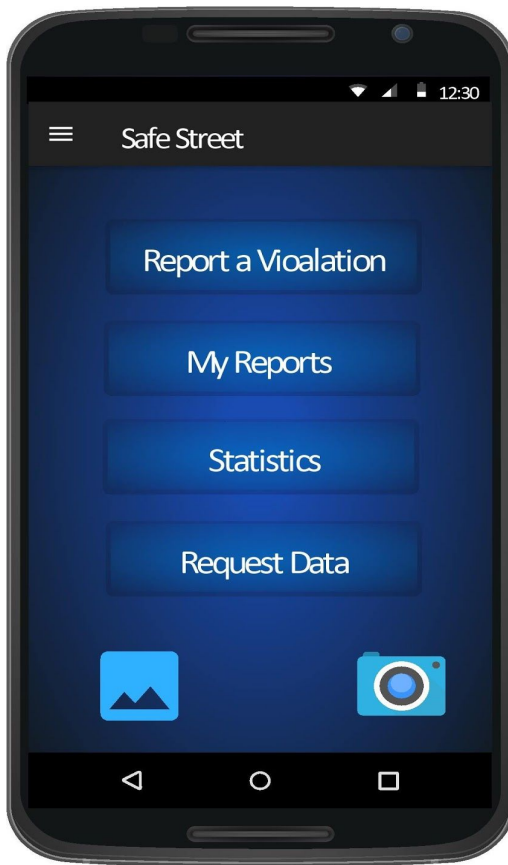


Figure 3.4 - Mockup: Regular user menu Menu

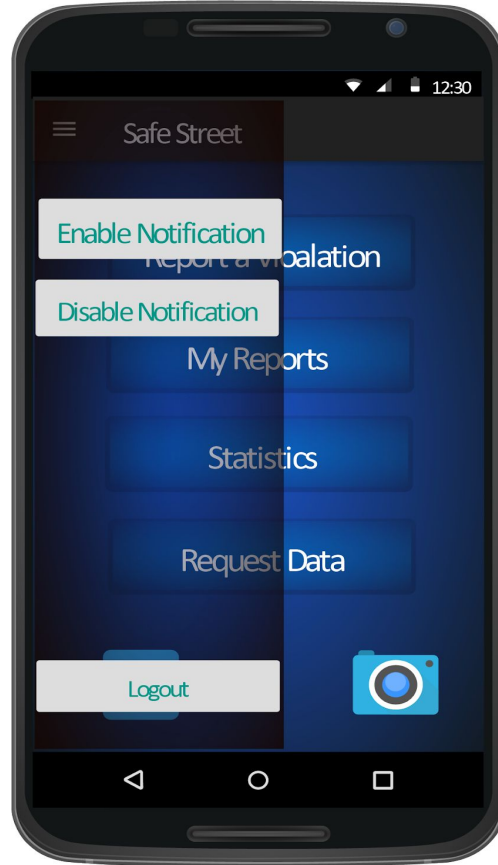


Figure 3.5 - Mockup: Account Settings

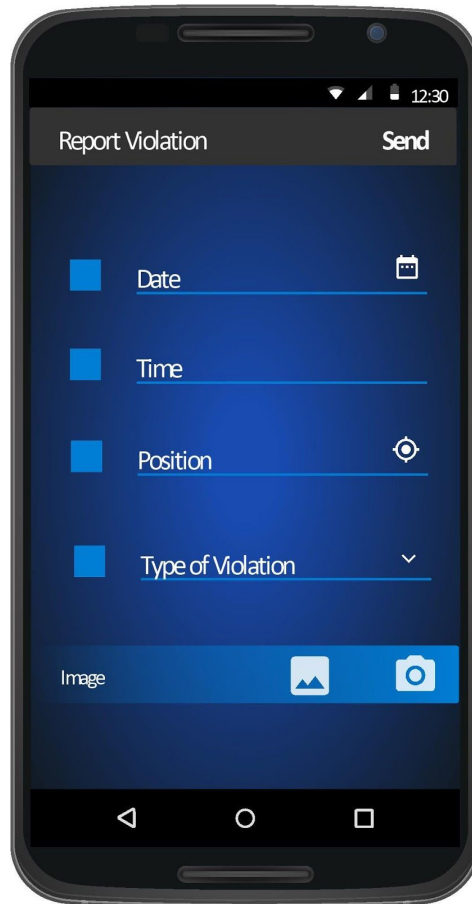


Figure 3.6 - Mockup: Reporting a violation

Figure 1 represents the login screen for all kinds of users (regular and municipalities). Figure 2 represents the registration screen for regular user that have not created an account yet, note that municipalities register through the webpage. Figure 3 represents the menu for regular users where they can choose either to report a violation, to see their reports, to see statistics about violations in general or to get a data for mining purposes sent to their email. Figure 4 represents the pop up settings menu where users can opt to stop receiving or receive again notifications, furthermore they can logout and see account information. Figure 5 represents the regular user screen for violation reporting.

3.1.2 Municipality Interfaces

Below are mockups of the user interface of the website for the municipalities representing the most important functionalities that municipalities will access.

The mockup shows a web interface for 'SafeStreet'. The header is yellow and contains the 'SafeStreet' logo on the left and three links: 'Access Data', 'Suggestion', and 'Log Out' on the right. The main body has a blue background. In the center, there are two white rounded rectangles. The top one is titled 'LOGIN' in blue and contains two input fields: one with an envelope icon for email and one with a padlock icon for password. The bottom one is titled 'ACTIVATION' in blue and contains one input field with an envelope icon for email.

Figure 6 - Mockup: Municipality Login/Activation

Figure 6 represents the screen where a municipality can opt in to the Safestreet initiative by typing in the email in the activation bar, they will receive an email from safe street with the password and they can login on this same screen using their email and password.

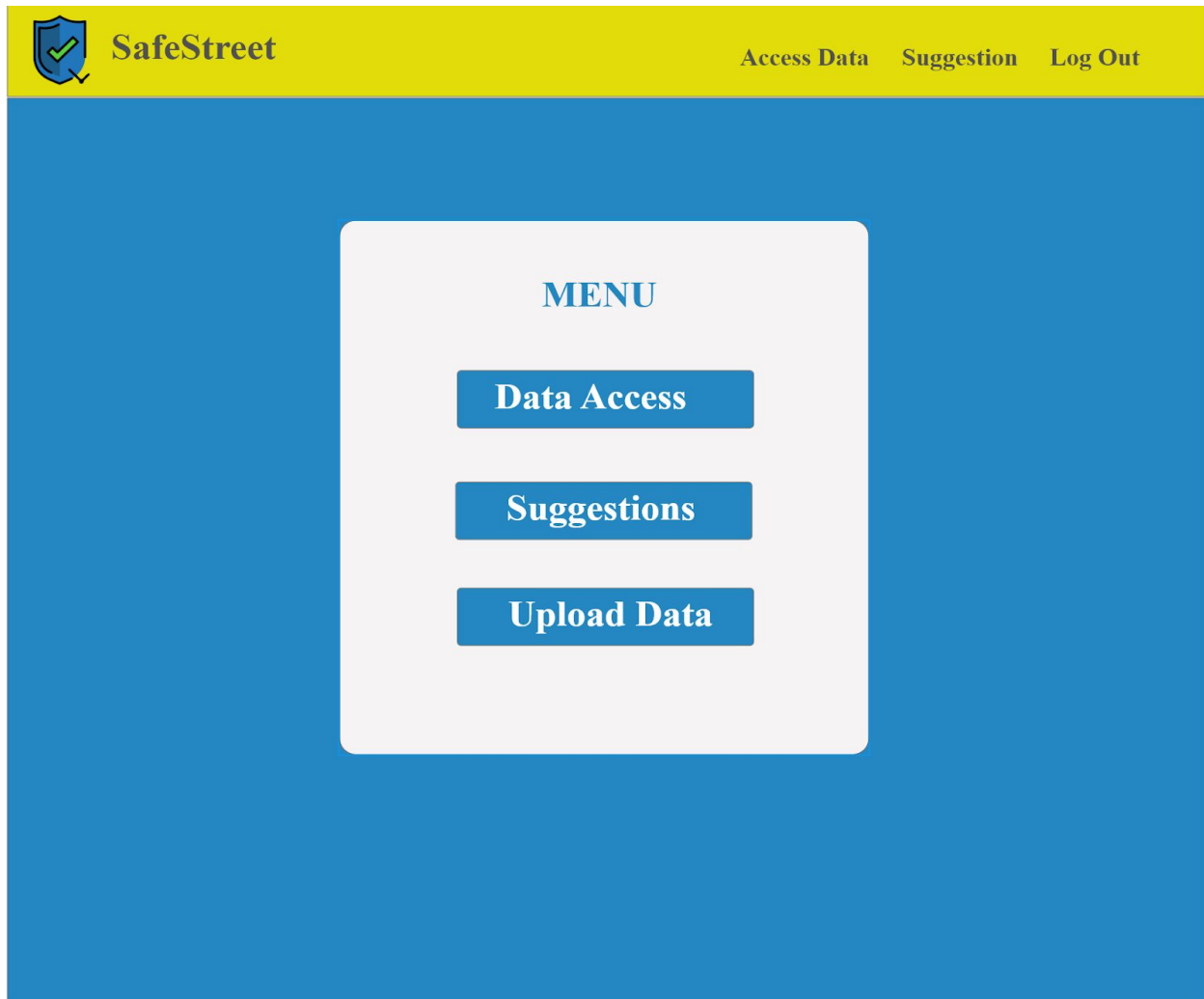
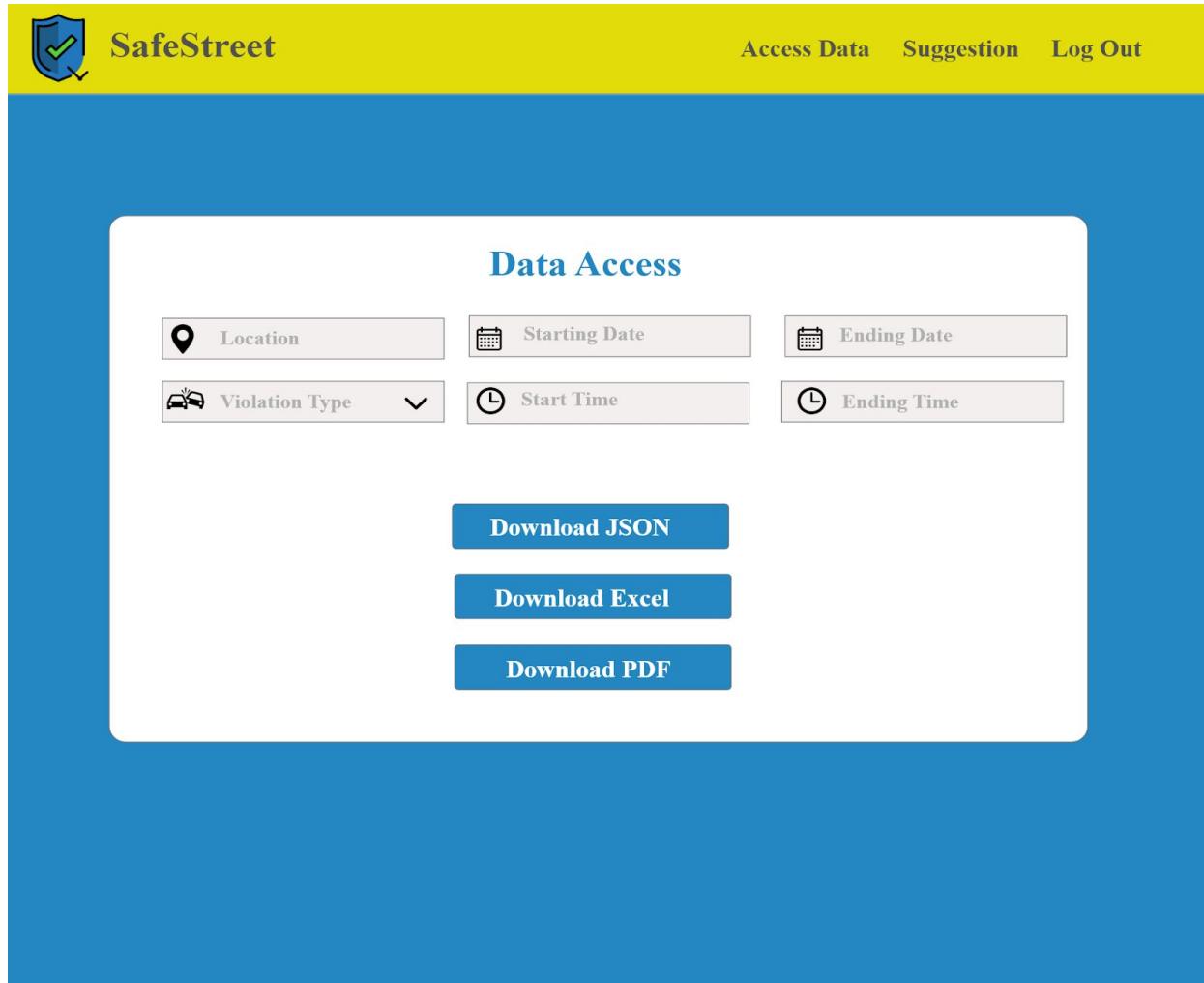


Figure 3.7 - Mockup: Municipality Menu

Figure 7 represents the Menu page where the logged in Municipality can select one of the options, if they click on the upload data, they will get a pop up where they can select an Excel, PDF or DB file to upload. The Data Access and Suggestion buttons will take them to the respective pages, described below.



The image shows a web application interface for 'SafeStreet'. The header is yellow with the 'SafeStreet' logo on the left and navigation links 'Access Data', 'Suggestion', and 'Log Out' on the right. The main content area has a blue background. In the center, there is a white rounded rectangle titled 'Data Access'. Inside this rectangle, there are six input fields arranged in two rows. The first row contains 'Location' (with a location pin icon), 'Starting Date' (with a calendar icon), and 'Ending Date' (with a calendar icon). The second row contains 'Violation Type' (with a car icon and a dropdown arrow), 'Start Time' (with a clock icon), and 'Ending Time' (with a clock icon). Below these fields are three blue buttons stacked vertically: 'Download JSON', 'Download Excel', and 'Download PDF'.

Figure 3.8 - Mockup: Municipality Login/Activation

Figure 8 represents the screen where a municipality can filter data that they would like a report sent in, they have the option of the report being sent in JSON, Excel or PDF file.

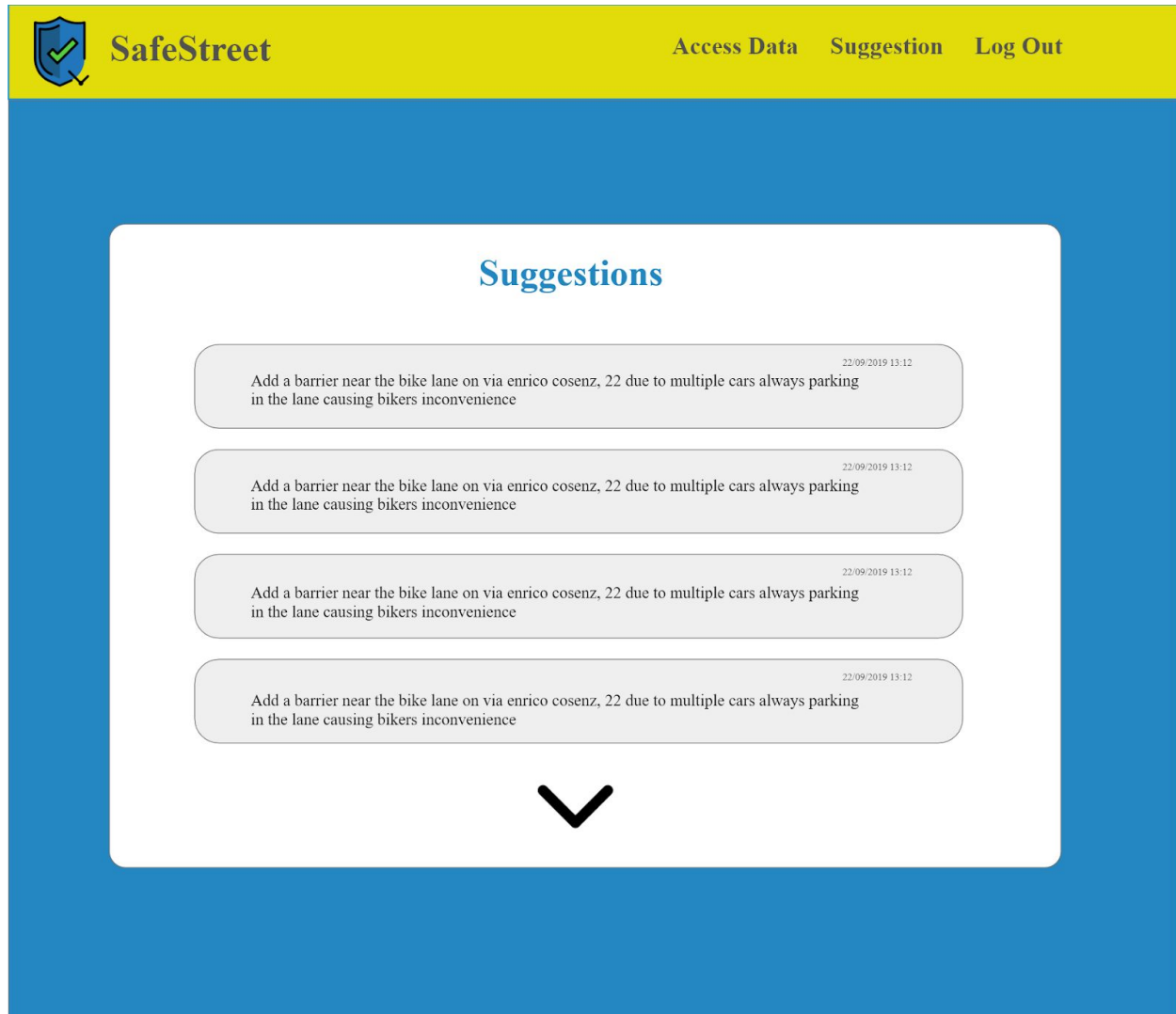


Figure 3.9 - Mockup: Municipality Suggestions

Figure 9 represents the suggestions page where the Municipality can see suggestions from Safestreet about how they can improve certain streets and prevent accidents and violations occurring there.

3.1.3 Hardware Interfaces

No hardware interface is used in the system.

3.1.4 Software Interfaces

The APIs the system uses is to redirect to and from the external service that provides an algorithm for license plate detection. In Addition SafeStreet access google maps when asking the user for a location.

3.2 Functional Requirements

3.2.1 Regular User

Scenario 1

Tom finds that someone parked their car in the bike lane, he opens up his Safestreet applications in order to report the violation. He realizes he does not have internet at the moment, he clicks on the button to take an image through the app and it is saved for later in order for him to upload the image and report the violation when he gains internet access again.

Scenario 2

Mario a shop owner is expecting a delivery for food goods, the delivery truck arrives but there is a car parked in a no parking zone in front of his shop. This is preventing the delivery truck from getting close enough to deliver the goods efficiently. Mario is very frustrated but remembers he heard about an app called Safestreet that lets him report traffic violations. He downloads the app and registers using his name and phone number. He can now login and click the report violation button and fill in the details to report this traffic violation.

Scenario 3

Becky is trying to park her car and sees that the only parking spot available has been double parked in, she decides to report this violation, she opens up the Safestreet app and takes a picture of the double parked car showing the violation and license plate. She has already got a Safestreet account registered so she logs in and clicks on report violation. There she fills in the required fields (location,time, date) and uploads the image. Safestreet then starts processing this violation.

Scenario 4

Elena, a Safestreet app user, is going to an area of town she isn't familiar with. She is not the most experienced driver and wants to see how safe the area is before she decides to go alone. She logs into the Safestreet app and clicks on the see statistics button. She then types in the street name that she will be going to, the application then gives her statistics about the safety of that area from the last week.

Scenario 5

Marco, a PHD student, is writing a report and needs statistics about accidents and violations in his city. He is a Safestreet app user, he logs in to the application and clicks request data, he will now wait for Safestreet to send him an email with up to 100,000 records of reported violation (excluding license plate numbers) for him to use for mining purposes.

Use Case Diagram

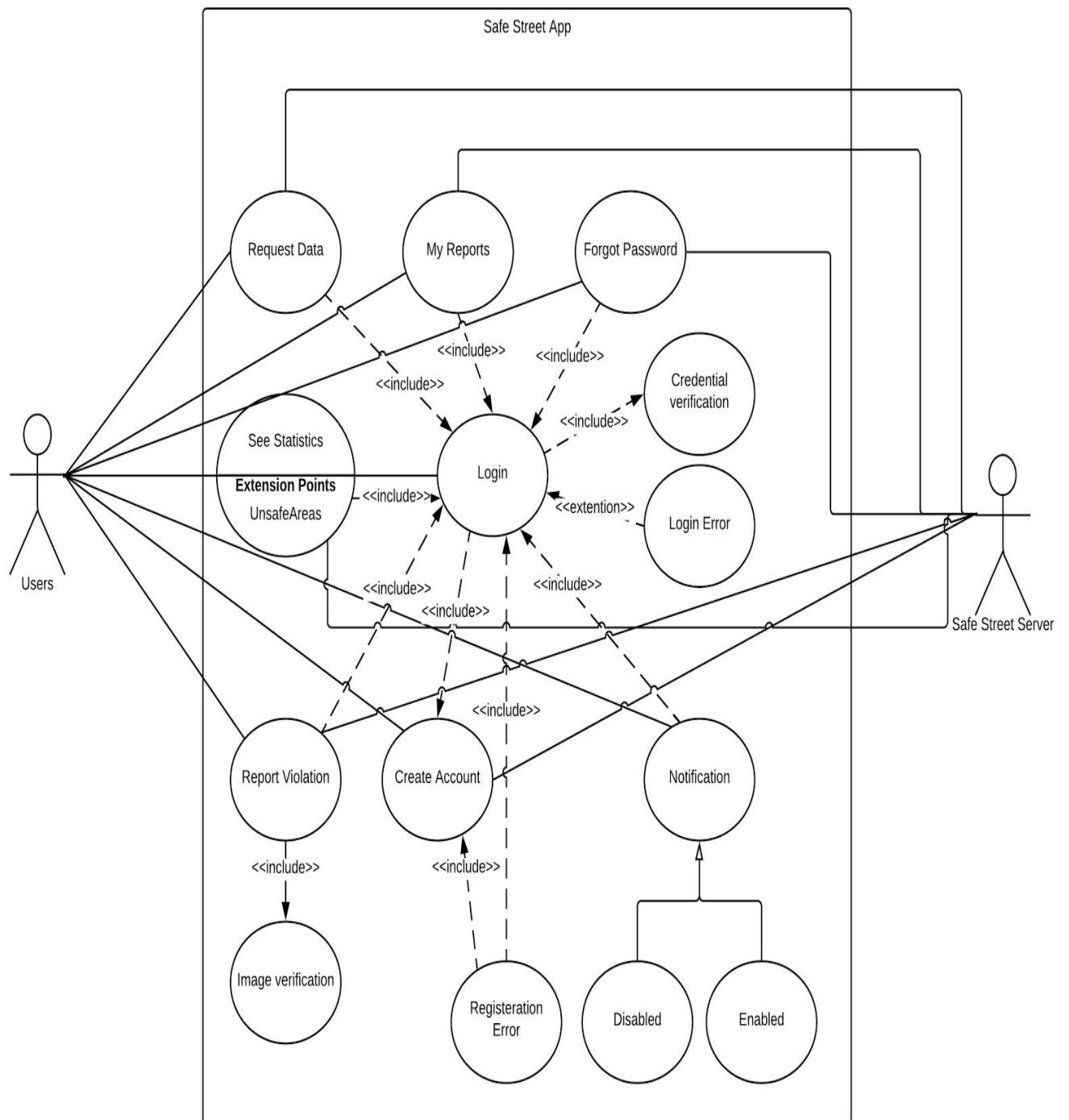


Figure 3.10 - Use Case Diagram: Regular User

Use Cases

ID:[UC1]

Name: Create Account

Actor: Users

Entry Condition: -

Event Flow:

- 1- The users press on "Create an Account"
- 2- The users fills all the mandatory fields and press "confirm"
- 3- The App sends the user data to be saved on the Server
- 4- The users receive a verification code on their phone and enter that code in the app

Exit condition: The System creates an account for the user

Exception: Show errors if the user already has an account or didn't satisfy the requirements

ID:[UC2]

Name: Login

Actor: User

Entry Condition: User already registered

Event Flow:

- 1- The user enters his email and password , then press on the "login" button

Exit condition: The system allows the user to access the main page and do various functions.

Exception: An error message will appear if the Credentials are wrong or missing .

ID:[UC3]

Name: Notification

Actor: User

Entry Condition: Logged in user

Event Flow:

- 1- The user is in the main page and press on the icon in the top left corner of the app
- 2- The user press on "Enable Notification"
- 3- The SafeStreet server keeps the user updated through sending notifications

Exit Condition: The system allows the user to get up-to-date information.

Exception: -

ID:[UC4]

Name: Report Violation

Actor: User

Entry Condition: The user is logged in and has an internet connection

Event flow:

- 1- The user presses on "Report a Violation" in the main page
- 2- The user fills the mandatory fields (Date , Time , Position) and select the type of the violation
- 3- Then the user take an image of the violation or choose an image from his phone.
- 4- The user then press on "Send" in the top right of the Report Violation page

Exit Condition: The App checks that the image is not altered and send the violation details along with the image to the Safestreet server .

Exception: If the selected image is altered the app doesn't send the image to the server, without notifying the user for security purposes.

ID:[UC5]

Name: My Reports

Actor: User

Entry Condition : The user must be logged in

Event Flow:

- 1- The user press on "My Reports" button, then the app retrieve the info locally from the user app

Exit condition: The app show the user a list of violations he has reported

Exception: -

ID:[UC6]

Name: Request Data

Actor: User

Entry Condition: The user must be logged in and have an internet connection

Event Flow:

- 1- The user presses on "Request Data"
- 2- A request is sent to the server

Exit Condition: The server sends an email which contain a link to download a censored version of the data

Exception: -

ID:[UC7]

Name: Forgot password

Actor: User

Entry Condition: Already Registered User

Event Flow :

- 1- The user press on "Forgot the Password"

Exit Condition : The system sends the user an email with a link to create a new password

Exception: The app show an error if the user is not registered

ID:[UC8]

Name: See Statistics

Actor: User

Entry Condition: The User should be logged in

Event Flow:

- 1- Press on "See Statistics"
- 2- Specify the street and the time interval
- 3- Press on "Confirm"
- 4- A request to the SafeStreet server to bring the mined data given the time and the location specified

Exit Condition:

The Server sends back the requested data to the app ,then the app shows the data in a user friendly chart

Exception: -

Sequence Diagram

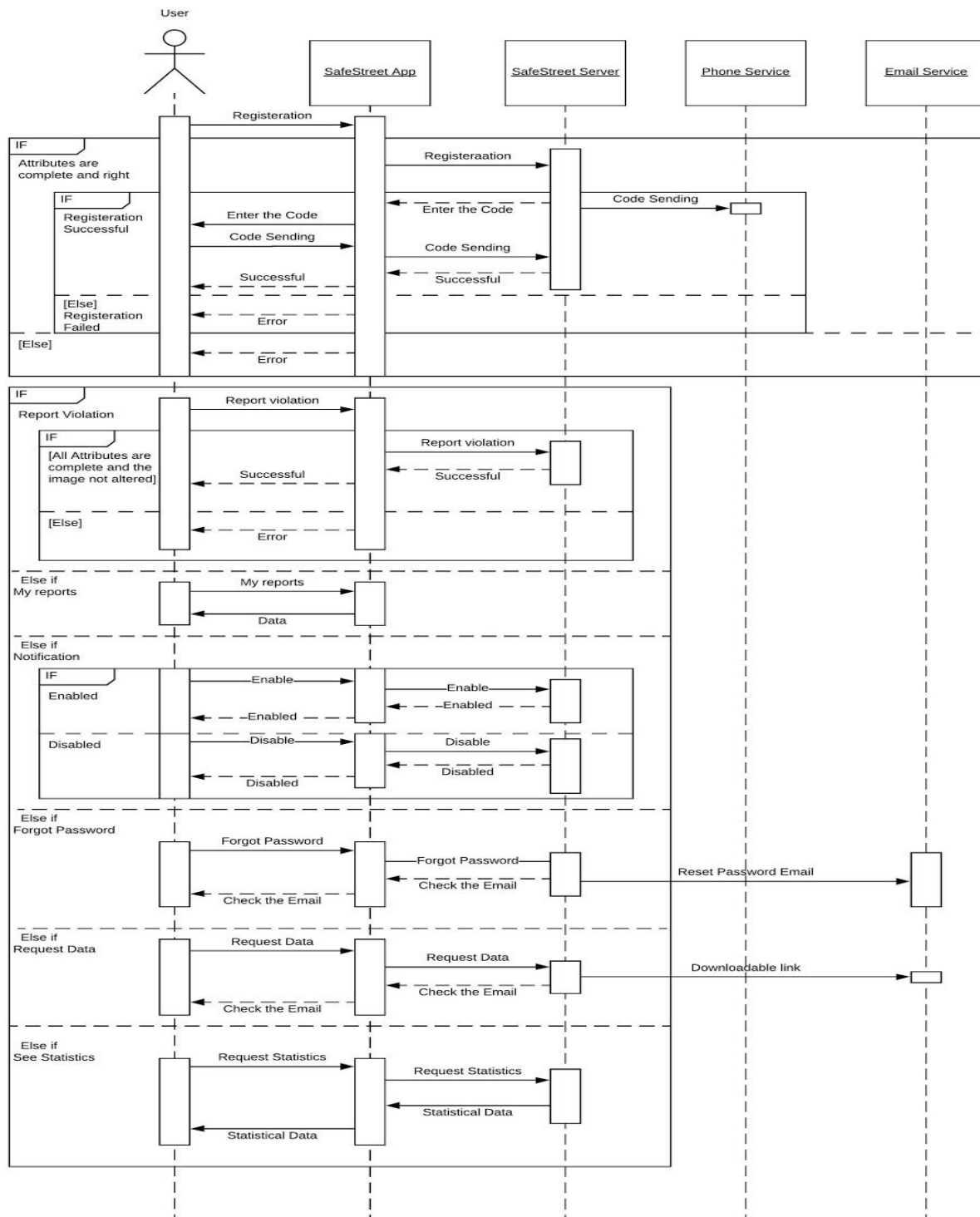


Figure 13 - Sequence Diagram: Regular User

3.2.2 Municipalities

Scenario 1

The Municipality of Milan has received an email from Safestreet introducing its initiative. The Municipality is very interested and decides to opt in. They click on the link to the website on the email and fill in their email in the activation page. Shortly after, they receive a confirmation email with the password to login to their account.

Scenario 2

The Municipality of Rome have just received a new budget for road works, they have opted in to the Safestreet initiative and would like to see where there may be some improvements needed. They login to the Safestreet website and click on suggestion. There they see that Safestreet have suggested a barrier be created near the bike lane on via Enrico Cosenz, 22, due to the high number of reports of cars being parked there. The Municipality has now found a good use to allocate some of its budget to.

Scenario 3

The Municipality of Torino have decided that they would like to look at some data about violations that are occurring in their city, in order to possibly ticket violations that are occurring way too frequently. They have opted in to the Safestreet initiative, they login to the Safestreet website, they click on access data and select the time period of one year and choose to receive the data about violations that have occurred in the past year. They then click the download pdf button and receive a pdf with the data. They see that parking in bike lanes has become a huge problem in their city and decide to start giving out hefty tickets to the perpetrators.

Use Case Diagram

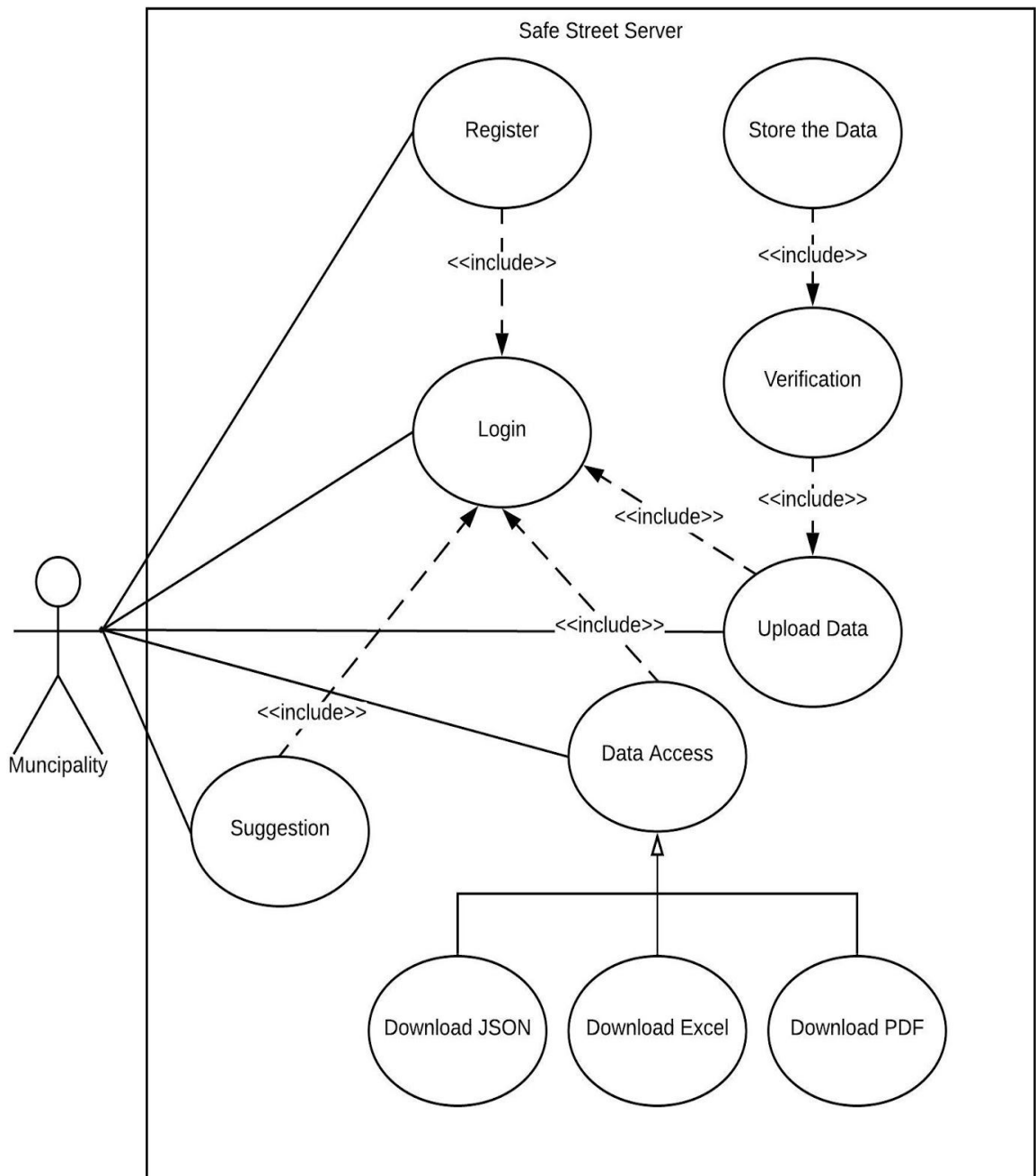


Figure 11 - Use Case Diagram: Municipalities

Use Cases

ID:[UC9]

Name: Sign Up

Actor: Municipality

Entry Condition: The Government should have handled SafeStreet a list of Verified emails that the Municipality use's

Event Flow:

- 1-The Municipality goes onto the website and fills in their email in the activation field
- 2-Safestreet cross checks this email with the list from the government and if correct sends login details password to that email.

Exit Condition: The Municipality receives an email with the login details

Exception: The email provided is not on the list and no login details are sent

ID:[UC10]

Name: Data Access

Actor: Municipality

Entry Condition: The Municipality must be logged into SafeStreet website

Event Flow:

- 1- The Municipality press on the "Data Access" button
- 2- The Municipality should fill the required attributes for downloading a related data
- 3- The Municipality can download the data if no attribute is specified
- 4- the Municipality choose its preference of the data extension

Exit Condition: The Municipality get a copy of the data on their local machine

Exception: -

ID:[UC11]

Name: Upload Data

Actor: Municipality

Entry Condition: The Municipality must be logged into SafeStreet website

Event Flow:

- 1- The Municipality press on "Upload Data"
- 2- A popped-up window will appear to upload the data
- 3- Then the Municipality navigate through their own files and choose a file to upload
- 4- Then press confirm

Exit Condition: Safe Street server receive data from the Municipality to cross information's

Exception: If the extension of the data doesn't comply with required data type , an error message will appear

ID:[UC12]

Name: Suggestion

Actor: Municipality

Entry Condition: The Municipality must be logged onto the SafeStreet website

Event Flow:

- 1- The Municipality presses on " Suggestions "
- 2- A list of suggestions from Safestreet will appear

Exit Condition: The Municipality receives suggestions from SafeStreet in order to help decide on intervention to reduce the number of violations

Exception: -

Sequence Diagrams

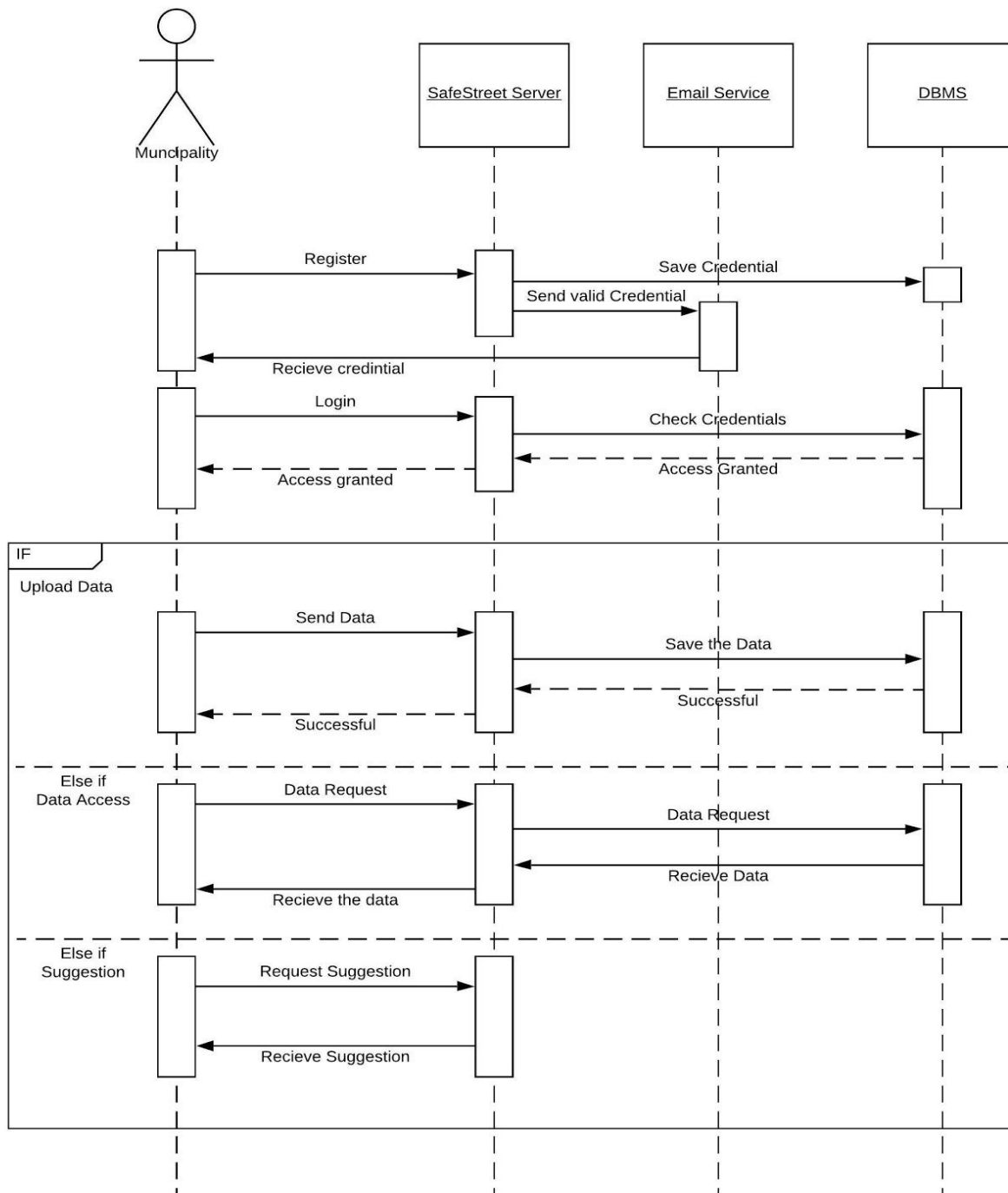


Figure 11 - Sequence Diagram: Municipalities

3.2.3 Requirements

Here we discuss the requirements needed in order to achieve each goal to the highest standard possible, taking into the domain assumptions.

G1: users can input data of violation to SafeStreet

- R1: The system must allow a user to register a new account
 - R2: The system must allow a user to login to his/her account
 - R3: The system must allow users to take images and to input details of violation
 - R4: The system must allow users to upload images and data of violation to its database
 - R5: The system must be able to apply a security mechanism able of detecting image tampering
 - R6: The system must be able to store data of violation on the user side
 - R7: The system must be able to communicate with its database
 - R20: The system must allow users to use a map service to locate violations
-
- D1: The system is authorized to store data of violation in its database
 - D2: User's mobile has enough space to store images of violation
 - D3: Users had allowed SafeStreets application to use storage, camera, internet service ,Location
 - D4: SafeStreets is using a secure communication protocol
 - D5: The details entered by users about location is valid

G2: Users can access anonymized data of violation

- R1: The system must allow a user to register a new account
 - R2: The system must allow a user to login to his/her account
 - R9: The system must be able to store data of violation in its database
 - R11: The system must allow users to send a request for data of violation
 - R12: The system must be able to anonymize data accessed by users
 - R13: The system must be able to send the data of violation to the email the user had registered with.
-
- D1: The system is authorized to store data of violation in its database

G3: Active municipalities can access data of violations

R23: The system must allow a municipality to activate its account

R24: The system must allow active municipality to access its account

R9: The system must be able to store data of violation in its database

R14: The system must allow active municipalities to filter data by type, time, date, and location of violation

R15: The system must allow municipalities to access the data of violation

R25: The system must allow active municipalities to download generated file of violations

D1: The system is authorized to store data of violation in its database

D5: The government has provided SafeStreets with information about municipalities

G4: Users can see customized statistics and list of unsafe areas

R1: The system must allow a user to register a new account

R2: The system must allow a user to login to his/her account

R9: The system must be able to store data of violation in its database

R10: The system must allow users to view list of violations

R11: The system must anonymize statistics of violation seen by users

R12: The system must allow users to filter the list of violation by location, type, time, and frequency

R16: The system must be able to communicate with active municipalities to acquire data of accidents

R17: The system must allow users to see a list unsafe areas

R18: The system must allow users to filter the list of unsafe areas by location, type, time, and frequency

R23: The system must allow a municipality to activate its account

R24: The system must allow active municipality to access its account

D1: The system is authorized to store data of violation in its database

D5: The government has provided SafeStreets with information about municipalities

D6: The system is authorized to store and analyze the data of accidents received from municipalities

G5: Users can opt to have notifications about the safety of an area

- R1: The system must allow a user to register a new account
- R2: The system must allow a user to login to his/her account
- R9: The system must be able to store data of violation in its database
- R16: The system must be able to communicate with active municipalities to acquire data of accidents
- R19: The system must allow users to send a request to subscribe to notification service
- R20: The system must allow users to get safety notifications about the area he is currently in
- R21: The system must be able to communicate with a map service to locate the areas
- R22: The system must be able to notify the user if the safety status of an area has changed
- R23: The system must allow a municipality to activate its account
- R24: The system must allow active municipality to access its account

- D1: The system is authorized to store data of violation in its database
- D3: Users had allowed SafeStreets application to use storage, camera, internet service ,Location
- D5: The government has provided SafeStreets with information about municipalities
- D6: The system is authorized to store and analyze the data of accidents received from municipalities

G6: Active municipalities can see SafeStreets' suggestions about how to improve unsafe areas

- R23: The system must allow a municipality to activate its account
- R24: The system must allow active municipality to access its account
- R16: The system must be able to communicate with active municipalities to acquire data of accidents
- R26: The system must allow active municipalities to check the suggestions offered by SafeStreets

- D1: The system is authorized to store data of violation in its database
- D5: The government has provided SafeStreets with information about municipalities

D6: The system is authorized to store and analyze the data of accidents received from municipalities

3.2.4 Traceability Matrix

RAW ID	GOAL ID	REQ ID	USE CASE ID
r1	G1,G2,G4,G5	R1	UC1
r2	G1,G2,G4,G5	R1	UC2
r3	G1,G2,G4,G5	R2	UC2
r4	G5	R1	UC3
r5	G5	R2	UC3
r6	G5	R9	UC3
r7	G5	R16	UC3
r8	G5	R19	UC3
r9	G5	R20	UC3
r10	G5	R21	UC3
r11	G5	R22	UC3
r12	G5	R23	UC3
r13	G5	R24	UC3
r14	G1	R1	UC4
r15	G1	R2	UC4
r16	G1	R3	UC4
r17	G1	R4	UC4
r18	G1	R5	UC4
r19	G1	R6	UC4
r20	G1	R7	UC4

r21	G1	R20	UC4
r22	G4	R1	UC5
r23	G4	R2	UC5
r24	G4	R9	UC5
r25	G2	R1	UC6
r26	G2	R2	UC6
r27	G2	R9	UC6
r28	G2	R11	UC6
r29	G2	R12	UC6
r30	G2	R13	UC6
r31	G1,G2,G4,G5	R2	UC7
r32	G4	R1	UC8
r33	G4	R2	UC8
r34	G4	R9	UC8
r35	G4	R10	UC8
r36	G4	R11	UC8
r37	G4	R12	UC8
r38	G4	R16	UC8
r39	G4	R17	UC8
r40	G4	R18	UC8
r41	G4	R23	UC8
r42	G4	R24	UC8
r43	G3,G6	R23	UC9
r44	G3,G6	R24	UC9
r45	G3	R23	UC10

r46	G3	R24	UC10
r47	G3	R9	UC10
r48	G3	R14	UC10
r49	G3	R15	UC10
r50	G3	R25	UC10
r51	G4,G5,G6	R23	UC11
r52	G4,G5,G6	R24	UC11
r53	G4,G5,G6	R16	UC11
r54	G6	R23	UC12
r55	G6	R24	UC12
r56	G6	R16	UC12
r57	G6	R26	UC12

3.3 Performance Requirements

The system must be able to service all users and municipalities simultaneously with the same high performance standard. Furthermore, the system must insure that the hashing algorithm or whatever security means we decide should prevent regular users from being able to upload altered images of any kind. Finally, the algorithm used for license plate detection must be accurate in order to not falsely accuse the wrong party.

3.4 Design Constraints

3.4.1 Standard Compliance

As per privacy policies, the system must ask appropriate permissions to be installed on the respective device. When the user registers the system must describe to the user how his or her information may be used and that he accepts the terms and conditions.

The Website and App complies with the W3C web standards.

3.4.2 Hardware Limitations

In order to use the system application, the user must have an internet connection in order to report violations and to see live statistics about traffic violations. The user however does not need an internet connection to simply take an image of a violation through the app.

3.5 Software System Attributes

3.5.1 Reliability

It is vital for the system to hash and process the images reliably, so this part of our systems reliability is vital and needs the lowest possible MTTR. The system must have two/three database mirrors in order for the system to avoid downtime due to a database failure.

3.5.2 Availability

In order for statistics used by municipalities to improve street infrastructure, traffic, etc, to be accurate, the system needs to be available 99.9% of the time because if users cannot report violations due to app failure, then the statistics will have gaps missing and the accuracy of them will be diminished.

3.5.3 Security

Security is an integral part of the application because the system needs to avoid false accusations from malicious users. The database must be heavily encrypted and the images uploaded must have a strong security measure ensuring that they have not been tampered with. The insure that an image has been taken by the app with the use of a hashing algorithm is a possible solution for the tampering of images.

3.5.4 Maintainability

The system must be designed in a way that allows for easy maintenance and updates of the system in both the short and long run. This means that the system and how it operates must be easily understood by current and future employees of safestreet. Further discussion of this matter will be present in future documents.

3.5.5 Compatibility

Everyone should be able to use this application in order to make it as effective as possible, the more violations reported the more reliable the statistics are. Therefore, in order to achieve more reliable statistics the system must be compatible with numerous different devices and technologies.

4. Alloy

4.1 Formalizing Essential Requirements

4.1.1 Purpose of the model

This model formalizes the basic requirements SafeStreets should be able to provide. These requirements include checking that users of the application have some unique attributes, be able to run a security check each image to detect altered images, and finally to be able to run a check in order to recognize similar reports reported from different users and referring to the same violation incident.

- Users should not have the same email, mobile number, and username
- Each parking violation should be associated to exactly one vehicle and occurring in exactly one location
- Vehicle have unique plate number
- Each image should be either Not_Altered or Altered but not both
- Only Not_Altered images are sent and thus labeled sent. Altered images labeled Not_sent
- From the server side, only Not_Altered images should be Received. Altered images are not sent in the first place

4.1.2 Alloy Model

// requirement 1: all users are unique in their Emails, Usernames, Mobile number

abstract sig User

{ name: one Name, surname : one Surname, username: one Username,
password: one Password, email: one Email, number: one Number }

abstract sig Name{}

```

abstract sig Username {}
abstract sig Surname{}
abstract sig Password {}
abstract sig Email {}
abstract sig Number {}
fact user_consistency { all disj u1 , u2 : User | (u1.username != u2.username )
and(u1.email != u2.email) and(u1.number !=u2.number) }
assert consistency1 { no disj u1,u2 : User | (u1.username = u2.username ) and(u1.email
= u2.email) and(u1.number = u2.number) }

```

/ requirement 2: SafeStreet recognize that two reports from two different users may refer to the same violation incident */*

```

sig Car{ plate_num: one Plate_num, violation: set Violation }
sig Plate_num{}

```

```

abstract sig Violation{ day:one Day, loc: set Loc }
one sig V1 extends Violation{}
one sig V2 extends Violation{}
one sig V3 extends Violation{}
one sig V4 extends Violation{}
one sig V5 extends Violation{}

```

```

sig Day{} sig Loc{}

```

*/*no two cars have the same plate and each violation should be at least associated with one car*/*

```

fact CarsAreUniqueAndViolationHasAtleastOneCar
{no disj c1,c2 :Car | c1.plate_num=c2.plate_num
all v : Violation | v in Car.violation
all d : Day | d in Violation.day all p : Plate_num| p in Car.plate_num all c : Car | no disj
v1 , v2 : c.violation |
((v1 in V1 and v2 in V1) or (v1 in V2 and v2 in V2) or (v1 in V3 and v2 in V3) or (v1 in V4
and v2 in V4)
or (v1 in V5 and v2 in V5) ) and (v1.day = v2.day )}

assert consistency2 {

```

all c : Car | no disj v1 , v2 : c.violation | (v1 in V2 and v2 in V2) and (v1.day = v2.day) }

// Requirement 4 run a security check each image to detect altered images

sig Image {test : one Test}

sig Sent { rec_img :one Rec_img}

sig Not_sent{}

sig Altered extends Test {not_sent : one Not_sent}

sig Not_altered extends Test {sent : one Sent}

abstract sig Test{}

sig Rec_img { assoc : one Image}

fact {

all alt : Altered | alt in Image.test

all not_alt : Not_altered | not_alt in Image.test

no disj not_alt1 , not_alt2 : Not_altered | not_alt1.sent = not_alt2.sent

all ns : Not_sent | ns in Image.test.not_sent

all s : Sent | s in Image.test.sent

all disj i1 , i2 : Image | i1.test != i2.test

all disj s1, s2 :Sent | s1.rec_img != s2.rec_img

all disj s1, s2:Sent | s1.rec_img != s2.rec_img

all disj rec1,rec2 : Rec_img | rec1.assoc != rec2.assoc

all i : Image | i.test.sent.rec_img.assoc = i}

assert consistency3 {all i : Image | ((i.test in Altered) and (i.test.not_sent in Not_sent)) or
((i.test in Not_altered) and (i.test.sent in Sent))

all i : Image | i.test.sent.rec_img.assoc = i }

check consistency3

check consistency1

check consistency2

check consistency3

```

pred show {}
run show {}

```

4.1.3 Generated Worlds

Figure 4.1 below presents the one sample of the generated world following the alloy script written in section 4.1.2. As it can be seen that the generated world respects all the requirements specified in section 4.1.1.

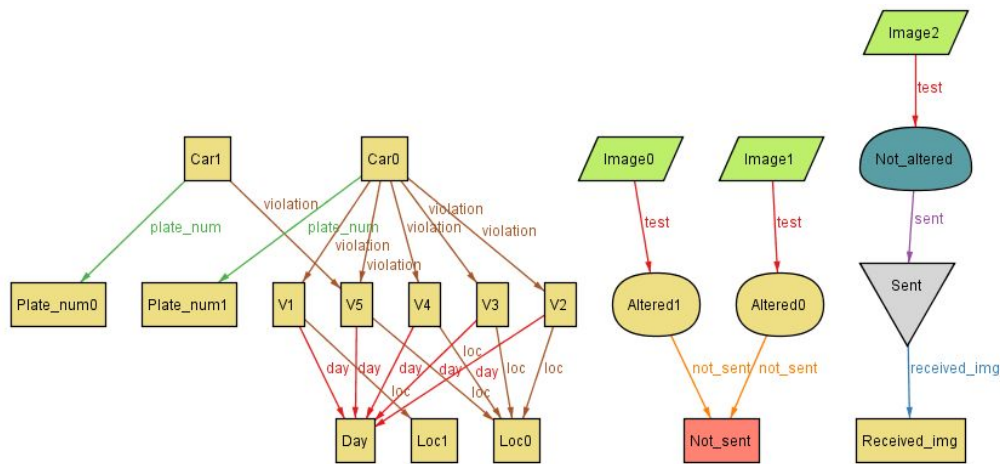


Figure 4.1 Generated World of Model 4.1

4.2 Formalization of Goal five

4.2.1 Purpose of the model

This model formalizes the crucial part of Goal 5. To do that we defined the following signature: User that is opting to receive notification about if the are is unsafe, Street in which can have User existing on it, and has Status which takes two values either Safe or Unsafe. Notification signature is declared which depends on the input of Status, and finally Accidentlevel Signature to define threshold above which area is considered Unsafe. The main requirements taken into account in this model are the following:

- Users of the same street get the same notification
- No user has two different notifications at a time

- No street has two accidents threshold
- Users of different streets should not get the same notification
- Notification is published if and only if the accident level of the street corresponding to that notification is above the threshold and that street has users on it

4.2.2 Alloy Model

open util / integer

sig User { street: one Street }

/ declaring signature for indicating the status of a street. A street should have exactly one status, either safe or unsafe */*

sig Street { status: one Status, accident1: one Accident }

// users should have at most one notification at a time

sig Notification { notification: some User }

// two possible status for a street

abstract sig Status { }

one sig Safe extends Status { }

one sig Unsafe extends Status { }

// declaring signature for indicating the accident severity and frequency on a street

sig Accident { accident: Int } { accident < 3 and accident > -1 }

//fact that no users have two different notifications

fact UserNotification { all u: User | no disj n, n': Notification | (n + n') in u.~notification }

//no streets have two Accident level and accident are associated to streets

fact UserNotification { all s: Street | no disj a, a': Accident | (a + a') in s.accident1 }

fact AccidentConnectedToStreet { no A: Accident | accident1.A = none }

/ if the AccidentParameter (p) of street > 3 then street is unsafe and users get notified
elses the street is safe and the user does not get notification */*

fact ComparingWithTreshold


```
{all s: Street | all p: s.accident1.accident | (p > 3 => (s.status = Unsafe and
notification.User != none)) and (p <= 3 => (s.status = Safe and notification.User =
none)))}
```

```
/*no two users of different street share the same notification and no notification exist if p
<= 3 */
```

```
fact NotificationIffAboveTreshold
{ no s: Street, n: Notification | all p: s.accident1.accident |
(p <= 3 => not (notification.User in n)) and (p > 3 => (notification.User in n)) and
all disj n1, n2: Notification | n1.notification != n2.notification}
```

```
/*This kind of a duplicate of the previous assertion but it helped us by closing all the
possible inconsistencies */
```

```
fact Notification {
no disj u1, u2: User | (u1.street != u2.street) <=> notification.u1 = notification.u2
all s: Street | s.status = Unsafe <=> not (notification.(street.s) = none)}
```

```
//assertions to check counter examples of facts
```

```
assert NotificationIffAboveTreshold{
all s: Street, n: Notification | all p: s.accident1.accident |
(p <= 3 => (notification.User in n)) and (p > 3 => not(notification.User in n))
no disj n1, n2: Notification | n1.notification != n2.notification}
```

```
check NotificationIffAboveTreshold
```

```
assert ComparingWithTreshold {
no s: Street | all p: s.accident1.accident |
(p > 3 => (s.status = Safe and notification.User = none))
and(p <= 3 => (s.status = Unsafe and notification.User != none))}
```

```
check ComparingWithTreshold
pred show {}
run show for 6
```

4.2.3 Generated Worlds

Figure 4.2 below presents the one sample of the generated world following the alloy script written in section 4.2.2. As it can be seen that the generated world respects all the requirements specified in section 4.2.1 which constitute the gist of Goal 5 regarding allowing the have the possibility to enable notifications. Here it was assumed that all users have turned on notification to simplify the model and to focus on the other crucial requirements of this goal. The model respects the two assertions in the code where no counter examples found.

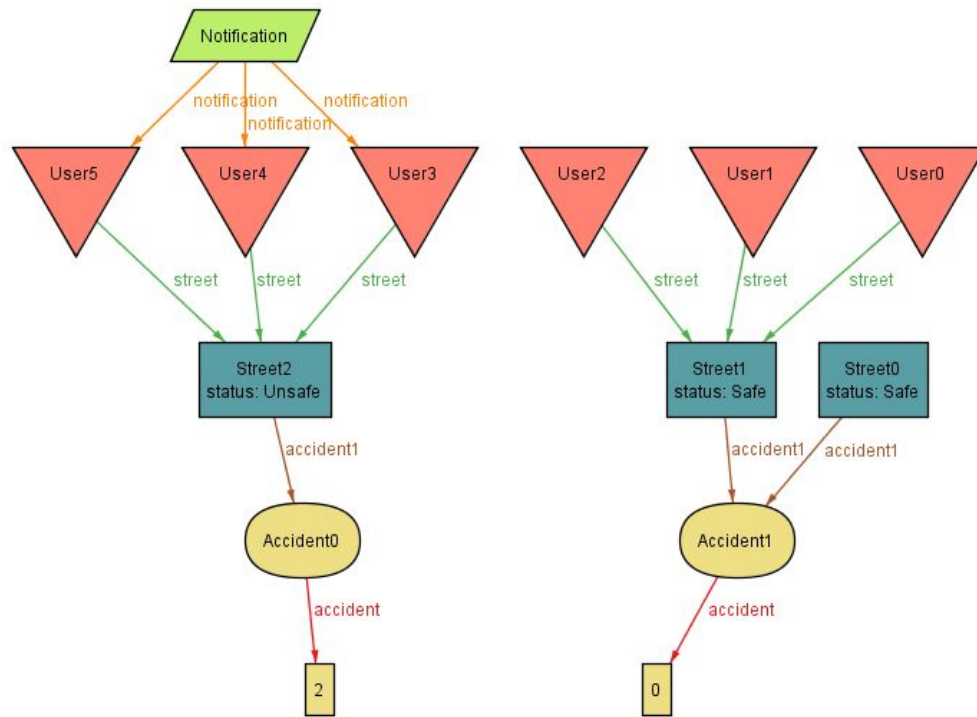


Figure 4.2: Generated world of model 4.3

5. Effort Spent

Taras Dlamini		Abdullah Quran		Hesham Abdelaziz	
Task	Hours	Task	Hours	Task	Hours
Purpose	2	Goals	3	User UI	10
Scope	2	Func. Req.	10	Use case diagrams	2
Definitions	2	Domain Assumption	2	Use case description	10
Product Perspective	1	World And Machine	2	Sequence diagrams	2
Product Function	2	State Diagrams	1	Alloy modelling	10
Municipality UI	5	Class Diagram	1		
EI Req	1	Alloy modelling	10		
Scenarios	1	Alloy layout	2		
Func. Req.	5				
Trac. Matrix	3				
Perf. Req.+Design Cons.	1				
SW system Attributes	1				
Layout & Formatting	8				
TOTAL	34	Total	34	Total	34