Faculdade de Engenharia da Universidade do Porto

FEUP

Mobile Environmental Noise Protection System

Group 3A

VERSION 1.0

Requirements and Specification Report

Ana Orvalho – ee08175
João Pinho – ee07241
Rui Costa – ee05185
Rui Pinto – ee07186
Vítor Araújo – ee07206

November 11th 2012

Version Control

Version	Date	Author(s)	Approved by the Documentation Manager	Modified Sections	Changes
0.1	06/11/12	Team	X	All of them	Creation of document
1.0	09/11/12	Team	X	Chapter 2, 3 and 4	Add a new user (Admin) Update market requirements and use cases Create the engineering requirements Association of use cases and market requirements

Index

List of Figures

List of Tables

List of Acronyms

CPU – Central Processing Unit

G*x* – Team *x*

M.E.N.P.S. - Mobile Environmental Noise Protection System

OS – Operating System

U.I. – User Interface

Chapter 1

Introduction

1.1. Presentation

Environmental noise can be defined as “all unwanted or harmful outdoor sound created by human activities”. Nowadays, this is a serious issue in all nations throughout the world. As such, there are some issues regarding lack of information and non-transparent situations that need solving.

The system to be developed has to cover the existing problems and provide a solution that allows the interaction between citizens, public authorities and city halls.

This report aims to clearly define the purpose and set of requirements of the Mobile Environmental Noise Protection System to develop, specifically focused on the crowd-sourcing and user interfaces section assigned to all groups 3. This document gives a detailed view of the user interface requirements, functionalities and restrictions of the system based on the users characteristics and their needs, defined by group 3A.

1.2. Document structure

This project requirement document is structured as follows:

- Introduction: brief explanation about the document and the project.
- System Overview: description of the project, its sub-systems, goals, features, potential users characteristics and main constraints;
- System Requirements: definition of the user-oriented system requirements, both functional and non-functional;
- Use cases: description of the functionalities offered by the system from the point of view of its different types of users.

Chapter 2

System Overview

2.1. Scope

The Mobile Environmental Noise Protection System is a distributed smartphone based system to protect citizens from illegal noise emissions.

It includes a set of inexpensive and easy to use monitoring boxes that fulfill the legal requirements imposed to noise measuring equipment. These boxes send data to a server that stores and analyses it and the results are displayed to the users through a web page and an Android compatible application.

The M.E.N.P.S. aims to provide a fast interface between city halls that publish licenses for noisy activities, citizens that consult this information and occasionally may want to report noise related incidents, and public authorities that must ensure that legal noise limits are being met. This integrated solution optimizes the response time of authorities to the requests and reports of the population.

2.2. Goals

The goals which the final system should achieve are:

- Be a cheaper solution than what exists in the market
- Give voice to the population while making sure reports are trustworthy
- Serve as a fast and safe way to inform law enforcement
- Be easy to use, implement and install
- Be well organized and logical
- Be the most automatic and autonomous possible
- Be information-wise transparent

2.3. General System Overview

The system is divided in three sub-systems, corresponding to the three groups presented in figure 1, with different responsibilities that will be explained next.

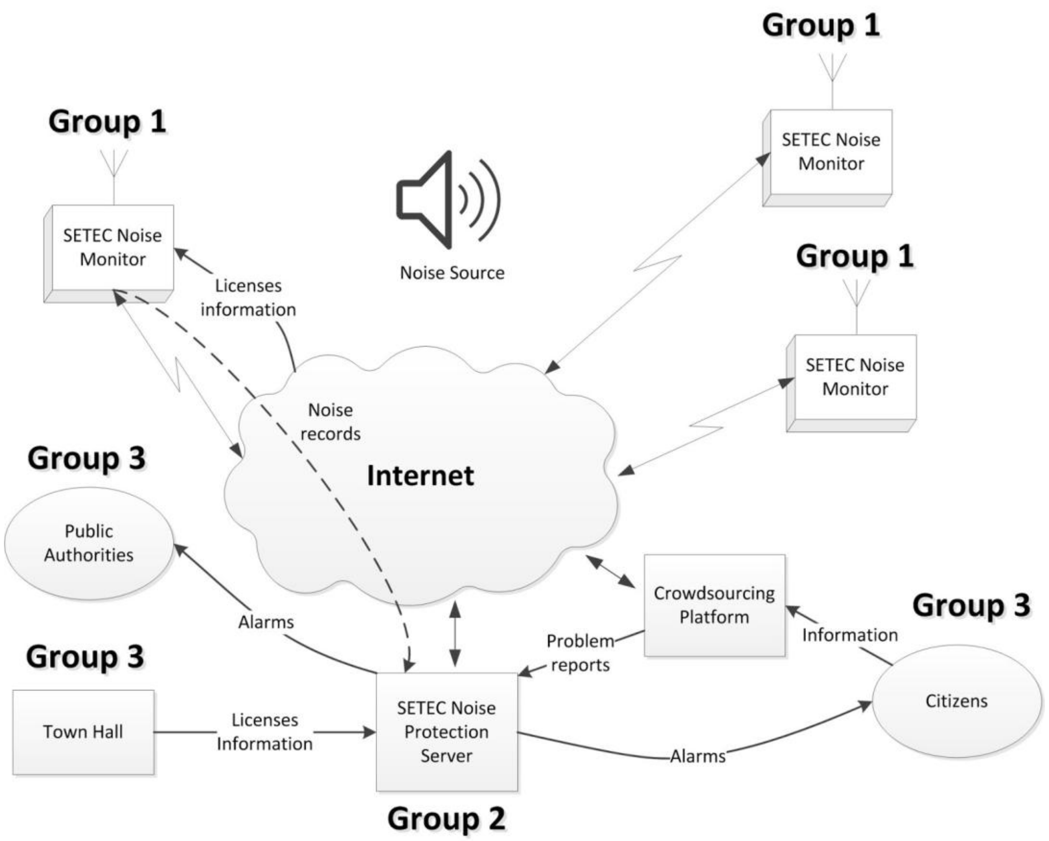


Figure 1 – Overall System Architecture

G1 - Noise Monitor

- plan, design and build the "noise monitor" hardware, including the box, microphones for sound capturing, a smartphone to transfer data to the server (G2), a battery for long periods of autonomous work and an energy source;
- develop sync-scheme between multiple monitors;
- agree on interface with G2 (noise protection server);
- develop noise measuring application (local part on the smartphone):
  - receive "licenses information" from G2;
  - automatically configure this sub-system (according to license, location, etc.);
  - perform standard measurements (includes influences from temperature, humidity, etc.);
  - cyclically forward data to the server (G2);
  - generate alarms when noise permits are exceeded;
- execute calibration with a reference device.

G2 - Noise Protection Server

- implement the different noise measurement aspects as defined by the law:
    - implement various measurement algorithms;
    - consider special noise permit rules;
    - automatically configure the system according to location of noise monitors;
    - document incidents;
    - produce life and past event noise maps and visualizations;
- agree with G1 (noise monitors) on an interface;
- agree with G3 (crowdsourcing and user interfaces) on interfaces.

G3 - Crowdsourcing and User Interfaces

- design and implement all user interfaces:
    - citizens (report incidents, information, etc.);
    - town-hall (report planned incidents, information from citizens, alarms);
    - public authorities e.g. Police (information from citizens, information related to licenses, noise alarms, etc.) for PC and smartphones;
- implement an integrated crowdsourcing platform:
    - geotag incident reports from citizens;
    - give feedback to citizens;
    - produce incident blogs.

2.4. Features

The system must present the following features:

1. Web interface: the major functionalities of the system can be used through a web page that works as main interface between the different users specified in the next section;
2. Android compatibility: in addition to the web page, the system will be accessible through an Android application, specially oriented to citizens;

3. External Microphones: to collect the environmental noise, the system will be able to connect to a set of boxes that contain external microphones, allowing the user to check the levels of noise.
4. Crowdsourcing: this type of model will allow citizens inform the authorities of high levels of noise in a certain place and create maps of reported incidents across the city.
5. Noise Maps: the environmental noise collected by the boxes allows the construction of noise maps of the cities and analyze critical areas of noise pollution.

2.5. User Characteristics

The M.E.N.P. System will be used by the common citizen, law enforcement and city hall, and also by a system administrator. All these types of users have different characteristics:

2.5.1. Citizen

This is the most common type of user. Citizens can have a general perception of the city's noise levels via a noise heat map. Once registered, the general population can report incidents of disturbance in the noise levels via either the system's web-page or the Android application. Citizens can also view the status of their own incident reports.

2.5.2. Authority

Once authenticated, law enforcement is able to access a detailed city noise heat map with every incident reported. Law enforcement is also notified whenever a box registers a noise level that is considered illegal according to the law.

2.5.3. City Hall

The Environmental Department or equivalent department within the City Hall, once authenticated on the M.E.N.P.S.'s website, can not only view noise heat maps and every incident reported but also submit licenses and request noise measurements.

2.5.4. Admin

Only the Admin can create high-level accounts for Authority and City Hall users. He can also edit and delete them.

2.6. Constraints

2.6.1. Interoperability

This project's making is divided by three types of teams:

1. Noise Measuring Box Hardware and Software;
2. Server side;
3. User interface.

There are several teams of each type making similar work. The final work must be interoperable between the teams.

2.6.2. Time

A system prototype must be delivered by December 10, 2012.
The final project will be delivered by January 21, 2013.

2.6.3. Android Compatibility

The system must be compatible with the Android OS for the user interface regarding the citizens.

Chapter 3

System Requirements

3.1. Marketing Requirements

This section presents a list of user-oriented requirements or, in other words, a list of users' needs satisfied by the M.E.N.P.S.

3.1.1. Functional Marketing Requirements

FM01 - The system must allow the registration of different types of users.

FM02 - The system must allow user authentication.

FM03 - The system should allow the consult of a generic noise heat map.

FM04 - The system must allow the consult of reported incidents.

FM05 - The system must allow incident reports.

FM06 - The system must provide feedback to the incident reports published whenever a measurement is done.

FM07 - The system must allow the input and consultation of licenses for planned noisy activities.

FM08 - The system must allow noise measurement requests.

FM09 - The system should give feedback whenever a requested measurement is done.

FM10 - The interface should include a help menu.

3.1.2. Non-functional Marketing Requirements

3.1.2.1. Performance Requirements

NF01 - The system should not consume excessive resources (CPU, battery, memory, etc.).

3.1.2.2. Reliability Requirements

NF02 - The system should always present accurate information and when in doubt no information is better than producing inaccurate one.

NF03 - Recovery time in case of failure should be the shortest possible.

NF04 - The system should have the minimum down-time possible.

3.1.2.3. Security Requirements

NF05 - The system should be the most safe possible.

NF06 - Data should be maintained confidential and respect the predetermined boundaries.

NF07 - Users can only make a limited number of reports in a determined period of time.

NF08 - Users can delete their report within a certain amount of time.

3.1.2.4. Usability Requirements

NF09 - The system should be easy to use.

NF10 - Context help should be provided in every operation.

NF11 - Error messages should have a clear and constructive language.

NF12 - Every operation done by the user should have system feedback.

NF13 - The user interface should be intuitive, simple, straightforward and alike throughout.

NF14 - The system should be accessible to users with different disabilities, with special regard to those with color-blindness and visual field limitation.

3.1.2.5. Maintenance Requirements

NF14 - The system should be easy to install.

NF15 - All necessary documentation should accompany the software.

NF16 - The system should be easily updatable.

3.2. Engineering Requirements

Marketing requirements	Engineering Requirements	Justification
FM01, NF03	System will sanitize the user input.	To protect the server from text input attacks and SQL injections.
FM02, FM03, NF05, NF06	The system must automatically manage user permissions according to their type and login.	To guarantee data confidentiality and access control.
FM02, FM03, NF05, NF06	The system should use encryption for the most sensitive data.	To be secure.
NF14	The system should use appropriate colours and contrast.	To promote user inclusion.
NF17	The programming should be organized in well-defined classes.	To allow the system to be easily updatable.

Table 1 - Engineering requirements

Chapter 4

Use Cases

The software developed by group 3A must satisfy all the requirements listed in the previous section, providing interfaces for the types of users already explained. As such, the use cases presented in figures 2 through 6 are identified and detailed below. Note that in the following diagrams the use cases were separated by actor to allow an easier reading.

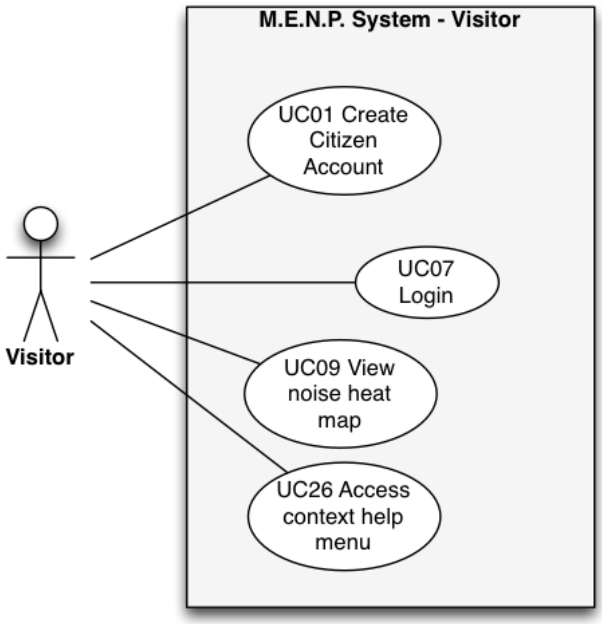

M.E.N.P. System - Visitor
Visitor
UC01 Create Citizen Account
UC07 Login
UC09 View noise heat map
UC26 Access context help menu

Figure 2 – Visitor use cases

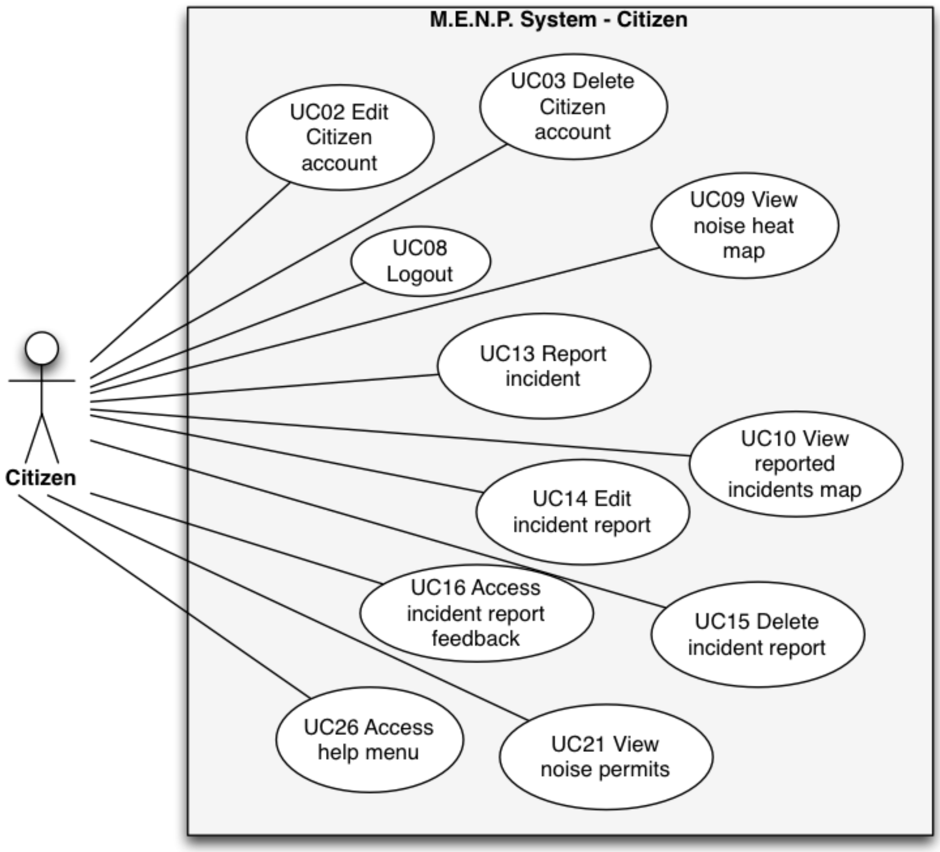

M.E.N.P. System - Citizen
Citizen
UC02 Edit Citizen account
UC03 Delete Citizen account
UC08 Logout
UC09 View noise heat map
UC13 Report incident
UC10 View reported incidents map
UC14 Edit incident report
UC16 Access incident report feedback
UC15 Delete incident report
UC26 Access help menu
UC21 View noise permits

Figure 3 – Citizen use cases

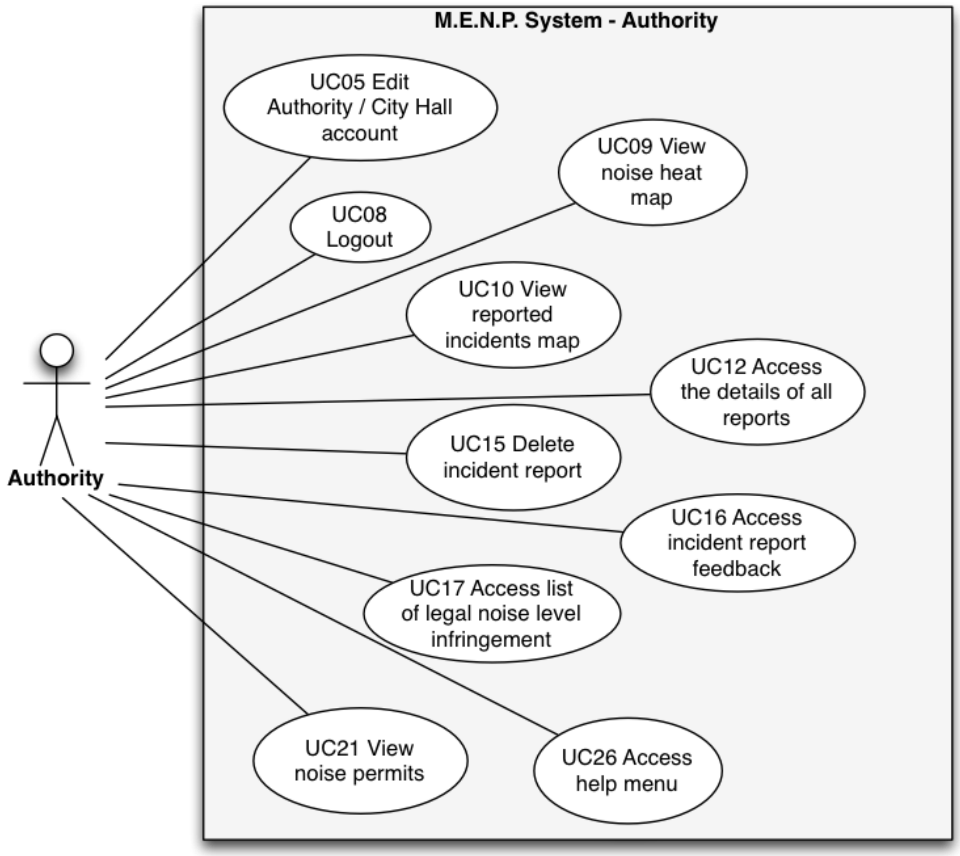

M.E.N.P. System - Authority
Authority
UC05 Edit Authority / City Hall account
UC09 View noise heat map
UC08 Logout
UC10 View reported incidents map
UC12 Access the details of all reports
UC15 Delete incident report
UC16 Access incident report feedback
UC17 Access list of legal noise level infringement
UC21 View noise permits
UC26 Access help menu

Figure 4 – Authority use cases

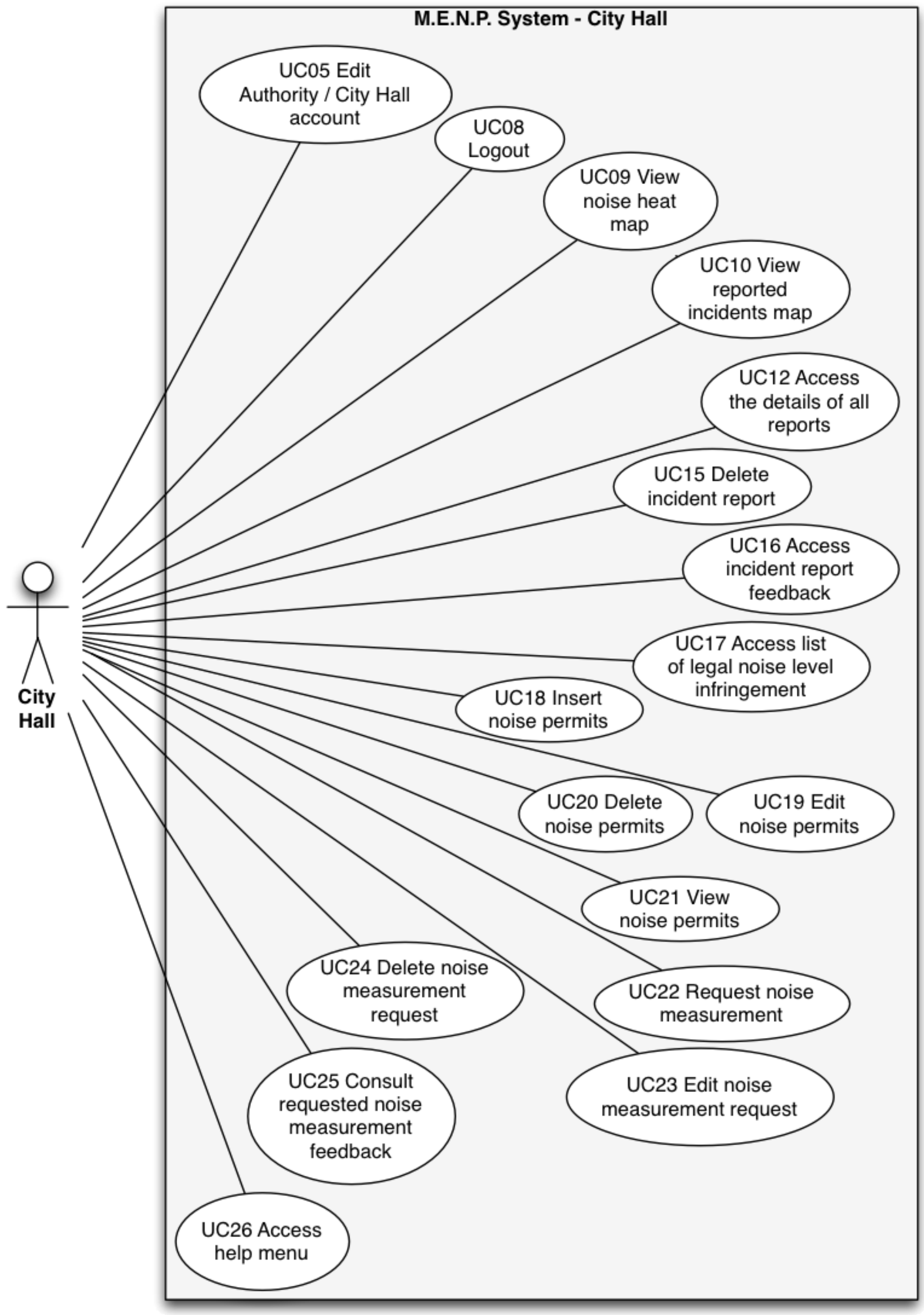

M.E.N.P. System - City Hall
City Hall
UC05 Edit Authority / City Hall account
UC08 Logout
UC09 View noise heat map
UC10 View reported incidents map
UC12 Access the details of all reports
UC15 Delete incident report
UC16 Access incident report feedback
UC17 Access list of legal noise level infringement
UC18 Insert noise permits
UC20 Delete noise permits
UC19 Edit noise permits
UC21 View noise permits
UC24 Delete noise measurement request
UC22 Request noise measurement
UC25 Consult requested noise measurement feedback
UC23 Edit noise measurement request
UC26 Access help menu

Figure 5 – City Hall use cases

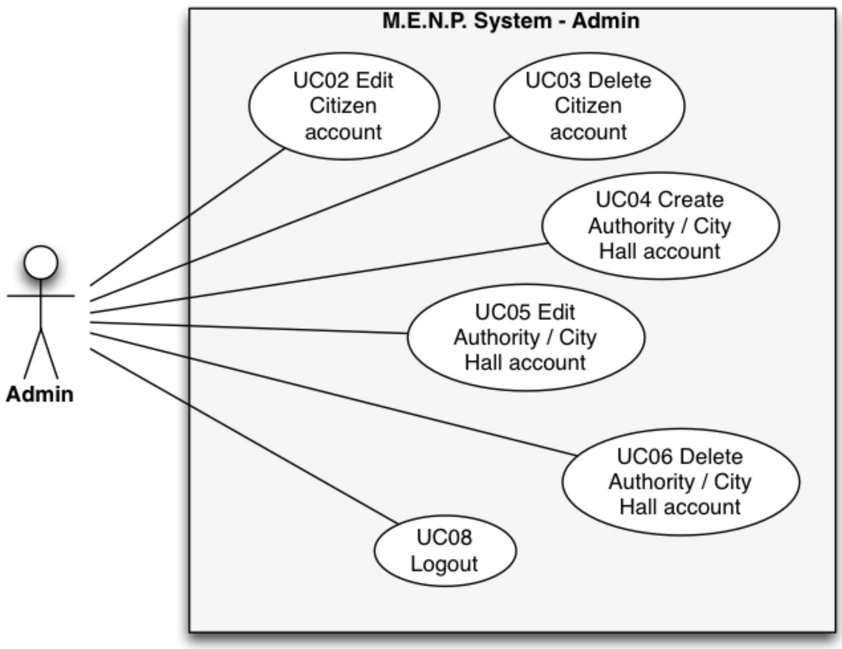

M.E.N.P. System - Admin
UC02 Edit Citizen account
UC03 Delete Citizen account
UC04 Create Authority / City Hall account
UC05 Edit Authority / City Hall account
UC06 Delete Authority / City Hall account
UC08 Logout
Admin

Figure 6 - Admin use cases

4.1. List of Use Cases by Functional Marketing Requirements

The following use cases satisfy the previously defined Functional Marketing Requirements.

FM01 - The system must allow the registration of different types of users.

Use Case	Actors	Priority (1- 5)	Description
UC01	Visitor	5	Create Citizen account
UC02	Citizen, Admin	4	Edit Citizen account
UC03	Citizen, Admin	5	Delete Citizen account
UC04	Admin	5	Create Authority/City Hall account
UC05	Admin, Authority, City Hall	4	Edit Authority/City Hall account
UC06	Admin	4	Delete Authority/City Hall account

Table 2 - FM01 use cases

FM02 - The system must allow user authentication.

Use Case	Actors	Priority (1- 5)	Description
UC07	Visitor	5	Login
UC08	Citizen, Authority, City Hall, Admin	5	Logout

Table 3 - FM02 use cases

FM03 - The system should allow the consult of a generic noise heat map.

Use Case	Actors	Priority (1- 5)	Description
UC09	Visitor, Citizen, Authority, City Hall	2	View noise heat map

Table 4 - FM03 use cases

FM04 - The system must allow the consult of reported incidents.

Use Case	Actors	Priority (1- 5)	Description
UC10	Citizen, Authority, City Hall	5	View reported incidents map
UC11	Citizen	4	Access incident report details
UC12	Authority, City Hall	5	Access the details of all reports

Table 5 - FM04 use cases

FM05 - The system must allow incident reports.

Use Case	Actors	Priority (1- 5)	Description
UC13	Citizen	5	Report incident
UC14	Citizen	2	Edit incident report
UC15	Citizen, Authority, City Hall	3	Delete incident report

Table 6 - FM05 use cases

FM06 - The system must provide feedback to the incident reports published whenever a measurement is done.

Use Case	Actors	Priority (1- 5)	Description
UC16	Citizen, Authority, City Hall	4	Access incident report feedback
UC17	Authority, City Hall	4	Access list of legal noise level infringement

Table 7 - FM06 use cases

FM07 - The system must allow the input and consultation of licenses for planned noisy activities.

Use Case	Actors	Priority (1- 5)	Description
UC18	City Hall	5	Insert noise permits
UC19	City Hall	3	Edit noise permits
UC20	City Hall	4	Delete noise permits
UC21	Citizen, Authority, City Hall	4	View noise permits

Table 8 - FM07 use cases

FM08 - The system must allow noise measurement requests.

Use Case	Actors	Priority (1- 5)	Description
UC22	City Hall	4	Request noise measurement
UC23	City Hall	2	Edit noise measurement request
UC24	City Hall	3	Delete noise measurement request

Table 9 - FM08 use cases

FM09 - The system should give feedback whenever a requested measurement is done.

Use Case	Actors	Priority (1- 5)	Description
UC25	City Hall	4	Consult requested noise measurement feedback

Table 10 - FM09 use cases

FM10 - The interface should include a help menu.

Use Case	Actors	Priority (1- 5)	Description
UC26	Visitor, Citizen, Authority, City Hall	1	Access help menu

Table 11 – FM10 use cases

4.2. List of Use Cases - Description and Basic Flow

UC01 - Create Citizen Account

Description: New Citizen Accounts can be created by an unregistered user.

Actor(s): Visitor

Priority (Min: 1, Max: 5): 5

Basic Flow:

1. Choose “Register new account”
2. Insert account details
3. Choose “Submit”

Preconditions: Not being authenticated

Post-condition: The user becomes authenticated and now has credentials

UC02 - Edit Citizen Account

Description: Citizen account details can be modified.

Actor(s): Self Citizen, Admin

Priority (Min: 1, Max: 5): 4

Basic Flow:

1. Open account details
2. Modify account details
3. Choose “Submit”

Preconditions: The user is authenticated as Citizen

Post-condition: The Citizen account is updated in the database

UC03 - Delete Citizen Account

Description: Citizen Accounts can be removed from the system.

Actor(s): Self Citizen, Admin

Priority (Min: 1, Max: 5): 5

Basic Flow:

1. (Choose Citizen account → only for Admin)

2. Choose “Edit Account”
3. Choose “Delete Account”
4. Confirm deletion

Preconditions: The user is authenticated as Citizen or Admin
Post-condition: The Citizen account is removed from the database, the user authenticated as Citizen becomes a Visitor

UC04 - Create Authority / City Hall Account

Description: Authority / City Hall Accounts can be created by the Admin.
Actor(s): Admin
Priority (Min: 1, Max: 5): 5
Basic Flow:

1. Choose “Create high-level account”
2. Insert account details
3. Choose “Submit”

Preconditions: The user is authenticated as Admin
Post-condition: The created account now exists in the database

UC05 - Edit Authority / City Hall Account

Description: Authority / City Hall Account details can be modified.
Actor(s): Admin, Authority, City Hall
Priority (Min: 1, Max: 5): 4
Basic Flow:

1. Choose account details (or Choose “Modify high-level account” and select the account → Admin)
2. Modify account details
3. Choose “Submit”

Preconditions: The user is authenticated as Authority, City Hall or Admin
Post-condition: The account details are updated in the database

UC06 - Delete Authority / City Hall Account

Description: City Hall / Authority accounts can be deleted.
Actor(s): Admin
Priority (Min: 1, Max: 5): 4
Basic Flow:

1. Choose “Modify high-level account”

2. Choose the account to modify
3. Choose “Delete Account”
4. Confirm deletion

Preconditions: The user is authenticated as Admin
Post-condition: The deleted account is removed from the database

UC07 - Login

Description: Authentication can be done by inserting username and password.
Actor(s): Visitor
Priority (Min: 1, Max: 5): 5
Basic Flow:

1. Choose “Login”
2. Insert username and password
3. Choose “Submit”

Preconditions: The user is registered
Post-condition: The user becomes authenticated

UC08 - Logout

Description: Authenticated users can de-authenticate themselves from the website or Android application.
Actor(s): Citizen, Authority, City Hall, Admin
Priority (Min: 1, Max: 5): 5
Basic Flow:

1. Choose “Logout”, available on any page

Preconditions: The user is authenticated
Post-condition: The user becomes a Visitor

UC09 - View noise heat map

Description: Users can view a generic noise heat map.
Actor(s): Visitor, Citizen, Authority, City Hall
Priority (Min: 1, Max: 5): 2
Basic Flow:

1. Choose “Noise Heat Map”

Preconditions: None
Post-condition: None

UC10 - View reported incidents map

Description: Authenticated users can view a map of reported incidents.
Actor(s): Citizen, Authority, City Hall
Priority (Min: 1, Max: 5): 5
Basic Flow:

1. Choose “Reported Incidents Map”

Preconditions: The user is authenticated as Citizen, Authority or City Hall
Post-condition: None

UC11 - Access incident report details

Description: Citizens can consult the full details of their reports.
Actor(s): SELF Citizen
Priority (Min: 1, Max: 5): 4
Basic Flow:

1. Open account details
2. Open reported incidents page
3. Choose the desired report

Preconditions: The user is authenticated as Citizen
Post-condition: None

UC12 - Access the details of all reports

Description: Users authenticated as Authority or City Hall can consult the full details of all reports.
Actor(s): Authority, City Hall
Priority (Min: 1, Max: 5): 5
Basic Flow:

1. Open reported incidents page
2. Choose the report to consult

Preconditions: The user is authenticated as Authority or City Hall
Post-condition: None

UC13 - Report incident

Description: Citizens can report high noise level incidents.
Actor(s): Citizen
Priority (Min: 1, Max: 5): 5
Basic Flow:

1. Open report incident page
2. Choose location of incident

Submit report

Preconditions: The user is authenticated as Citizen

Post-condition: The report is submitted, the user receives action confirmation

UC14 - Edit incident report

Description: Citizens can edit their own reports, within a certain time-frame.

Actor(s): SELF Citizen

Priority (Min: 1, Max: 5): 2

Basic Flow:

1. Open account details
2. Open reported incidents page
3. Choose report to edit
4. Modify desired fields
5. Choose “Submit”

Preconditions: The user is authenticated as Citizen

Post-condition: The incident report details are updated in the database

UC15 - Delete incident report

Description: Citizens can delete their own reports within a certain time-frame and Authority or City Hall can delete any report.

Actor(s): SELF Citizen, Authority, City Hall

Priority (Min: 1, Max: 5): 3

Basic Flow:

a. For Citizen:
    1. Open account details
    2. Open reported incidents page
    3. Choose report to delete
    4. Choose delete action
    5. Confirm deletion
b. For Authority or City Hall:
    1. Open list of reported incidents
    2. Choose report to delete
    3. Choose delete action
    4. Confirm deletion

Preconditions: The user is authenticated as Citizen, Authority or City Hall

Post-condition: The incident report is removed from the database

UC16 - Access incident report feedback

Description: Noise incident report feedback can be consulted after completion. Citizens can only consult their own reports.

Actor(s): Self Citizen, Authority, City Hall

Priority (Min: 1, Max: 5): 4

Basic Flow:

1. (Open account details → Citizen)
2. Open reported incidents page
3. Choose Incident to consult

Preconditions: The user is authenticated as Citizen, Authority or City Hall

Post-condition: None

UC17 - Access list of legal noise level infringement

Description: A list of all infringing reported incidents can be accessed.

Actor(s): Authority, City Hall

Priority (Min: 1, Max: 5): 4

Basic Flow:

1. Open reported incidents page
2. Choose to view only infringing incidents

Preconditions: The user is authenticated as Authority or City Hall

Post-condition: None

UC18 - Insert noise permits

Description: New noise permits can be inserted into the database.

Actor(s): City Hall

Priority (Min: 1, Max: 5): 5

Basic Flow:

1. Choose Permits page
2. Choose “Insert new Permit”
3. Insert permit details
4. Choose “Submit”

Preconditions: The user is authenticated as City Hall

Post-condition: The new permit is added to the database

UC19 - Edit noise permits

Description: Noise permits can be modified.

Actor(s): City Hall

Priority (Min: 1, Max: 5): 3

Basic Flow:

1. Choose Permits page
2. Choose permit to edit
3. Choose “Edit Permit”
4. Change permit details
5. Choose “Submit”

Preconditions: The user is authenticated as City Hall

Post-condition: The permit is updated in the database

UC20 - Delete noise permits

Description: Noise permits can be removed.

Actor(s): City Hall

Priority (Min: 1, Max: 5): 4

Basic Flow:

1. Choose Permit page
2. Choose permit to remove
3. Choose “Delete Permit”
4. Confirm deletion

Preconditions: The user is authenticated as City Hall

Post-condition: The permit is removed from the database

UC21 - View noise permits

Description: Authenticated users can consult permits previously added in the system.

Actor(s): Citizen, Authority, City Hall

Priority (Min: 1, Max: 5): 4

Basic Flow:

1. Open noise permits page
2. Select the noise permit to open

Preconditions: The user is authenticated as Citizen, Authority or City Hall

Post-condition: None

UC22 - Request noise measurement

Description: City Hall can request a noise measurement for a specific zone.

Actor(s): City Hall

Priority (Min: 1, Max: 5): 4

Basic Flow:

1. Choose “Request Measurement”
2. Insert measurement details
3. Choose “Submit”

Preconditions: The user is authenticated as City Hall

Post-condition: The user receives action confirmation

UC23 - Edit noise measurement request

Description: City Hall can edit measurement requests previously made.

Actor(s): City Hall

Priority (Min: 1, Max: 5): 2

Basic Flow:

1. Open measurement requests page
2. Choose the measurement request to edit
3. Choose “Submit”

Preconditions: The user is authenticated as City Hall

Post-condition: The measurement request is updated in the database

UC24 - Delete noise measurement request

Description: City Hall can delete measurement requests.

Actor(s): City Hall

Priority (Min: 1, Max: 5): 3

Basic Flow:

1. Open measurement requests page
2. Choose the measurement request to delete
3. Choose delete action
4. Confirm deletion

Preconditions: The user is authenticated as City Hall

Post-condition: The measurement request is removed from the database

UC25 - Consult requested noise measurement feedback

Description: City Hall can view the result of measurement requests previously made

Actor(s): City Hall

Priority (Min: 1, Max: 5): 4

Basic Flow:

1. Open reported incidents page
2. Choose the measurement request to consult

Preconditions: The user is authenticated as City Hall

Post-condition: None

UC26 - Access help menu

Description: All users can access a context help menu with basic instructions.

Actors: Visitor, Citizen, Authority, City Hall

Priority (Min: 1, Max: 5): 1

Basic Flow:

a. On the website:
   1. Click “Help”, available on every page

b. On the Android application:
   1. Open application menu
   2. Choose “Help” option

Preconditions: None

Post-condition: None

References

[1] AZEVEDO, Américo – Requirements, v1.0, October 30th 2012

Annexes

Annex 1 - Group 3 Use Cases

Use cases from Group 3, as of November 11th 2012. Please note that three use cases were assigned for each team to complete with the respective details.

C1. Basic Usability (G3A)
C2. Authentication (G3B)
C3. Registration of citizens (G3C)
C4. Registration of authority + city hall (G3A)
C5. Incident Report (G3B)
C6. Licence Submission (G3C)
C7. Noise Measurement Requests (G3A)
C8. Noise Measurement Requests Feedback (G3B)
C9. Authority's Complaint Access (G3C)

C1. Basic Usability (G3A)

Abstract:

If an unregistered user accesses the system he may consult the map of incidents only. This is the basic usability of the system.

Description:

An unregistered user accesses the system's website or mobile application and is shown a noise heat map with no specific incident information. A registration and login option is shown.

Priority (Min: 1, Max: 5): 5

Preconditions:

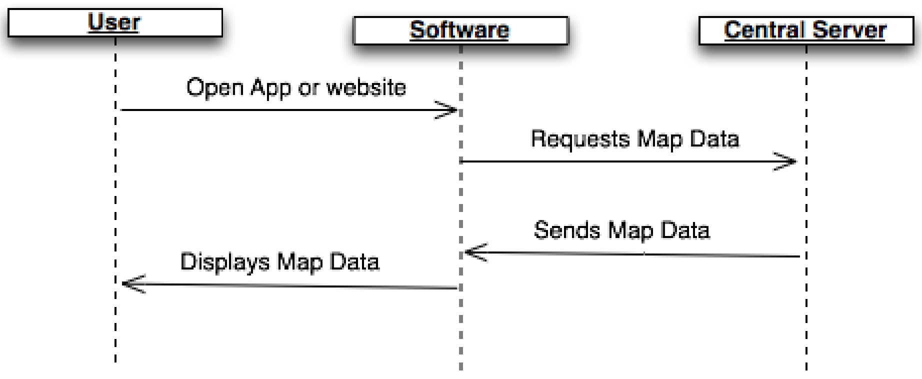
- None

Actor: Visitor, Central Server

Basic Flow

1. Visitor opens website or application on mobile phone
2. Website or application sends heat noise map request to server
3. Server replies with heat noise map data
4. Website or application shows heat noise map to user

Inter Component Flow:



Messages:

1) **Request Map Data →Website**
   a. **Protocol :** HTTP / PHP
   b. **URL:** http://nms.pt/resources/map/{coordinates}
   c. **Data Format :** JSON
   d. **HTTP Method :** GET
   e. **Parameters:**
      i. Coordinates: Top left and bottom right coordinates of the map to retrieve

f. **Response:**
   - **i.** {200 OK} if request succeeded
   - **ii.** {500 Internal Server Error} if request failed

g. **Example:**
   - **i.** HTTP-GET: http://nms.pt/resources/map/{coordinates}
   - **ii.** Response: {200 OK, webpage}

2) Request Map Data →Application

a. **Protocol :** HTTP / REST

b. **URL:** http://nms.pt/resources/map/{coordinates}

c. **Data Format :** JSON

d. **HTTP Method:** GET

e. **Parameters:**
   - **i.** Coordinates: Top left and bottom right coordinates of the map to retrieve

f. **Response:**
   - **i.** {200 OK} if request succeeded
   - **ii.** {500 Internal Server Error} if request failed

g. **Example:**
   - **i.** HTTP-GET: http://nms.pt/resources/map/{coordinates}
   - **ii.** Response: {200 OK, webpage}

3) Server sends map data to Application

a. **Protocol :** HTTP / REST

b. **URL:** {response from server, not applicable}

c. **Data Format :** JSON

d. **HTTP Method:** PUT

e. **Parameters:** Not applicable

f. **Response:**
   - **i.** {200 OK} if request succeeded
   - **ii.** {204 No Content} if no data is returned (possible network problems)

g. **Example:**
   - **i.** HTTP-POST: {return IP}:{map contents}

4) Server sends map data to Website

a. **Protocol :** HTTP / PHP

b. **URL:** {response from server, not applicable}

c. **Data Format :** JSON

d. **HTTP Method :** POST

e. **Parameters:** Not applicable

f. **Response:**

   i. {200 OK} if request succeeded

   ii. {204 No Content} if no data is returned (possible network problems)

g. **Example:**

   i. HTTP-POST: {return IP}:{map contents}

C2. Authentication (G3B)

Abstract:

The person visiting the website fills the fields required to be authenticated. After the authentication the user will have access functionalities.

Description:

The visitor opens the website or android application. Fills the information required to be authenticated. This information is sent to the server to be validated. If it is correct the user will be able to perform new functionalities besides the ones he already had. If it is wrong an error message will be displayed.

Priority (Min: 1, Max: 5): 5

Preconditions

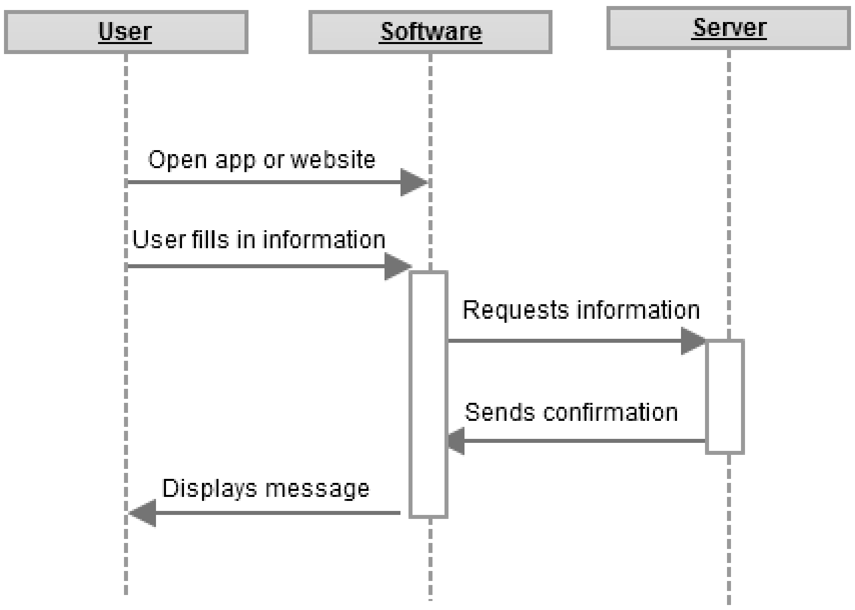
- Use Case C3 or C4: The user is registered.

Actor: Visitor, Central Server

Basic Flow

1. Visitor opens the website or android application
2. Fills in the information
3. The information is sent to the central server
4. Central server sends a confirmation message
5. The user will be authenticated or an error message will occur

Inter Component Flow:



Messages:

1.1) Request User Information (Website)

a. **Protocol :** HTTP / PHP

b. **URL:**
c. **Data Format :** JSON
d. **HTTP Method :** GET
e. **Parameters:**
    i. Username: User's username
    ii. Password: User's password
f. **Response:**
    i. {200 OK} if request succeeded
    ii. {500 Internal Server Error} if request failed
g. **Example:**

1.2) Request User Information (Application)

a. **Protocol :** HTTP / REST
b. **URL:**
c. **Data Format :** JSON
d. **HTTP Method:** GET
e. **Parameters:**
    i. Username: User's username
    ii. Password: User's password
f. **Response:**
    i. {200 OK} if request succeeded
    ii. {500 Internal Server Error} if request failed
g. **Example:**

2.1) Server Sends Information (Application)

a. **Protocol :** HTTP / REST
b. **URL:** http://nms.pt/users/login
c. **Data Format :** JSON
d. **HTTP Method:** PUT
e. **Parameters:** Not applicable
f. **Response:**
    i. {200 OK} if request succeeded
    ii. {404 Not Found} if the user information it's not in the database
g. **Example:**

2.2) Server Sends Information (Website)

a. **Protocol :** HTTP / PHP

b. **URL:** http://nms.pt/users/login

c. **Data Format :** JSON

d. **HTTP Method :** POST

e. **Parameters:** Not applicable

f. **Response:**

   i. {200 OK} if request succeeded

   ii. {404 Not Found} if the user information it's not in the database

g. **Example:**

C3. Registration of citizens (G3C)

Abstract:

Any visitor that accesses the website can register. By doing so the visitor will become a citizen being able to perform the functionalities available to this type of user.

Description:

The visitor accesses the website and clicks on “Register as new Citizen”. In order to register, the visitor will fill in a form with the required information.

After inserting the user data, the visitor sends the form. The website will exchange messages with the central server to save the citizen’s data. A new citizen user is then created.

Priority (Min: 1, Max: 5): 5

Preconditions

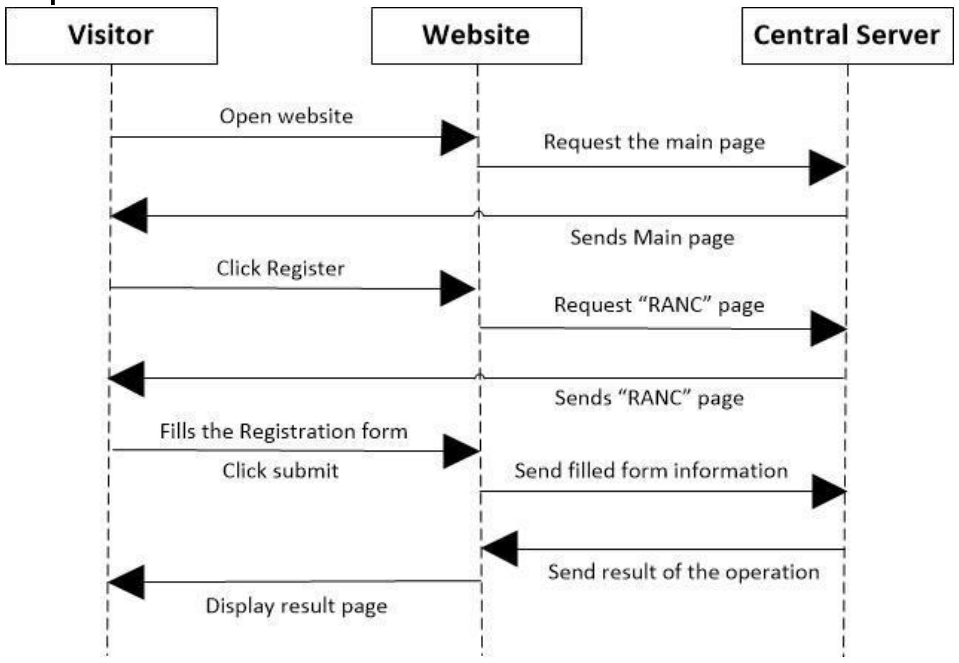
- Use Case C1: Basic Usability

Actor: Visitor, NPS

Basic Flow

1 Open NMS’ website
2 Click “Register as new Citizen”.
3 Fill registration form.
4 Submit.
5 Central Server sends a confirmation or an error message

Inter Component Flow:



Messages:

1 **Request page**
   - a **Protocol :** http / REST
   - b **URL:** http://nms.pt/
   - c **Data Format :** JSON
   - d **HTTP Method :** GET
   - e **Parameters:**
     - i None
   - f **Response:**
     - i {200 OK} if request succeeded
     - ii {500 Internal Server Error} if request failed
   - g **Example:**
     - i HTTP-GET : http://nms.pt/

2 **Register new Citizen**
   - a **Protocol** : http / REST
   - b **URL:** http://nms.pt/register.php?type=c&data={...}
   - c **Data Format :** JSON
   - d **HTTP Method :** Post
   - e **Parameters:**
     - i **type :** Type of user (Citizen)
     - ii **data :** Username, password, name, location, contact, etc.
   - f **Response:**
     - i {Success Message} If user is successfully registered
     - ii {Error Message - Bad Data} If one or more fields aren't correctly filled
     - iii {Error Message - Server Error} if the server failed to save the information
   - g **Example:**
     - i **HTTP-PUT :** http://nms.pt/register.php?type=c&{data}
     - ii **PUT parameters :** Type -- c (Citizen); data -- {username, password, first name, last name, location, ...}

C4. Registration of authority + city hall (G3A)

Abstract:

When there is a need to add users with the level “authority” or “city hall”, this should be done by the administrator only.

Description:

The Administrator accesses the website and clicks on “Register new Top Level User”. After inserting the user data, the administrator clicks Register. A new top level (authority or city hall) user is created,

Priority (Min: 1, Max: 5): 5

Preconditions

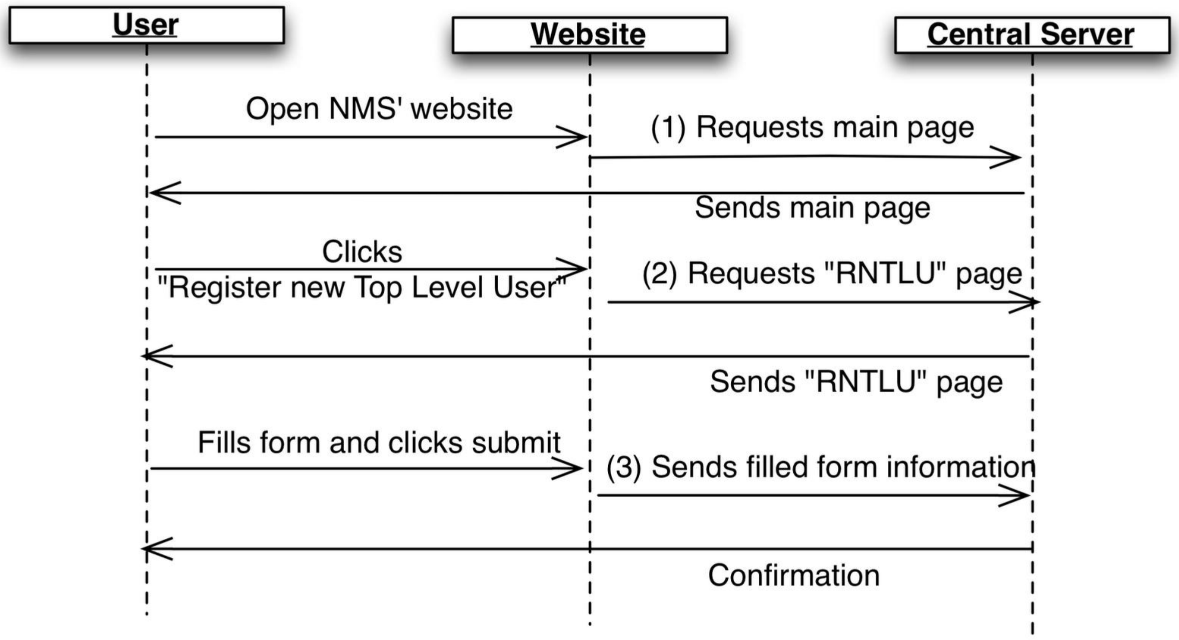
- Use Case C1: Basic Usability
- Use Case C2: Authentication

Actor: Administrator

Basic Flow

1 Open NMS’ website
2 Log in using administrator credentials
3 Click “Register new Top Level User”
4 Insert new user information, including type of user (Authority or City Hall)
5 Clicks Register

Inter Component Flow:


User
Website
Central Server
Open NMS' website
(1) Requests main page
Sends main page
Clicks "Register new Top Level User"
(2) Requests "RNTLU" page
Sends "RNTLU" page
Fills form and clicks submit
(3) Sends filled form information
Confirmation

Messages:

1) **Register new Top Level User**
    a. **Protocol :** http / PHP
    b. **URL:** http://nms.pt/register/{parameters}
    c. **Data Format :** JSON
    d. **HTTP Method :** PUT
    e. **Parameters**
        i. type : Type of user (Authority or City Hall)
        ii. other information : Username, password, name, institution, contact, etc.
    f. **Response:**
        i. {200 OK} if request succeeded
        ii. {500 Internal Server Error} if request failed
    g. **Example:**
        i. HTTP-Put : http://nms.pt/register/{parameters}
        ii. PUT Parameters : {type, username, password,...}

C5. Incident Report (G3B)

Abstract:

After being validated, the general user (Citizen, City Hall or Police) must be allowed to report incidents and/or complaints about excess noise being observed. This report is sent to the Central Server that will store it. Once submitted a report, the general user may access information, from the Central Server, about his report including available ongoing or previous measurements on the location of the complaint and existing special noise permits.

Description:

The authenticated user opens the website or android application. Fills the information about the incident. This information is sent to the server along with the location of the incident. The server stores this information and replies with a confirmation or an error message.

Priority (Min: 1, Max: 5): 5

Preconditions

- Use Case C2: The user must be authenticated

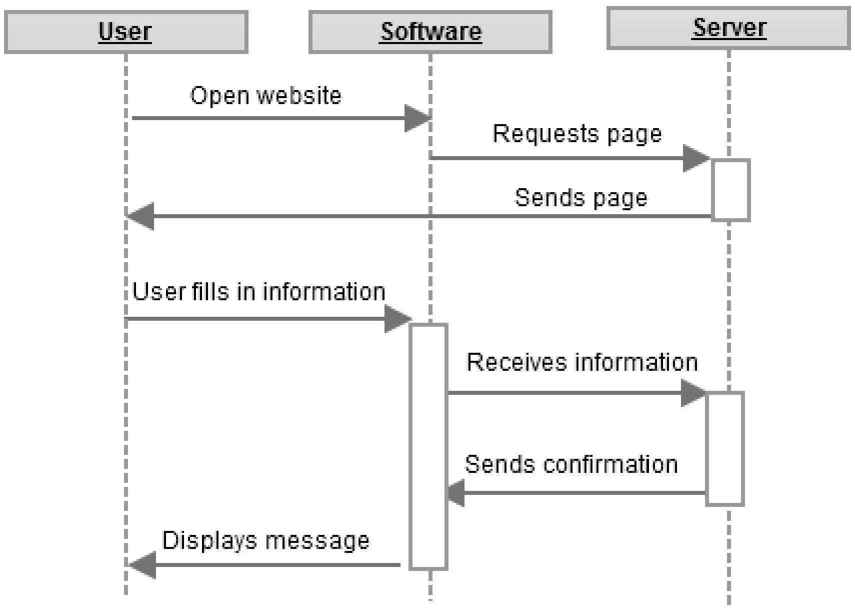
Actor:

Actor: Visitor, Central Server

Basic Flow

1. The user fills in the information about the report
2. The information is sent to the central server
3. Central server sends a confirmation message
4. A confirmation or an error message will be displayed

Inter Component Flow:



Messages:

1) **Request Website Page**
   a. **Protocol :** HTTP
   b. **URL:**
   c. **Data Format :** HTML
   d. **HTTP Method :** GET
   e. **Parameters**
   f. **Response:**
   g. **Example**

2.1) **Send Report Information (Website)**
   a. **Protocol :** HTTP / PHP
   b. **URL:**
   c. **Data Format :** JSON
   d. **HTTP Method :** POST
   e. **Parameters:**
      i. Description: The user's observations about the incident
      ii. Coordinates: GPS localization of the incident
   f. **Response:**
      i. {200 OK} if request succeeded
      ii. {500 Internal Server Error} if request failed
   g. **Example:**

2.2) **Send Report Information (Application)**
   a. **Protocol :** HTTP / REST

b. **URL:**
c. **Data Format :** JSON
d. **HTTP Method:** POST
e. **Parameters:**
   - **i.** Description: The user's observations about the incident
   - **ii.** Coordinates: GPS localization of the incident

f. **Response:**
   - **i.** {200 OK} if request succeeded
   - **ii.** {500 Internal Server Error} if request failed

g. **Example:**

2.1) Server Sends Confirmation (Application)

a. **Protocol :** HTTP / REST
b. **URL:**
c. **Data Format :** JSON
d. **HTTP Method:** PUT
e. **Parameters:** Not applicable
f. **Response:**
   - **i.** {200 OK} if request succeeded
   - **ii.** {404 Not Found} if the user information it's not in the database

g. **Example:**

2.2) Server Sends Confirmation (Website)

a. **Protocol :** HTTP / PHP
b. **URL:** http://nms.pt/users/login
c. **Data Format :** JSON
d. **HTTP Method :** PUT
e. **Parameters:** Not applicable
f. **Response:**
   - **i.** {200 OK} if request succeeded
   - **ii.** {500 Internal Server Error} if request failed

g. **Example:**

C6. Licence Submission (G3C)

Abstract:

Once validated as City Hall, the user may submit a noise license. The licence is sent to the NPS and stored to extend the system's capability to infer about the legality of the detected noise levels at a certain location.

Description:

To allow the submission of special noise permits, an authenticated user, with City Hall permissions, accesses the noise permit submission page. In this page he is required to fill out a form with the permit information. Once the form is filled he submits it and the information is verified by the system. If all the information is valid the form is sent to the noise protection server to be stored. A confirmation message is then sent to the user.

Priority (Min: 1, Max: 5): 5

Preconditions

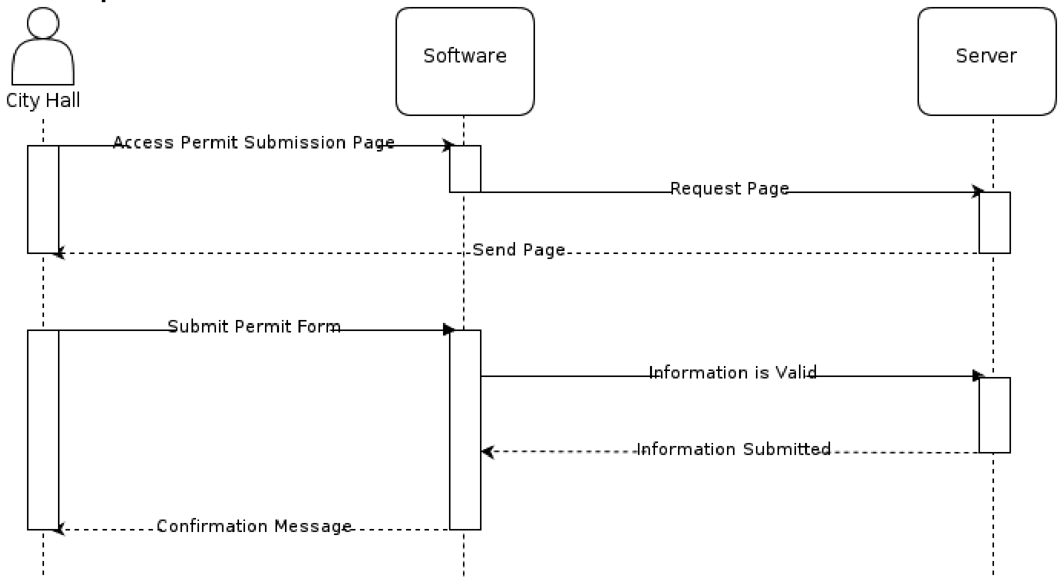
- Use Case C2: The user must be authenticated as City Hall.

Actor: City Hall, NPS

Basic Flow

1. User accesses the permit submission page
2. The user fills the form and submits it
3. The fields on the form are verified
4. If the form is validated it is sent to the server
5. The user receives a submission response

Inter Component Flow:



Messages:

1 **Request permit submission page**
   - a **Protocol :** HTTP
   - b **URL:** http://nms.pt/psf?id
   - c **Data Format :** JSON
   - d **HTTP Method :** GET
   - e **Parameters**
     - i id : The user's unique id
   - f **Response:**
     - i {200 OK} if the request is successful
     - ii {500 Internal Server Error} Failed request
   - g **Example:**
     - i HTTP-GET : http://nms.pt/psf?ch123

1 **Submit permit form**
   - a **Protocol :** http
   - b **URL :** http://nms.pt/psf?id&{form}
   - c **Data format :** JSON
   - d **HTTP Method** : PUT
   - e **Parameters:**
     - i id : The user's unique id
     - ii form : the form data // Needs required fields!
   - f **Response:**
     - i {Success Message} if all the fields are filled correctly and the form is submitted successfully to the server.
     - ii {Error Message - Bad Data} if one or more fields aren't correctly submitted
     - iii {Error Message - Server Error} if the server failed to save the information
   - g **Example:**
     - i HTTP-PUT : http://nms.pt/psf?ch123&{form}
     - ii PUT parameters : {type, issuedTo, location, ...}

C7. Noise Measurement Requests (G3A)

Abstract:

Once validated as City Hall, the user may submit requests for noise measurement to be taken. The location, time, duration and metrics to be used in the measurement must be defined. This request will be sent to the Central Server for storage.

Description:

On the website, once validated as City Hall, the user will have a menu option to submit requests for noise measurements. Once opened, a form will be available in which the user will input the location, time and duration of the desired measurement. When finished, the user will click on the "Submit" button in the UI and if the server receives the information successfully, a confirmation message will be sent to the user.

Priority (Min: 1, Max: 5): 5

Preconditions

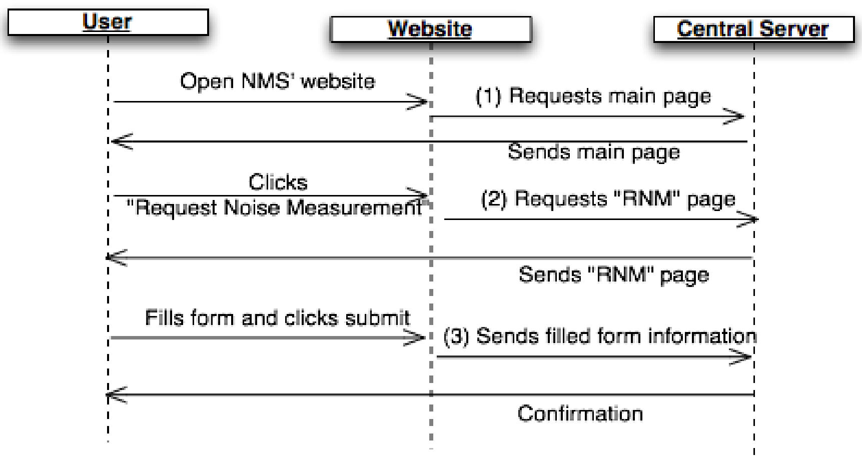
- Use Case C4: The City Hall user is registered;
- Use Case C2: The City Hall user is authenticated;

Actor: City Hall, Central Server

Basic Flow

1. **Open NMS' website**
2. **Log in using given credentials**
3. **Click "Request Noise Measurement"**
4. **Inserts location and time of said measurement**
5. **Clicks Submit**

Inter Component Flow:



Messages:

1) **Request main page**
   a. **Protocol :** HTTP
   b. **URL:** http://nms.pt/
   c. **Data Format :** HTML
   d. **HTTP Method :** GET
   e. **Parameters:** not applicable
   f. **Response:**
      i. {200 OK} if request succeeded
      ii. {500 Internal Server Error} if request failed
   g. **Example:**
      i. HTTP-GET: http://nms.pt/
      ii. Response: {200 OK, webpage}

2) **Request "Request Noise Measurement" page**
   a. **Protocol :** HTTP
   b. **URL:** http://nms.pt/requests/measurement/
   c. **Data Format :** HTML
   d. **HTTP Method :** GET
   e. **Parameters:** not applicable
   f. **Response:**
      i. {200 OK} if request succeeded
      ii. {500 Internal Server Error} if request failed
   g. **Example:**
      i. HTTP-GET: http://nms.pt/requests/measurement/
      ii. Response: {200 OK, webpage}

3) **Send filled form information**
   a. **Protocol :** HTTP / PHP
   b. **URL:** http://nms.pt/requests/measurement/{parameters}
   c. **Data Format :** JSON
   d. **HTTP Method :** POST
   e. **Parameters:**
      i. location: approximate location of requested measurement
      ii. time: day and hour of measurement request
   f. **Response:**
      i. {200 OK} if request succeeded
      ii. {204 No Content} if no data is returned (possible network problems)

g. Example:

i. PHP-POST:

http://nms.pt/requests/measurement/?l=place&t=2012110622
30

ii. Response: {200 OK}

C8. Noise Measurement Requests Feedback (G3B)

Abstract:

Once validated as City Hall, the user may give feedback on completed measurement requests. The information is stored in the server and provided when the user views details about the measurement request.

Description:

After a measurement request has been made the user can provide feedback on that request when the measurement is completed. The user accesses a request, adds information about the measurement's operations and the website sends the information to the server where it will be stored.

Priority (Min: 1, Max: 5): 5

Preconditions

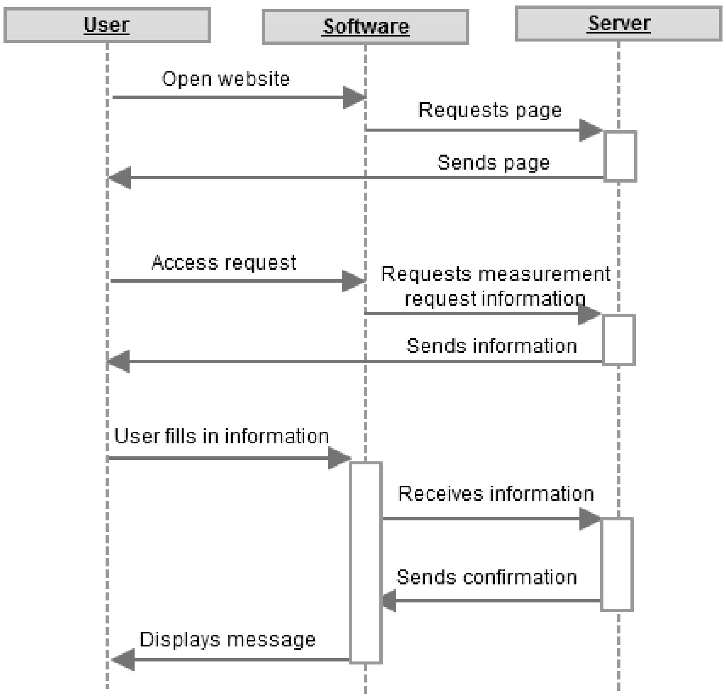
- Use Case C2: The City Hall user is authenticated;
- Use Case C7: A request has been made

Actor: City Hall, Central Server

Basic Flow

1 Access the requests page
2 Access the request information
3 Fills the request feedback
4 Server sends confirmation

Inter Component Flow:



1) **Request Website Page**

a. **Protocol :** HTTP
b. **URL:**
c. **Data Format :** HTML
d. **HTTP Method :** GET
e. **Parameters**
f. **Response:**
g. **Example**

2) **Request Measurement's Request Page**
   a. **Protocol :** HTTP
   b. **URL:**
   c. **Data Format :** HTML
   d. **HTTP Method :** GET
   e. **Parameters**
   f. **Response:**
   g. **Example**

3) **Send Feedback Information (Website)**
   a. **Protocol :** HTTP / PHP
   b. **URL:**
   c. **Data Format :** JSON
   d. **HTTP Method :** POST
   e. **Parameters:**
      i. Feedback: The request's feedback information
   f. **Response:**
      i. {200 OK} if request succeeded
      ii. {500 Internal Server Error} if request failed
   g. **Example:**

4) **Server Sends Confirmation (Website)**
   a. **Protocol :** HTTP / PHP
   b. **URL:**
   c. **Data Format :** JSON
   d. **HTTP Method :** PUT
   e. **Parameters:** Not applicable
   f. **Response:**
      i. {200 OK} if request succeeded
      ii. {500 Internal Server Error} if request failed

g. Example:

C9. Authority's Complaint Access (G3C)

Abstract:

Once validated as Public Authority, the user may access information about all the complaints in the system, and view alarms set off by the system about illegal noise emissions.

Description:

The user goes to the system website and logins as Public Authority. When the page is loading it requests the stored (and not read) alarms and complaints from the NPS. When the page is shown the user can see a message informing that there are new alarms or complaints. If he clicks in the Show Alarms button a page loads showing information about all the alarms. If the user then, selects one of them he can see the location of the measuring box that sent the alarm, the time when the alarm was triggered and the maximum intensity of the recorded noise. The message is automatically marked as read, and that information sent to the NPS.

Priority (Min: 1, Max: 5): 5

Preconditions

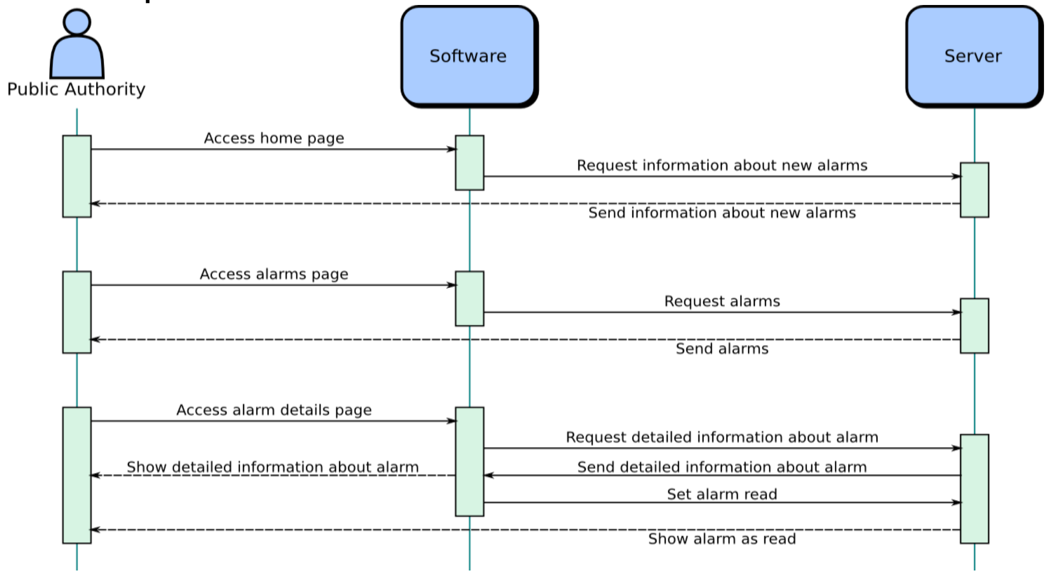
- Use Case C2: The user must be authenticated as Police.

Actor: Police, NPS

Basic Flow

1. User access alarms and complaints page
2. User access a specific alarm or complaint page
3. On the page load the User receives information

Inter Component Flow:


Public Authority
Software
Server
Access home page
Request information about new alarms
Send information about new alarms
Access alarms page
Request alarms
Send alarms
Access alarm details page
Request detailed information about alarm
Show detailed information about alarm
Send detailed information about alarm
Set alarm read
Show alarm as read

Messages:

1 **Request information about new alarms**
  - a **Protocol :** HTTP
  - b **URL:** http://nms.pt/alarm?id&{msg}
  - c **Data Format :** JSON
  - d **HTTP Method :** GET
  - e **Parameters**
    - i id : The user's unique id
    - ii msg: The code for new alarms request
  - f **Response:**
    - i {700 YES} if there are unread alarms
    - ii {800 NO} if there are no unread alarms
    - iii {500 Internal Server Error} Failed request
  - g **Example**
    - iv HTTP-GET : http://nms.pt/alarm?pa213&{331}

2 **Request alarms**
  - b **Protocol :** HTTP
  - c **URL:** http://nms.pt/alarm?id&{msg}
  - d **Data Format :** JSON
  - e **HTTP Method :** GET
  - f **Parameters**
    - i id : The user's unique id
    - ii msg: The code for alarm list request
  - g **Response:**
    - i {750 Alarm List} if there are unread alarms
    - ii {730 No alarms} if there are no unread alarms
    - iii {500 Internal Server Error} Failed request
  - h **Example:**
    - i HTTP-GET : http://nms.pt/alarm?pa213&{542}

3 **Request alarm detailed information**
  - a **Protocol :** HTTP
  - b **URL:** http://nms.pt/alarm?id&aid
  - c **Data Format :** JSON
  - d **HTTP Method :** GET
  - e **Parameters**
    - i id : The user's unique id
    - ii aid: The alarm's unique id
  - f **Response:**
    - i {710 Alarm Details} if the request is successful
    - ii {500 Internal Server Error} Failed request
  - g **Example:**
    - i HTTP-GET : http://nms.pt/alarm?pa213&a541
    - ii

4 **Set alarm read**
  - a **Protocol :** HTTP
  - b **URL :** http://nms.pt/alarm?id&aid&{msg}
  - c **Data format :** JSON

d **HTTP Method** : PUT

e **Parameters:**

  i id : The user's unique id

  ii aid : The alarm's unique id

  iii msg: The code for setting an alarm as read

f **Response:**

  i {200 OK} if the request is successful

  ii {500 Internal Server Error} Failed request

g **Example:**

  i HTTP-PUT: http://nms.pt/alarm?pa213&a541{553}