Fault Manager Lite - Technical report

Triforce
Iván Márquez Pardo, Víctor de Juan Sanz, Guillermo Julián Moreno

v1.0 - Draft

February 22, 2015

Changelog

Revision	Date	Description
0.1	02/02/2015	Brainstorm and Abstract.
0.2	02/02/2015	Added Introduction.
0.3	09/02/2015	Added research on the Internet about similar technologies.
0.4	09/02/2015	Subsystems of the application.
0.5	10/02/2015	Added Functional and Non-Functional Requirements of the application.
0.6	11/02/2015	Added Interview with the maintenance manager.
0.7	12/02/2015	Added Project Definition, Scope and Limitations.
0.8	19/02/2015	Revision of Functional and Non-Functional Requirements of the application.
0.9	20/02/2015	Mock-ups of the application finished and added.
1.0	21/02/2015	Reflection Paper and Conclusions of the project.

Abstract

In order to improve their performance, more and more maintenance services in charge of different entities or organizations have started to automate their labour using technologies and software systems specially designed taking into account the needs this sector has.

Among these needs we can find the lack or poor coordination between members or teams of the maintenance staff, the manual assignment of repair tasks and the great waste of time and personnel resources scheduling revisions of the facilities in order to find faults that may have appeared (revisions that usually are fruitless and do not detect all the faults in time, remaining unrepaired for even days).

In this paper, we present an project proposal of a web application tailored to needs of the UAM. The UAM has detected some potential maintenance problems which include the ones mentioned above.

Our main contribution relates to the development of an application, Fault Manager Lite, whose functionality is gathered in this document. We state that our application, developed after an exhaustive proccess of investigation and analysis, is able to get rid of any of the problems that the maintenance service of the UAM suffers from.

Contents

1 Introduction **3**
1.1 Structure . . . 3
1.2 Definitions and abbreviations . . . 3
1.3 Methodology . . . 4

2 Project Definition **5**
2.1 Goals and functionalities . . . 5
2.2 Subsystems or Modules . . . 6

3 Catalog of Requirements **7**
3.1 Roles . . . 7
3.2 Operating environment . . . 7
3.3 Requirements . . . 8
3.3.1 Functional requirements . . . 8
3.3.2 Non-Functional . . . 9

4 Design and Mock-ups **11**
4.1 Motivation . . . 11
4.2 Design . . . 11
4.2.1 Reporting a fault . . . 11
4.2.2 Fault tracking . . . 11
4.2.3 Interface for technicians . . . 15
4.2.4 Interface for admins . . . 15

5 Conclusions **18**

6 References **19**

A Brainstorming **20**
A.1 Ideas about new development . . . 20
A.1.1 Roles of the Application . . . 20
A.1.2 Task priority . . . 20
A.1.3 Location . . . 20
A.1.4 Manage task assignation . . . 21
A.1.5 Graphical interface . . . 21
A.1.6 Repeated faults . . . 21
A.1.7 Communication . . . 21
A.2 Questions to ask . . . 21

B Evaluate current technology **22**
B.1 Suggested solutions . . . 22
B.1.1 UAM Maintenance Fault Report System . . . 22
B.1.2 Issues Tracking System (ITS) . . . 22

B.1.3 Clean Up Control . . . 23
B.1.4 Línea verde . . . 24
B.1.5 Reparaciudad . . . 24
B.1.6 In Situ Murcia . . . 25
B.1.7 Madrid council fault report website . . . 25
B.2 Own research . . . 25
B.2.1 Wunderlist . . . 25
B.2.2 Apple Watch . . . 26
B.2.3 Trello . . . 26
B.2.4 Asana . . . 26

C Meetings **27**
C.1 Meeting Announcements . . . 27
C.1.1 Meeting Announcement - 1 . . . 27
C.1.2 Meeting Announcement - 2 . . . 27
C.1.3 Meeting Announcement - 3 . . . 28
C.1.4 Meeting Announcement - 4 . . . 28
C.1.5 Meeting Announcement - 5 . . . 28
C.1.6 Meeting Announcement - 6 . . . 29
C.2 Meeting Minutes . . . 29
C.2.1 Meeting Minutes - 1 . . . 29
C.2.2 Meeting Minutes - 2 . . . 30
C.2.3 Meeting Minutes - 3 . . . 30
C.2.4 Meeting Minutes - 4 . . . 31
C.2.5 Meeting Minutes - 5 . . . 31
C.2.6 Meeting Minutes - 6 . . . 32

D Interview **33**

Chapter 1

Introduction

The Autonomous University of Madrid (UAM) has reported the numerous problems it has on detecting faults that arise on its campus and its facilities, whose reparation usually takes excessive time and is poorly organized. A late detection of faults in the facilities delays its reparation, stopping its users to continue using them as normal and complicating the maintenance staff labour. Regarding the wishes of the campus users and realizing the maintenance problems it has, the UAM has organised a contest to choose the best project proposal that solves them. This is where our organization, Triforce, enters the scene: we have analysed the problem exhaustively and designed a web application, Fault Manager Lite (FML), that meets all the requirements expected, solves the problems and also includes new extra features that makes it even more useful.

Purpose The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the functionalities of the FML system. This document will cover each of the system's intended features, as well as offer a preliminary glimpse of the software application's User Interface (UI).

1.1 Structure

In this chapter, we introduce you to the purpose of this document, some definitions and the methodology used during the development of this proposal.

An exhaustive definition of this project can be found on chapter 2, where we state the goals of the system, its functionalities and the proposed modules.

All the requirements are detailed in the chapter 3, where functional and non-functional requirements are specified, explained and justified.

Chapter 4 defines the visual design of the application, stating the motivation for our choices and the user experience that will be delivered with this project.

Finally, we conclude with chapter 5, where we will sum up all the points exposed in this report and will present compelling reasons for why our proposal should be the chosen one.

As annexes, in appendix A we write down the transcription of our initial brainstorming, the seed of this proposal. We evaluate the current technology that could be used to solve the UAM's problems in appendix B, and in appendix C we reproduce the announcements and minutes of our meetings. We also include the interview with the UAM's maintenance administrator in appendix D

1.2 Definitions and abbreviations

- **SRS:** Software Requirements Specification.
- **FML:** Fault Manager Lite. This is the System's name.
- **UAM:** *Universidad Autónoma de Madrid*, Madrid's Autonomous University, the client for whom this project will be developed.

- **GUI:** Graphical User Interface.
- **GPS:** Global Positioning System.
- **SQL:** Structured Query Language, often used as synonymous of relational database systems.
- **Devops:** Team responsible for the deployment and maintenance of the software.

1.3 Methodology

The methodological procedure followed for preparing the SRS of the software system FML we have developed includes the following techniques:

1. Deep analysis of the information given by the UAM: the potential maintenance problem it has, the causes that led to this problem and various aspects we must consider before proceeding to the next step (web application that must work on PCs, tablets and smartphones; target users; basic purpose).
2. Application of the technique of brainstorming to generate ideas for the application. Brainstorming was then applied to these general ideas in order to give shape to them. The results of the brainstorm carried out are shown in the Annex A. This step resulted on the detection of 5 subsystems.
3. Research on the Internet about other similar systems, analyzing the functionality of these systems and describing them in a structured way by specifying their advantages and disadvantages, and extracting good ideas to incorporate to our own project, increasing its market value. These ideas also helped putting the final touches to our ideas from the brainstorm. The research is included in annex B.
4. Interview with one member of the UAM's technical staff. In this interview we clarified the obscure ideas we still had, and modified some of our previous ideas to adapt them to the answers given. The answers to the interview were well-considered; as the technical staff will be the main user of our application, their point of view is important to us. The interview is included in annex D.
5. Departing from the ideas obtained, define the requirements (functional and non-functional) of the application and design attractive mock-ups that fulfill them. These mock-ups are only a demo and may change in the final version of the application.

Chapter 2

Project Definition

FML is based on the jolly cooperation between the users of the facilities of the UAM campus and the maintenance staff in charge of them.

As the users will be the ones that will detect the sooner faults on the facilities the campus offers them, they are also the most suitable to report the problems they are having, in order to getting them fixed as soon as possible. With the FML web application, it will only take less than a minute to fill the form and send it to the maintenance staff.

Using the reports of faults detected in the campus, the maintenance personnel will stop losing its precious time revising the installations looking for faults and will be able to focus on the repairs. Apart from this benefit, the maintenance staff will also have a better way to coordinate efforts, as the FML will provide an automatic assignment system to assign repairs to each of the members avoiding overloading any of them and taking into account their distance to the problem, saving time on displacements.

In summary, the users will be able to easily report faults on facilities they are using in order to have them fixed in the less time possible, while the maintenance personnel will multiply its current performance as the majority of their resources will stop being wasted on revisions, but on repairs. We state that the FML web application is the answer to the UAM problems, as we will show you with this SRS.

2.1 Goals and functionalities

Our main goal is the development of an extremely functional, bug-free web application, that runs either on PCs, tablets or smartphones and facilitates the labor of the maintenance staff of the UAM.

Based on the jolly cooperation between the members of the UAM community (students, teaching and research staff and administration and services personnel) and the maintenance personnel of the facilities and installations of the UAM campus, this application has been designed to solved the potential maintenance problems that the UAM has declared to have.

The problems we aim to alleviate with this application include the following:

- Difficulty of detecting faults on facilities of the campus.
- Late detection of the problems, which implies repairs are not done on time.
- Excessive resources and time wasted on revisions looking for potential faults.
- Bad coordination between members of the technical staff.
- Frequent overloading of some repairmen because of bad coordination.
- Inability of a repairman to instantly inform that a fault has just been solved.
- Difficulty of the users to report faults to the maintenance services. The report system lacks of mobile functionality.
- Lack of real-time visualization of pending and finished repairs.

- Lack of real-time visualization of assigned tasks.
- No faults history.
- No statistics of faults.

2.2 Subsystems or Modules

The FML system has been designed to resolve these problems making use of a user-friendly interface, avoiding unnecessary or distracting buttons or effects. This system has been divided in several modules listed below, each of those offering one specific functionality and, in conjunction, solving the problems mentioned above:

- Task Manager.
- Report System.
- Notifications and Messaging System.
- Users and Profile Managers.
- Faults History.

There are several applications 'on the market' which are similar to our proposed app in some way or another and they could represent real and competitive alternatives to ours. However, none of these web applications can offer all the features FML offers (furthermore, we plan on adding some features which are not currently available in any of these applications), so our system is indeed the best solution proposed to solve the problems the UAM maintenance staff has to deal with. After our research of the Internet, a comparison of these competitors applications is gathered in Annex B.

Chapter 3

Catalog of Requirements

Before we start defining FML's requirements we need to define two things, the **roles** inside FML and the **Operating environment** in which FML will work.

3.1 Roles

Reporter role This is the role applied to anyone who want to report some fault. It will be taken into account the person's status inside UAM community (student, PhD, teacher, maintenance technician, etc). Every user in FML's system will be able to report faults.

Maintenance Personnel This are the roles applied to people in charge of maintenance. Inside maintenance personnel we need to distinguish 2 subcategories:

- Boss: The people in charge of each department.
- Maintenance technician: The maintenance personnel responsible to fix the faults.

As there are 8 departments, there will be 8 categories. Each maintenance person should be tagged in, at least, 1 department. It is also compatible being boss and technician.

Exceptions The **cleaning department** only needs a person in charge of the whole department and it is his responsibility to assign faults to their employees and to mark faults as solved.

This department is also special because there is a different department in each faculty, so the must be at least 1 person in charge of each faculty cleaning department. As we mentioned before, maintenance personnel can also report faults.

Administrator Role Is the person (or group of people) in charge of UAM's maintenance system in general. All administrator will have permissions to change everything, as they are in charge of everything related with maintenance.

3.2 Operating environment

The main component of Fault Manager Lite is the application, HTML based, that will run in PCs, tablets and smartphones. To improve usability and integration with operating systems, a native application wrapping the HTML interface will be developed so the users won't need to run the browser in order to access the application.

The system will require a backend exposing a RESTful API to which the HTML clients will connect. Separation of concerns is important in order to improve maintainability and ability to respond quickly to user feedback.

The server will require an installation of Python 3, and a working connection to a SQL server. Our application will be independent of the specific implementation of the server (database maintenance will be responsibility of the devops[1] team).

Specific implementation details are out of the scope of this proposal, but we will develop the application with scalability in mind: the interface, API and database applications will be independent and loosely coupled in order for the devops team to deploy each one to as many machines as needed. Cloud deployments are an option, but is not required (actually, we recommend using the university datacenter in order to comply with spanish privacy laws).

3.3 Requirements

3.3.1 Functional requirements

3.3.1.1 Task Manager subsystem

Faults definition Each fault will have 3 possible states: *Pending to assign, assigned, solved.*

If a fault is impossible to be fixed, it will correspond to the administrator to take care personally of it.

Categorizing faults There should be categories to tag each fault depending on the estimated difficulty (numeric value between 1 and 5), the urgency (urgent or not) and the department who needs to take care of it.

Fault's assignments FML will assign to maintenance personnel faults to be fixed (depending on the fault's category and maintenance person load of work and capability to solve the fault)

This will be set automatically and the system's administrators will be able to reassign the faults as they pleased.

The person in charge of a department will also be able to reassign faults that have been assigned to his department or to someone inside his department to his employees.

As we mentioned before, the cleaning departments are special, because all cleaning faults in a building will be assign to the person in charge of that building's cleaning.

Task managing Everyone from the maintenance personnel will have read access to the faults data base through a Graphical User Interface (GUI).

They will just have write access to faults they have been assigned to. [2]

Admin will have write access to all faults in the system, and the person in charge of a department will have write access to all faults assigned to someone inside his department.

When a fault is fixed The maintenance person assigned (or the manager of that department) will mark as solved the fault. At that time, the reporter will get notify and thanked.

3.3.1.2 Report subsystem

Reporting a fault

Reporting a fault with the following information: *Location, photograph (optional) and description*

See and check reported faults and see its state.

Answer questions the maintenance personnel will ask about the reported fault.

[1] I.e., the University's IT team.

[2] Write access is needed so the maintenance person change fault's state to *solved* or to *in progress*

Duplicated faults This is a big problem to be solved. What we propose to solve it has 2 aspects.

- FML will be able to mark as *possible duplicated*, so all possible duplicates will be assign to the same maintenance people, so he can verify if they are duplicated or not.
- On the other side, to prevent duplicated faults reports, FML will show a message showing possible duplicated faults to the reporter. If the reporter marks the fault he is reporting as duplicated, he will earn some points too and will be added into the list of people to be notified when the task is solved.

3.3.1.3 Communication subsystem - Notifications and messaging

Communication Maintenance person assigned to fix a fault should be able to ask the fault's reporter for more information.

3.3.1.4 Users and Profile manager system

Log in Everyone must be able to login using UAM credentials. We will use the service provide by UAM to authenticate an user.

If the user is logging in on a smartphone, the system shouldn't ask for the password more than once (except for some specific operations defined in the role-depending requirements, user role category (3.3.2.2))

Updating profile What if a student become a teacher and then his email needs to be changed? To fix this, every user must be able to update his profile's information.

3.3.1.5 Faults History

Seeing history As we think transparency is really important, every reporter will be able to see all history of faults. In non-functional requirements we define how this should be done.

Statistics In the profile page each user should be able to see the faults he has reported and it's state.

Plus, he should be able to see his conversations with maintenance personnel.

3.3.2 Non-Functional

3.3.2.1 General Requirements

This requirements are defined for all roles inside FML (Administrator, maintenance personnel and user)

Profile Everyone should have a profile with some private information such as name or email.

Lightweight application FML will be lightweight enough to run in 4 years old Chrome, Firefox, IE's versions.

Read Access to task Every FML's user will be able to see all reported faults (we think transparency is really important). This will be shown on a map of the UAM. Each fault will appear as an arrow pointing to fault's location and color-tagged depending on its state (solved, pending, assigned).

3.3.2.2 Role-depending requirements

Reporter FML should provide users the following abilities:

- The specifics requirements where re-authentication is needed are: *Answering questions asked by maintenance personnel* and *updating profile information.*

- The interface to report a fault should be easy enough to let the user report the fault without losing much time.
- The amount of points earned should be visible inside profiles page. It will be also shown the conversion in ECTS credits so the user knows how much he have earned.

Chapter 4

Design and Mock-ups

4.1 Motivation

This application's purpose is to solve a problem, not to create new. Thus, the design should not interfere with the technicians or the users work. Instead, it must reduce friction to almost zero, so it has to be easy to use and easy to learn. Adoption by the student corp of the application is critical to this project, so the design should accommodate to this requirement.

We want the application to be full-featured, so all the users of the app should be able to use all the features from whichever device they're working in: tablet, smartphone or PC. We don't want anybody to stop reporting a fault because they don't have their laptop with them, for example.

This leads to a natural decision regarding the technologies used to develop the interface: we will use web technologies and responsive design to maximize code reuse. In this way, we will be able to deliver a high quality user experience and homogeneous among devices with minimal code and improved maintainability.

In turn, this will allow us to reduce costs and will make the development more agile, as we will be able to deliver quickly changes to the application responding to feedback from users.

In this report, we will include, for brevity, the mockups for a mobile application, as we consider it will be the most used interface for the system. Given that we will be using responsive design, the looks will be the same for every device, except for widened interface elements and a navigation menu that shows up if there's enough space (see figure 4.1 compared with the corresponding mobile view on figure 4.9).

4.2 Design

We show some mock-ups of the application, specifically designed to show the flow in certain use cases.

4.2.1 Reporting a fault

From the main screen (figure 4.2), when the user presses the button "Report", he will be able to choose a category for the fault, then will fill the details and then get a page acknowledging the report and showing the estimated response time (figure 4.3). They may see a message popup (4.4) if a possible duplicated report is detected.

For the location, we will show the user a map and also a list with all the possible options (figure 4.5). The user will be able to select specific places (for example, a certain lab or classroom) or more generic (e.g., third floor or the whole building). If localization APIs are available on the device (either location by AGPS or WiFi), the application will use them to show the user the best location option.

4.2.2 Fault tracking

The user will be able to see the faults he's been subscribed too, either because he reported them or because he marked them as duplicated. The figure 4.6 shows how can he see all his faults and the chat screen that

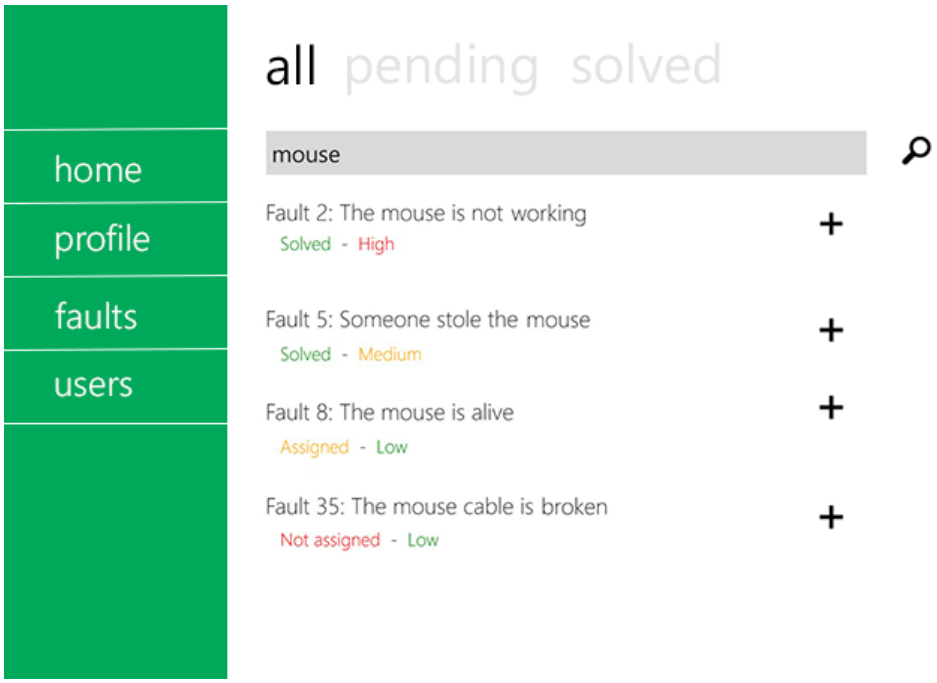

Figure 4.1: Sample responsive design in wide screens (laptops, tablets in landscape position, PCs) with the menu showing up and large interface elements .

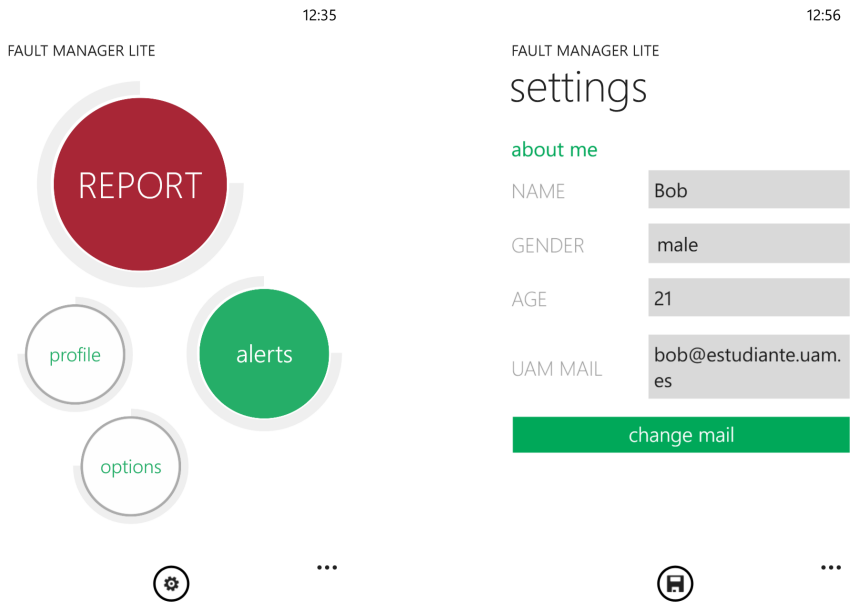

Figure 4.2: Main page and settings page for Fault Manager Lite

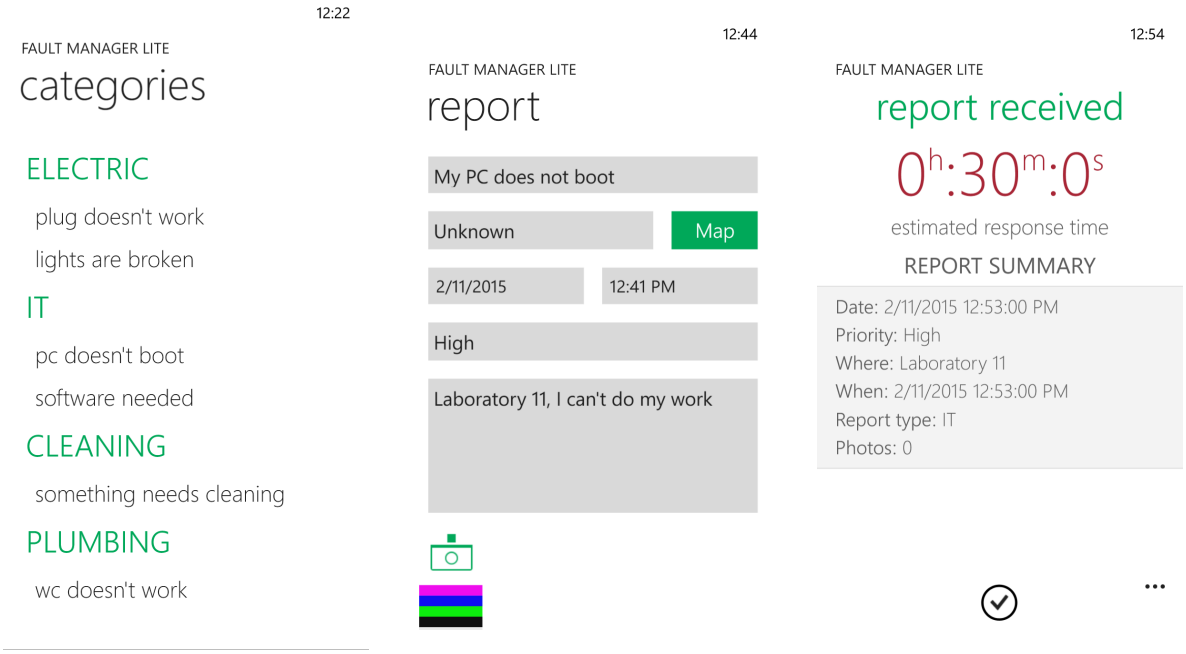

Figure 4.3: Basic flow for reporting a fault.

.

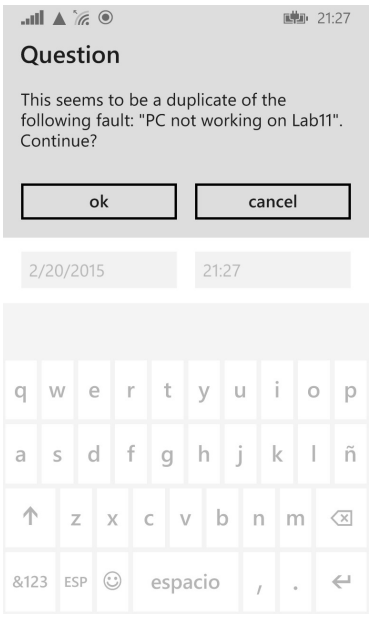

Figure 4.4: When a possible duplicate is found, the user is asked if he or she wants to continue with the report.

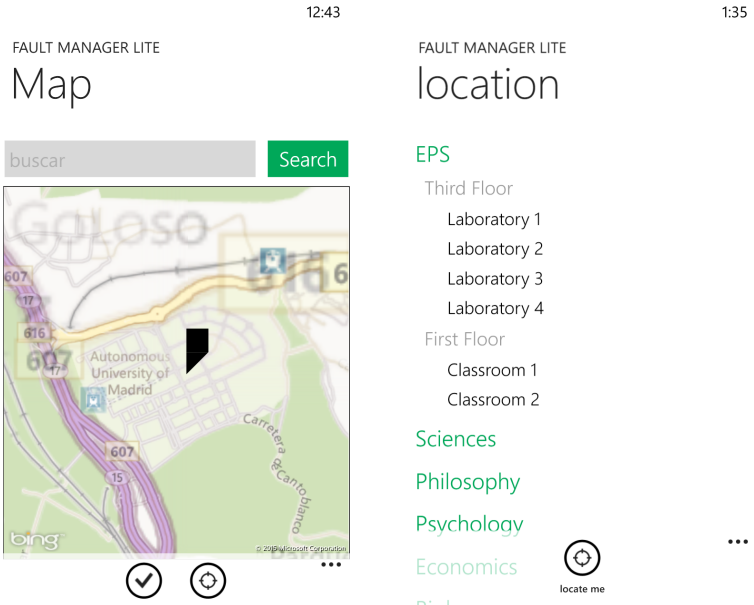

Figure 4.5: Location screens for the application

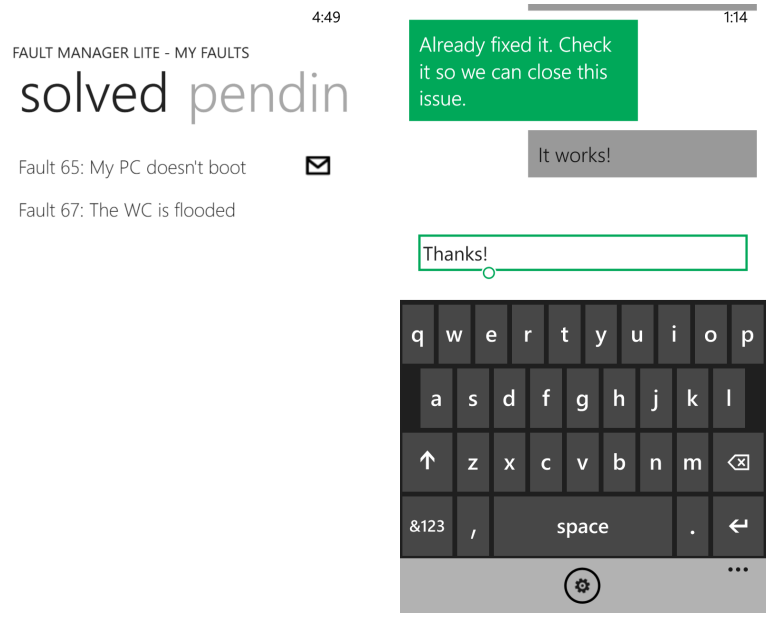

Figure 4.6: Fault log and messaging screen for the application.

will show up if the technicians need to clarify details of the report.

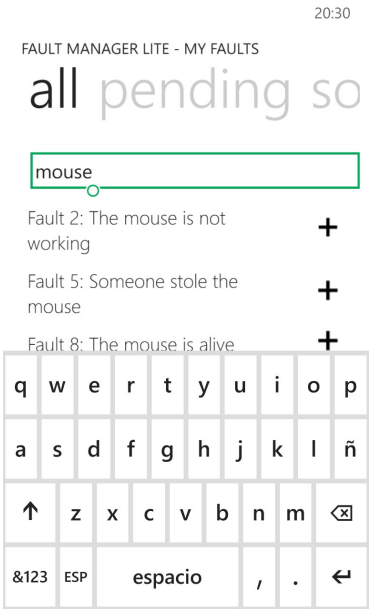

Figure 4.7: The users can see all the faults.

As specified on the requirements, the users will be able to see all the faults in the system (figure 4.7). To improve usability, we will include a search option so the users can find teh reports they're interested in.

4.2.3 Interface for technicians

To reduce costs and improve mantainability, most of the code and interface will be shared between user roles. This means that the common screens (main screen, settings, fault lists and report information) will be the same for technicians, with maybe some additional elements (see figure 4.8).

The technicians will be able to see their assigned tasks in a slightly modified interface of the one normal users see (figure 4.9), so they will be able to check the priority and state of the fault. The contextual menu of each task will allow the user to mark the fault as duplicate or assign it to someone (if the technician hasn't got managing responsibilities, he will only be able to assign the to him/herself).

4.2.4 Interface for admins

As stated in the previous section, most of the code will be shared. This means that admins will mostly see the same screens as technicians with some minor tweaks. For example, when they list all the faults (figure 4.9), they will be able to assign them to any technician under their management.

The screens that are specific to admins are shown in figure 4.10, they're related to the user management subsystem.

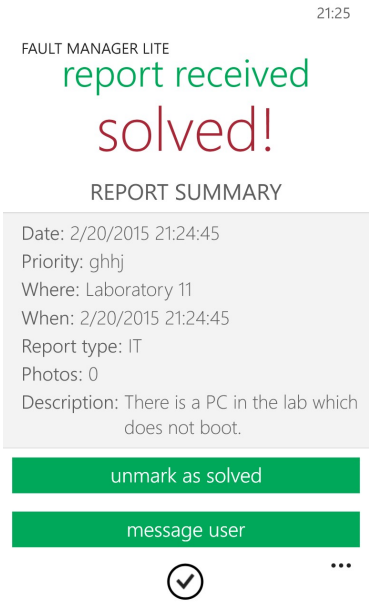

Figure 4.8: The technicians will have controls to mark a fault as solved or communicate with the users.

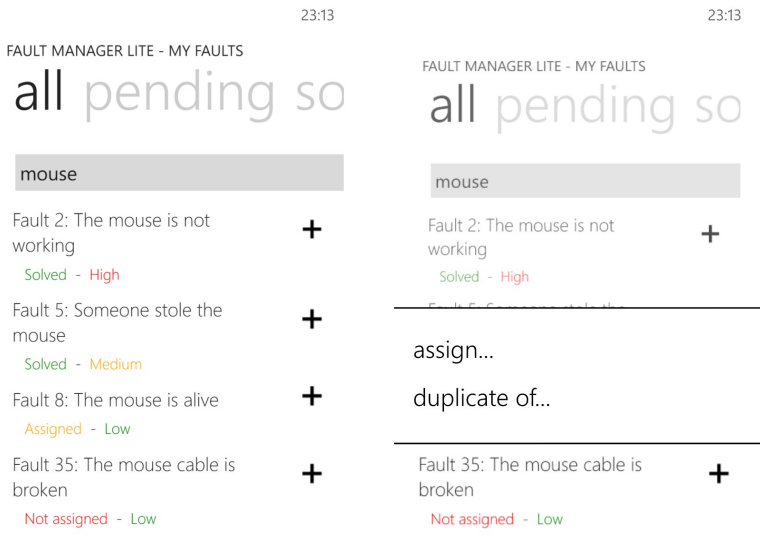

Figure 4.9: The technicians will see additional information in the faults page.

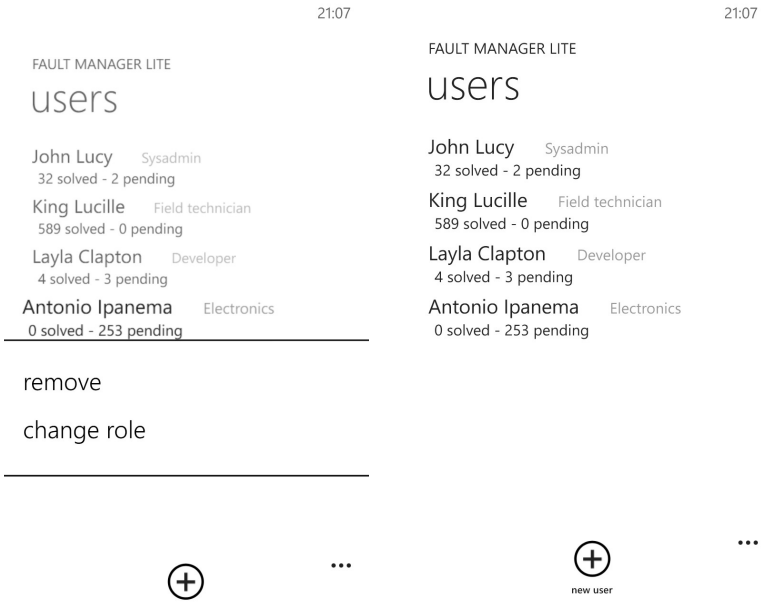

Figure 4.10: The managers will be able to see all the users, change their role or remove them.

Chapter 5

Conclusions

As we stated in the introduction, we think Triforce's solution, presented in this report, is the best tailored to solve current UAM's maintenance structure problems. We can deliver in a timely fashion a reliable, easy to use and useful system that can replace without hassle the currently deployed one.

Our proposal will allow us to listen to every user of the application and react accordingly, applying agile development techniques and continuous feedback and testing.

Fault Manager Lite will be easily and quickly deployed: because of our care and dedication to a polished user experience, formation will not be required for any user as the application and the system will be easy to use for everyone.

We consider the quality to price ratio of our proposal is high, and awarding the project to Triforce would be the best way to invest university assets currently being wasted in an inefficient system.

For this application to work properly, the cooperation with UAM community members is **essential**. We are concerned of it so our decisions aim to achieve this goal. We think our specially crafted, easy to use interface, will smooth the path to it. The option of earning points that can be traded ECTS credits - if the user is a student, of course - will also help to boost adoption. For all other members of the community there is no rewarding system, more than the satisfaction of helping the UAM to improve and get better for everybody on campus.

Chapter 6

References

- UAM Maintenance Fault Report System @ UAM.es
- ITS @ TDC Software
- Clean Up Control @ fourtrack.biz
- Línea Verde Municipal
- Reparaciudad
- In Situ Murcia
- Fault reporting @ madrid.es
- Wunderlist
- Watch @ Apple.com
- Trello
- Asana

Appendix A

Brainstorming

A.1 Ideas about new development

A.1.1 Roles of the Application

We define the user as the reporter of the incidence and the maintenance personnel as the people in charge of fixing the faults.

Every one uses the same application, but it will be shown a different menu depending on the role after the authentication.

- **Admin role:** The person or people in charge of banning and modifying manually what was automatically set by the system.
- **Maintenance person:** You can see, as a Maintenance person, what task has been assigned to you and you can mark them as finished or as a false fault.

  You can also ask for more information about the fault if it's not precise enough.
- **User role:** We may define sub roles such as professor, student or workers (waiters, secretaries, ...) because of the liability of each group of people.

Everyone is a user and everyone can report faults but not every one can be a Maintenance person or an user.

A.1.2 Task priority

We talked about 2 options:

- The user sets the priority of the task and the system has to take into account the reliability of the user, extracted from previous reports, to set a real priority to the task.
- There are tags and categories the user can add to the report and the system computes a real priority to the task based on those.

  This idea would require us to add *others* category, as we can't describe every possible situation.

A.1.3 Location

We have to distinguish 2 cases: if you are inside a building or not.

If you are not inside a building, the location will be automatically determined by GPS technology. However, when faults are located inside of buildings material, it will be necessary to ask the user the exact location. This will be achieved by the use of an interactive map and listings of the available locations inside the buildings (classrooms, laboratories, etc).

We will also use WiFi positioning, based on known access points, to try to infer, if possible, the location of the fault without user intervention.

A.1.4 Manage task assignation

The system automatically assigns tasks, depending on the location of the faults and the Maintenance person and the availability of the maintenance personnel.

Admin can set manually the task assignation.

We should take care of the following situation:

A Maintenance person works a lot and fix every task he was assigned, and, because he is free, the system assigns some more. On the other side, a Maintenance person is not working. As he has lot of pending task, system will not add more task to him.

Plus, we can't just count the number of faults fixed, because there are ones more difficult than others. May be we should add *difficulty of fault* depending on category to try control the how much job you have been assigned.

A.1.5 Graphical interface

We have Guille's application in which we will add just a profile page in which users can see

- the false faults they have reported .
- the fault's reported state.

Every user's profile will be visible to the admin but he won't be able to see any personal data.

The admin can see profiles so before he can ban an user, he can check how many false and repeated faults he has reported.

Maintenance person interface The interface will show by color code the faults assignments classifying them as pending, solved or not started.

A.1.6 Repeated faults

When an user is going to report a fault, the system will suggest if it's repeated showing possible duplicates. If the users says it's a new fault, it will be still marked as a possible repeated fault.

When a task is assigned to a Maintenance person, all possible repeated faults will be assigned to the same Maintenance person. It's Maintenance person's responsibility to say if they are the same or not.

If they are not the same, the task will be enqueued and the system will automatically be assigned to a Maintenance person, maybe the himself or may be not.

A.1.7 Communication

When a fault is fixed, the system will notify the user who reported it.

A.2 Questions to ask

- When an emergency occurs in UAM (e.g., a fire, a flood...), does everyone get an alert?
- Gamified experience? Do we "reward" the user for reporting faults or shall we instead offer detailed reports so the IT department can reward users as they want? (ECTS reward is possible).
- We should take care of the following situation:

  A Maintenance person works a lot and fixes every task he was assigned. As he hasn't any more work assigned, the system assigns him more tasks. On the other side, suppose a Maintenance person is not working. As he has a lot of pending tasks, system will not add more tasks to him.

  If he can assure us every one will work normally, that will solve the problem.

Appendix B

Evaluate current technology

We have summarized here the research on the Internet carried out in order to compare our web application Fault Manager Lite to other similar apps already on the market. For each of these potential competitors, we have mentioned some advantages and disadvantages and extracted some useful ideas that we could adapt or integrate into our own app.

B.1 Suggested solutions

B.1.1 UAM Maintenance Fault Report System

The current maintenance report system of the UAM is based on two simple forms: one for general revisions and the other one only for glass cabinets. These forms can be filled by members of the UAM community to have some facility repaired.

- ✓ The user can choose the service required from a list of general services.
- ✓ The user can specify and describe more the task he/she has asked for.
- ✓ Provides some fields in the form which give a precise location of the problem.
- × The form that it provides is too basic.
- × It does not let you attach a photo of the problem.

Ideas extracted:

- Fields on the fault report formulary to specify a precise location to the maintenance staff.
- Dropdown list allows to easily choose the service required.

Link UAM Maintenance Fault Report System @ UAM.es

B.1.2 Issues Tracking System (ITS)

Tool developed to be used as a task manager for software development teams, which will increase their performance thanks to it. It stores issues such as bugs or rquirements and then makes automatic assignments to the members of the team. Information chart and graphs can also be displayed to get an overall summary.

- ✓ Magnificent software system that surely improves coordination of team members and delivers up-to-the-minute information about current issues (bugs, requirements), fostering better communication and collaboration.

- ✓ Automatic assignment of tasks to members of the team.
- ✓ Extremely detailed form for issues (bugs).
- ✓ Statistic graphics that allow to get a general idea in just a quick view. Issues graphs grouped by: status, type, severity, users and priority.
- ✓ Everyone involved in a project can obtain status, reports, charts and graphs showing trends and problem areas.
- × Not user-friendly at all, as it has lots of different features and lots of details that complicate its usage for new users.
- × Old-looking user interface.

Ideas extracted:

- The automatic assignment of tasks can suit well in our app, so that the maintenance tasks would be assigned automatically to the repairmen. This may require to separate the repairmen into groups, depending on the abilities they have.
- The issues form, although seems too detailed, could give us good ideas for our fault report forms.
- The team functionality is also desirable, as all the members of the technical staff would have access to up-to-the-minute information of pending and solved faults. This would solve coordination problems.
- The system statistics and summary charts could be useful to detect problem areas that should be revised more often.

Link ITS @ TDC Software

B.1.3 Clean Up Control

This application is used to manage the quality control of the cleanse service in any kind of area, no matter its structure. It saves information about every inspection made and shows the cleanse levels of each zone, giving more priority to those that have failed the inspection or it is still pending.

- ✓ Profile information about clients, inspectors and managers.
- ✓ Two modules: PC and PDA app (mobility).
- ✓ Locations organised in a tree structure, which makes easier to find them in the app.
- × Synchronization between the PC and the PDA requires direct cable connection or a modem accessory for the PDA.
- × Obsolete user interface.
- × Does not cover any kind of team coordination.

Ideas extracted:

- Organising all the locations in a tree structure is a good idea.
- The revisions of problem areas could be timetabled in the same way as they are in this application

Link Clean Up Control @ fourtrack.biz

B.1.4 Línea verde

This website includes an Intranet for the town council, through which the incidents and facilities faults stated by the citizens are managed. Furthermore, it includes the visualizations of statistics and reports of the management.

- ✓ Great mobility, being accessible by their website and also by smartphone (IOS and Android apps are available).
- ✓ Automatic geolocation of the user when he/she sends a report.
- ✓ Photo of the problem can be attached to your report.
- ✓ Incidents are organised into different types to make easy to specify your problem.
- ✓ Simple and user-friendly report system.
- × Closed categories of faults stop the user from reporting any other kind.

Ideas extracted:

- Faults hierarchy or main categories into which we could organise our installations faults.
- Mobility (tablets, smartphones), geolocation and photo of the fault are already features we were planning to add to our system.

Link Línea Verde Municipal

B.1.5 Reparaciudad

This extremely user-friendly application allows the user to report an incident in any street of Spain, choosing its typology and describing or uploading a photo of it. It also has at the users disposal a map containing the incidents reported by everyone, which serves to detect problem areas very easily.

- ✓ Complete mobility, as the service is offered in PCs (internet browsers), but also on smartphones app (Android, IOS, Blackberry and Windows Phone platforms).
- ✓ Hierarchy of incidents.
- ✓ It shows a map with icons which represent the last incidents reported. Moreover, it also shows some useful information about those reports (type, date, exact location, status and other comments).
- × Registration is needed in order to report incidents.

Ideas extracted:

- The map of the incidents is a superb idea which we will try to add to our system.
- The report system allows voting, which could give us an idea of the users that want that incident to be fixed.

Link Reparaciudad

B.1.6 In Situ Murcia

The town council of Murcia offers in this website a way to report any kind of incident or fault in the facilities of the city and request their repairs. This website goal is similar to reparaciudad, but limited to incidents in the region of Murcia.

- ✓ No login needed, you just have to introduce your email and mobile phone number in order to send a report.
- ✓ Very user-friendly and modern website.
- ✓ It shows a map with icons which represent the last incidents reported. Moreover, these icons follow a colour code depending on their current status.
- × No mobility alternatives.
- × It is mandatory to send your geolocation in order to report an incident, becuase the service is limited to the region of Murcia.

Ideas extracted:

- The colour code of the icons in the map make it possible to get a general idea of the faults status in just one look.

Link In Situ Murcia

B.1.7 Madrid council fault report website

The town council of Madrid offers in this website a way to report any kind of incident or fault in the facilities of the city and request their repairs. This website goal is similar to reparaciudad, but limited to incidents in the region of Madrid.

Trying to access this website, it says that we need some special certificates (electronic DNI) to report any kind of incident, so we have not been able to see the forms.

- × Requires authentication with special certificates, which makes most of the people to leave the website instead of reporting faults of their city.
- × Apart from links to two forms (which we could not access to), this website does not offer anything more: no map with incidents, no faults history, nothing special.

Link Fault reporting @ madrid.es

B.2 Own research

B.2.1 Wunderlist

- ✓ Nice interface to take into account for the task management.
- ✓ Maybe shared lists so every handyman in a given building can see pending tasks assigned to other people.
- ✓ See completed tasks: faults you've fixed..
- ✓ Deadline for each fault, automatically set by the system taking into account the priority and the category.
- × Hard to track tasks if there are a lot of them.
- × An useful search system is not present.

Ideas extracted

- Design carefully the interface so it's easy to use.
- Include a search system.

Link Wunderlist

B.2.2 Apple Watch

✓ Add some sort of really fast communication between handymen, so they don't have to call one another.

✓ Add few types of default messages like "I'm going" or "Impossible" or "Need help".

× Dedicated hardware required.

Ideas extracted:

- What about broadcasting messages to all handyman? Or may be just to a few of them? Maybe making groups and you send notification to a group.
- We could add some not so fast but more efficient communication way for handyman.

Link Watch @ Apple.com

B.2.3 Trello

✓ Taskboard with access control (set to public or private).

✓ Add one more state to task (pending, working, finished) better than wunderlist system (done or todo).

✓ Really useful to get a global glance of all tasks.

× Data is not easily exportable.

Link Trello

B.2.4 Asana

✓ Deadline for tasks given in days or even in hours from now.

Link Asana

Appendix C

Meetings

C.1 Meeting Announcements

C.1.1 Meeting Announcement - 1

From: Guillermo Julián Moreno
To: Iván Márquez Pardo, Víctor de Juan Sanz.
Date and Time: 26/01/2015, 14:00.
Place: third floor of the EPS Library.

Purpose First meeting as a team. Start working on the project.

1- Agenda:

1. Overview of the project assigned.
2. Divide work and start the project (if possible).

2- Decision Follow-Up: None.
3- Documentation:

1. Statement/definition of the project.

C.1.2 Meeting Announcement - 2

From: Víctor de Juan Sanz
To: Iván Márquez Pardo, Guillermo Julián Moreno.
Date and Time: 28/01/2015, 14:00.
Place: third floor of the EPS Library.

Purpose Brainstorming session.

1- Agenda:

1. Brainstorming session.
2. Gather ideas in a draft paper.

2- Decision Follow-Up:

1. The three of us have read the whole documentation.
2. We have to come to this meeting with some ideas for the project.

3- Documentation:

1. Statement/definition of the project.

C.1.3 Meeting Announcement - 3

From: Iván Márquez Pardo.
To: Víctor de Juan Sanz, Guillermo Julián Moreno.
Date and Time: 29/01/2015, 11:45.
Place: Science building, module 11.

Purpose Brainstorming session.

1- Agenda:

1. Second (and final) brainstorming session.
2. Gather ideas in a draft paper.

2- Decision Follow-Up:

1. The three of us have read the whole documentation.
2. We have to come to this meeting with some ideas for the project.

3- Documentation:

1. Brainstorming draft paper.

C.1.4 Meeting Announcement - 4

From: Guillermo Julián Moreno
To: Iván Márquez Pardo, Víctor de Juan Sanz.
Date and Time: 02/02/2015, 19:00.
Place: third floor of the EPS Library.

Purpose Review of work.

1- Agenda:

1. Review of work already done.
2. Assignment of new tasks for the week.

2- Decision Follow-Up:

1. The three of us have completed their previous tasks.

3- Documentation:

1. Statement/definition of the project.

C.1.5 Meeting Announcement - 5

From: Víctor de Juan Sanz.
To: Iván Márquez Pardo, Guillermo Julián Moreno.
Date and Time: 09/02/2015, 14:00.
Place: third floor of the EPS Library.

Purpose Preparation of the interview.

1- Agenda:

1. Revision of subsystems.

2. Preparation of the interview with the maintenance manager.

2- Decision Follow-Up:

1. We still have to approve the subsystems which form the application.

3- Documentation:

1. Parts of the project already done, which would be analyzed to clarify obscure parts in the interview.

C.1.6 Meeting Announcement - 6

From: Iván Márquez Pardo.
To: Víctor de Juan Sanz, Guillermo Julián Moreno.
Date and Time: 11/02/2015, 11:35.
Place: Science building, module 11.

Purpose Overview of the interview.

1- Agenda:

1. Overview of the interview.
2. Clarify some aspects of the interview.
3. Correction of some parts of the project.
4. Review of the presentation.
5. Practice the oral presentation.

2- Decision Follow-Up:

1. Subsystems are now fixed.

3- Documentation:

1. Interview craft paper.

C.2 Meeting Minutes

C.2.1 Meeting Minutes - 1

Date and Time: 26/01/2015, 14:00.
Length: 28 minutes.
Participants: Guillermo Julián Moreno, Víctor de Juan Sanz, Iván Márquez Pardo.

Topics:

1. Overview of the project assigned.
2. Divide work and start the project (if possible).

Decisions Made:

1. The next meeting will consist in a brainstorming session.

Activity	Person Responsible	DeadLine
Read the whole documentation given in order to comprehend better the main problem we have to solve and think of general ideas for our project proposal	Víctor de Juan	29/01/2015
Read the whole documentation given in order to comprehend better the main problem we have to solve and think of general ideas for our project proposal	Guillermo Julián Moreno	29/01/2015
Read the whole documentation given in order to comprehend better the main problem we have to solve and think of general ideas for our project proposal	Iván Márquez Pardo	29/01/2015

C.2.2 Meeting Minutes - 2

Date and Time: 28/01/2015, 14:00.
Length: 1 hour.
Participants: Guillermo Julián Moreno, Víctor de Juan Sanz, Iván Márquez Pardo.

Topics:

1. Brainstorming session.
2. Gather ideas in a draft paper.

Decisions Made:

1. We could not finish the brainstorm, so a second brainstorming session is needed.

Activity	Person Responsible	DeadLine
Revision of the current state of the brainstorm	Víctor de Juan	29/01/2015
Look for contradictions (if any) between the ideas from the brainstorm and the project definition	Guillermo Julián Moreno	29/01/2015
Think of more original ideas to add in the brainstorming final session	Iván Márquez Pardo	29/01/2015

C.2.3 Meeting Minutes - 3

Date and Time: 29/01/2015, 14:00.
Length: 47 minutes.
Participants: Guillermo Julián Moreno, Víctor de Juan Sanz, Iván Márquez Pardo.

Topics:

1. Second (and final) brainstorming session.
2. Add new ideas to the draft paper.
3. Start with other parts of the project.

Decisions Made:

1. Revise ideas and finish the brainstorming paper.
2. Start with other parts of the project.

Activity	Person Responsible	DeadLine
Write down clearly in Latex format all the ideas obtained from the brainstorming session	Víctor de Juan	02/02/2015
Start thinking about the design of the application	Guillermo Julián Moreno	02/02/2015
Write down the introduction of the project	Iván Márquez Pardo	02/02/2015

C.2.4 Meeting Minutes - 4

Date and Time: 02/02/2015, 19:00.
Length: 15 minutes.
Participants: Guillermo Julián Moreno, Víctor de Juan Sanz, Iván Márquez Pardo.

Topics:

1. Review of work already done.
2. Assignment of new tasks for the week.

Decisions Made:

1. Corrected some obscure parts of the work done.
2. It has been arranged the date for the next meeting, in which we will prepare the interview with a member of the maintenance staff will.

Activity	Person Responsible	DeadLine
Research on the Internet about similar applications in order to get new ideas to our project.	Víctor de Juan	09/02/2015
Define the subsystems of the application.	Guillermo Julián Moreno	09/02/2015
Mandatory part of the research on the Internet about certain applications of websites whose functionalities are similar to the ones we have to offer.	Iván Márquez Pardo	09/02/2015

C.2.5 Meeting Minutes - 5

Date and Time: 09/02/2015, 14:00.
Length: 24 minutes.
Participants: Guillermo Julián Moreno, Víctor de Juan Sanz, Iván Márquez Pardo.

Topics:

1. Revision of subsystems.
2. Preparation of the interview with the maintenance manager.

Decisions Made:

1. Questions for the interview with the maintenance manager have been prepared.
2. Time to start preparing the presentation of our project proposal.

Activity	Person Responsible	DeadLine
Prepare the presentation: Susbsystems and Requirements.	Víctor de Juan	12/02/2015
Prepare the presentation: Mock-ups of the application.	Guillermo Julián Moreno	12/02/2015
Prepare the presentation: Introduction, Project Definition, Scope, Conclusions.	Iván Márquez Pardo	12/02/2015

C.2.6 Meeting Minutes - 6

Date and Time: 11/02/2015, 11:35.
Length: 51 minutes.
Participants: Guillermo Julián Moreno, Víctor de Juan Sanz, Iván Márquez Pardo.

Topics:

1. Overview of the interview.
2. Clarify some aspects of the interview.
3. Correction of some parts of the project.
4. Review of the presentation.
5. Practice the oral presentation.

Decisions Made:

1. If we complete all tasks on time, only the reflection paper would remain undone.

Activity	Person Responsible	DeadLine
Finish the Functional and Non-Functional requirements of the project.	Víctor de Juan	20/02/2015
Finish the mock-ups of the project.	Guillermo Julián Moreno	17/02/2015
Finish the project definition and conclusions.	Guillermo Julián Moreno	20/02/2015
Write down the clean-up version of the interview, research, meetings, introduction and scope of the project.	Iván Márquez Pardo	12/02/2015

Appendix D

Interview

On February 10th 2015, we had an interview with the manager of the maintenance service of the UAM, Mr. Roberto. In the interview we clarified several aspects of the maintenance service in order to have a more detailed view of the problem and know by first hand how things are performed right now.

This section gathers the most important points of this interview between us, the Triforce representatives, and the maintenance manager.

Triforce: *Can you please tell us more about how is the maintenance service organized right now?*
Mr. Roberto: The maintenance service is quite decentralized in several groups depending on the tasks they perform. These groups are auto-organized, assigning repair tasks on their own and they don't contact me usually. This division into independent groups and the lack of coordination between them sometimes cause some budget and personnel problems.

Triforce: *So, basically, you want us to try to centralize the maintenance service to sistematize the task assignment, right?*
Mr. Roberto: Yes, indeed.

T.: *How many different departments are there in total?*
R.: There are 9 different departments: Information Technologies, Electricity, Plumbers, Elevators, Heating, Air conditioning, Cleaning, Gardeners and Garbage collectors.

T.: *And you said they are auto-organized, but how?*
R.: They all received their corresponding repair tasks and they are manually assigned to members of each department. However, there are two departments that work in a different way: the Cleaning and the Information Technologies department.

T.: *Can you give us more details about the Cleaning department?*
R.: Yes. The Cleaning department is formed by 8 different cleaning companies, each of those has a manager for the building they are in charge of. Your application should take that into account, because cleaning tasks will be assigned to the cleaning manager of the building the fault has been encountered. As they are different companies, that manager will internally assign that task to one of its employees; your application should not interfere in that.

T.: *And what about the Information Technologies department? How do they organise?*
R.: Well, the Information Technologies department has its own application to automatically assign tasks. Their application is based on emails which are sent by the users, who fill a form in the maintenance service website. The system reads those emails and assigns the repair task automatically to a member of the technical staff; if it can't assign it automatically, then the manager of this department reads it and assigns it manually.

T.: *You want us to overwrite this system or maintain it along with our app?*
R.: The organization of the Information Technologies department should be the same as the others, so this system must disappear when you finish setting up your app.

T.: *Ok, we will overwrite the IT system... Can we count on their collaboration?*
R.: Yes, the whole maintenance service has agreed to start using your app in order to encourage the collaboration and coordination between departments.

T.: *Perfect. We have clarified some details of the old organization of the maintenance service, now let's concentrate on the new organization, our application. Are the staff used to mobile phones?*
R.: Yes, they are used to smartphones, but the more user-friendly you could make your app, the better.

T.: *We will take it into account while making the mock-ups, don't worry about it. We had thought that it could be useful to include a system of alerts and notifications so that, in case of emergency (fire in a building, for example), a user could press a big button and all the other users of the app would receive an alert telling them information about the danger and its location.*
R.: It could be useful, indeed. It is a good idea to include it in the app.

T.: *We are pleased you like it. Would you like us to also include statistics of faults so that the manager could visualize them?*
R.: For me, it is pointless to create any type of statistics about faults. The only thing I need is a fault history, but no fault statistics.

T.: *In that case, fault statistics discarded. We thought that the urgency of faults should be set automatically, but users could classify faults as urgent while filling the report and their opinion would be automatically taken into account depending on their previous reports. Is this correct for you?*
R.: In the beginning, yes, it is. But there is one thing I need you to include: I must be able to classify manually a fault as urgent in order to be able to accelerate certain repairs.

T.: *No problem, we will include that option in our system. One last question: what happens if users are not able to identify the category of a fault, how can we assign it to the corresponding member of the technical staff? For example, if there is a problem with the watering system and the user does not know if its a problem of plumbery or for the gardeners, what do we do with that report?*
R.: In case there are any unclassified or extraordinary reports, I, as general maintenance manager, will be in charge of reading them and manually assign their repair task to the correct technician.

T.: *Perfect, in that case, we have finished our interview. Thank you for your help, Mr. Roberto.*
R.: You are welcome, I am looking forward to see your application.