

**Volunteers’ Management System**

**Final Report**

**By:**

**Apollo Group**

# Table of Contents

[Table of Contents 2](#_Toc262232601)

[1 Product Explanation 3](#_Toc262232602)

[1.1 Problem Statement 3](#_Toc262232603)

[1.2 Product Description 4](#_Toc262232604)

[1.3 Features 4](#_Toc262232605)

[2 Deliverables 6](#_Toc262232606)

[2.1 Project plan 6](#_Toc262232607)

[2.2 Risk plan and Risk list 6](#_Toc262232608)

[2.3 Quality Assurance Plan 7](#_Toc262232609)

[2.4 Test Plan 7](#_Toc262232610)

[2.5 Test Cases 9](#_Toc262232611)

[2.6 Test Report 9](#_Toc262232612)

[2.6.1 Function Test 9](#_Toc262232613)

[2.6.2 User interface test 9](#_Toc262232614)

[2.6.3 Load, stress, and performance test 9](#_Toc262232615)

[2.6.4 Security and access control test 9](#_Toc262232616)

[2.6.5 Code coverage analysis 10](#_Toc262232617)

[2.6.6 Code static analysis 10](#_Toc262232618)

[2.6.7 Software qualification test 10](#_Toc262232619)

[2.6.8 Inspection white box Testing 10](#_Toc262232620)

[2.6.9 Unit testing 10](#_Toc262232621)

[2.7 Use-cases 10](#_Toc262232622)

[1.1 Software Architecture Document 11](#_Toc262232623)

[1.2 Software Design Document 12](#_Toc262232624)

[2.8 Manager’s manual 13](#_Toc262232625)

[2.9 Volunteer’s manual 13](#_Toc262232626)

[2.10 Web Portal Installation Guide 13](#_Toc262232627)

[3 Conclusion 13](#_Toc262232628)

# Product Explanation

## Problem Statement

Earthquakes, hurricanes, terror attacks and wars are some examples of disasters and crises which are happening in our world that causes the death and loss of many people. As a common result of these crises, the normal city life is corrupted. Even after the crisis, problems remain as epidemic or starvation.

Many organizations such as military forces, civil defense organizations, Red Cross or other international help organizations have helped people to overcome the difficulties and challenges during crises. However, most of them operate through their trained members. Militaries have educated soldiers, Red Cross has members with certificated training and Civil Defence in Sweden has hired experts. This puts a limitation for a regular citizen to join any of such organizations. Many of these organizations require the accomplishment of different training programs and attendance of regular meetings, more over some even require membership fees. Their organizational structure is not built on regular citizens but on their educated active members.

Beyond the difficulties for being a member of help organizations, there is another obstacle. People don’t want to be a member of such organizations. Unless they faced to a crisis just near them, they are not so eager to join those help organizations like Civil Defence or Red Cross. However, when they face a crisis, if they have the opportunity, most of them want to help other people who are in difficulty.

The problem is that; help organizations miss out regular citizens who can assist them in crisis time. Help organizations form their structure before the crisis happens. However, people don’t want to join or they can’t join these organizations before crisis time. People only consider joining help organizations if there is a crisis but currently there is no quick way for this.

Ad de Raad, UN Volunteers Executive Coordinator, emphasizes the problem clearly in the World Conference on Disaster Reduction in Japan, 1995: “In the wake of disasters, we have seen repeatedly how considerably large numbers of generally unskilled, untrained, and unaffiliated citizens are mobilized through volunteerism to lend a hand. However, much of it is ad hoc and uncoordinated. At best, this can result in a considerable loss of effectiveness. At worst, undirected volunteers can become part of the problem in a disaster situation.”[[1]](#footnote-1)

If we imagine a city wide earthquake that 40% of all citizens die, 20% are under wreck and remaining 40% is fine, then we can assume that, at least 10% of all city population can help rescue operations. For a small city of 100.000 populations, that means 10.000 people. Even if a so small portion of them willing to attend help facilities, that size would be great, many times greater than all of help organizations in that city.

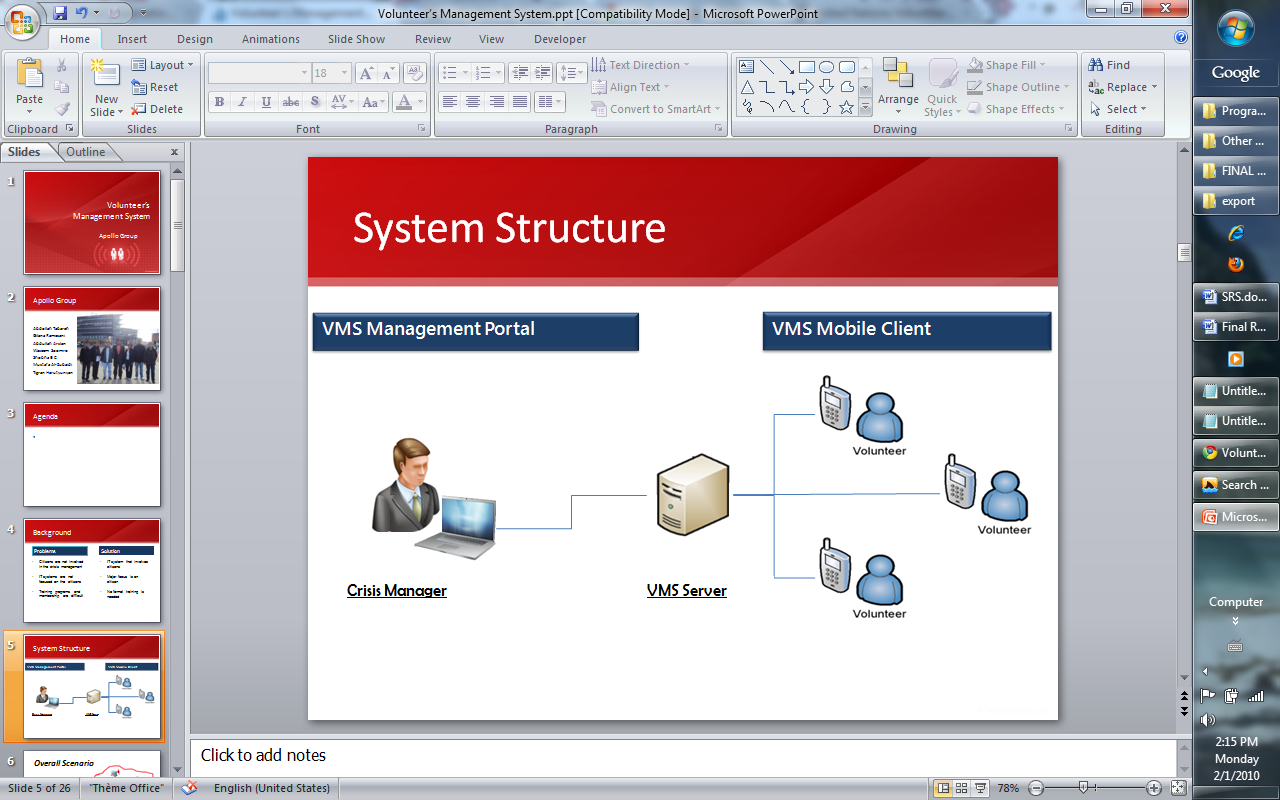
If there is a system that allows people to join help facilities not only before crisis but also after a crisis happens, and that enables crisis manager to manage a huge amount of people effectively, it would be a great solution.

This solution is our system which is Volunteers´ Management System (VMS).

## Product Description

VMS is a composite software system that establishes a bridge between help organizations and citizens in crisis times. It is based on the idea that people consider helping others when there is a crisis. VMS provides a quick and easy way to attend the help facilities and help crisis manager to organize thousands of volunteers in any crisis efficiently and effectively.

VMS includes a central web portal to manage a crisis and a mobile application as an interaction channel for volunteers. The following figure shows the way of communication between the volunteers and the manager through VMS.



When a crisis occurs, volunteers download and install the mobile application and become a part of rescue operations by registering to the system. At the other hand, crisis manager defines the problems, that are incidents in VMS context, and all of the resources needed to resolve the problems and that’s it. The unique automatic requesting feature of VMS automatically asks and gathers all the required resources to solve the problems based on the nature of the problem and volunteers locations. Thus in a few minutes hundreds of volunteers can be organized and directed to a problem for help.

Additionally, VMS utilizes volunteers as an information source for reporting new incidents and also the status of ongoing incidents. At the chaos times which immediate response is crucial and official organizations become impotent in reporting and notification of new problems, VMS can be a vital information source.

Thanks to the easy-to-use user interface, crisis managers can monitor and manage the crisis area and all the problems easily on the crisis map. Manager operates trough web browser, since it is a web application he can access it from any place, which is a very useful during crisis time. This system also uses xml web service technology, which provides the possibility to increase the capability to thousands volunteers by including more hardware resources.

Being developed with J2MEE technology, the Mobile Application of VMS can run on almost all mobile phones; thus avoid the compatibility issues for many mobile phones.

## Features

The following features make the Volunteers’ Management System appropriate for handling crises.

1. **Resource Management**

Volunteers and their useful stuff which they can provide to the needy will be managed easily by the managers. VMS intensively assists the crisis manager for resource management by selecting the volunteers according to region, type of the crisis and some other constraints.

Manager creates request by defining the need list and set the severity of the incident. Once the send button is pressed, rest other tasks like selecting the volunteers and collecting the resources will be automatically handled by the VMS.

1. **Registration and membership**

Before using the VMS, the user should be registered. Based on the user credentials, different levels of access to the system are provided.

Registration procedure requires some basic information about the members and gives user name and password for subsequent logging into VMS.

1. **Profile Management**

System keeps tracks of all the information about the members to help the crisis affected people efficiently. Members at any point of time can change/delete their profile; subsequently the system adjusts to current requirement. For example if a volunteer adds additional training information, then the person will be selected for a crisis which requires that particular training.

1. **Incident Management**

Any number of incidents can be managed by the VMS. System behaves differently to the different incidents. There can be different needs, different resource assignments, different states like active/resolved for each incident. More than one incident can be raised in a crisis and each incident will have individual need list and requirements, these will be handled by VMS easily according to the nature of the incident.

1. **Information Gathering**

Volunteers can be used as information source to address the crisis on time with more efficiency. The volunteers can send two types of information report to the help organization. They are

* **Reporting new incidents**

Volunteers can send new incident report, which helps the manger to create new incident in the VMS and helps in handling the crisis early.

* **Reporting the statues of the existing incidents**

Volunteers can send status of the incident which they are working on. This helps in knowing the statues of the incidents, which might help in handling the crisis. The information will be filtered and updated accordingly to the VMS.

1. **Communication**

Help organization center communicate with volunteers by sending messages via web portal and the volunteers reply them via mobile application. There are different types of communication in VMS like warning messages, request messages.

* **Warning the user**

Management Center can warn users according to their location or any other information like age, gender. The alert messages will make the volunteers aware of possible dangers.

* **SendING Request for the help**

Manager can send a help request to volunteers based on their locations and the type of crisis. By sending help request, the manger gets the response from volunteers. With the help these responses further course of actions will be incited by the manger to address the crisis.

1. **Log Analysis and Backup**

VMS system logs every event and information related to each crisis for future use. Logs will be analyzed and used in managerial aspects, for achieving this system has a feature to generate different reports such as historical status of crisis etc.

Different logging levels can be set like a detailed level which allows to records every action in camper to the log levels will record only the major actions.

# Deliverables

## Project plan

Project plan is one of the major documents of software projects. It is the document which guides the project manager during the project.

Project management plan has sections which are very important for the project and project manager have to follow the document all the time.

Apollo group’s project plans contains all the necessary information for the carrying out the project. This document you can find in the folder “ProjectPlan” within the deliverables under the name “FinalProjectPlan”.

Project plan contains the objectives and the scope of the project.

It contains information about the group member of Apollo group. Every member had two roles during the project. More information about the roles and responsibilities can be found in the “FinalProjectPlan” document.

The project’s Work Break Down structure was created and according to that structure estimations were performed in the beginning of the project and included in the project plan. According to the project estimations the project schedule was created.

The schedule of the project shows the phases and the iterations of the phases. Every phase ends with the milestone which says which goals and objectives need to be reached in the particular phase. Phases and milestones had been planed according to RUP. Every phase consists of the iterations. Each iteration has its objectives and the tasks needed to be done in that iteration.

Another important part of the project plan is the measurement plan. Measurement plan is a tool which allows the project manager to measure the progress of the project and to see the status of the project.

More about the project plan you can read in “FinalProjectPlan” document which is located in “ProjectPlan” folder within the deliverable.

## Risk plan and Risk list

During the project expected and unexpected problems can happen. That kind of problems can have different impact on the project. These problems are the risks of the project. To mitigate the risks of the project the Risk plan was created. It is a guideline for the team to identify the risks of the project and record them. The important part of the risk plan is the risk list.

Risk list is a document which contains the list of risks of the project which had been identified by the team. The structure of the Risk list document is explained in the Risk plan.

Risk list contains the short description of the risks, probability (probability of the occurrence of the risk in the percentage), severity of the risk, impact which is calculated according to probability and severity and the mitigation plan for the particular risk.

More about the risk plan and risk list you can find in “RiskManagementPlan” and “RiskList” documents which are located in “ProjectPlan” folder within the deliverable.

## Quality Assurance Plan

Quality Assurance Plan will define the quality objectives, metrics and measurements, activity details, schedule and processes needed to identify and flag if something goes wrong in the project. These warnings should come at appropriate time to compensate the wrong things with the least cost for the project.

The quality criteria of the product are defined based on ISO 9126 quality standard in a measurable way and with clear categorization. You will find more details about these criteria in a document named “ProductQuality” in a folder named “QualityAssurance” within deliverables. To achieve those specified product quality attributes and also project quality attributes, several types of activities are defined and detailed in quality assurance plan and other related documents. They’re summarized in the following table.

|  |  |
| --- | --- |
| Activity | Timing |
| Document Reviews | For significant documents on finish |
| Code Reviews and Inspection | For each development module on finish |
| Audit | At the end of every iteration and phase |
| Test Planning | Early in Elaboration phase |
| Test Execution | For each development module on finish |
| Test Coverage Evaluation, logging, and Reporting | For overall system test |
| Quality Reporting | At the end of each iteration |
| Acceptance Test | At project delivery |
| Defect Tracking | Throughout the project |

In order to perform different kind of inspections, reviews, and audits, checklists and guidelines have been provided in Quality Assurance Plan appendices and inside the folder named “Checklists” which is inside the “QualityAssurance” folder within deliverables.

There also documents related to details of testing plans and execution reports which will be explained later in this document.

## Test Plan

Test plan is a part of quality assurance plan and it specifies all the necessary activities to test the main deliverable of this project, working system. An outline of the test plan for this project is provided here, but for more details you can refer to the document named “TestPlan” in a folder named “QualityAssurance” within deliverables.

According to test plan, this project employs white box and black box testing as the test strategies in order to reach to an appropriate level of quality for software.

The white box testing will be performed in the following three forms:

* Code coverage analysis
* Code static analysis
* Software qualification test

The black box testing will be performed mostly based on test cases. Test cases are captured from different places. There are test cases for:

* Functionality
* Performance
* User interface design
* Security
* Other product quality attributes

Test plan also mentions that the test cases will be executed in several stages. Unit test, iteration test, integration and system test. The integration and system test will be the same and will perform together. It means that there will not be two separate tests for integration and system tests. So to perform system test, all of the test-cases will be executed. The test cases will be prioritized and will be executed based on their priority. At the beginning of overall system test, using a method called “Mark recapture”, the total number of defects for the software should be estimated. This will happen by executing at max 5% of test cases. In the following table you’ll find the summary of testing types.

|  |  |
| --- | --- |
| Test Type | Description |
| Function Test | Ensure proper application functionality, including navigation, data entry, processing, retrieval, update, and search. |
| UI Test | Assure user interface of the software conforms to User Interface Design Guidelines defined as a part of quality assurance plan of the project. |
| Load, Stress, and Performance Test | The objective of performance testing is to demonstrate that a system functions in accordance with its performance requirement specifications regarding acceptable response times. The analysis of performance test results helps support performance tuning. Stress testing involves the process of running the client machines in high-stress scenarios to see when and if they break. |
| Security and Access Control Test | To test if system is protected from unauthorized access |
| Code Coverage Analysis | To verify and assure about the sufficiency of test-cases |
| Code Static Analysis | To find the system defects in development level before executing the code. |
| Software Qualification Test | To verify and assure about the needed standards and quality of coding. |
| Inspection | To assure accuracy of algorithmically important parts of the code. |
| Unit Testing | To assure accuracy of utility modules and methods which are widely used throughout all subsystems of the project. |

You can see more details about these testing types in Test Plan document.

The following tools decided to be used during testing for different purposes.

|  |  |
| --- | --- |
| Purpose | Tool |
| Reporting bugs | Assembla Bug Tracking System |
| Static analysis | JetBrain Resharper |
| Performance, loud and stress automated test | Selenium |
| Code coverage analysis | NCover |
| Unit test generation | Visual studio.net |
| Logging test execution | MS excel |
| Load and stress test | MS visual studio.net |

## Test Cases

“Equivalence Class Portioning[[2]](#footnote-2)” method is used to define test cases in the VMS project. For each user interface section, all of the input variables are listed and then their equivalence classes are defined. After that several test cases are produced to efficiently cover the project as much as possible.

All of the test cases are documented in excel files without traditional step-by-step style. Each excel file contains numerous test cases that are contextually similar and related to same user interface section. There are two pages in each excel file. First page contains equivalence classes for all of the input types. The actual test cases are contained in the second page with a descriptive name, equivalence class selection for each input type and the expected result.

Besides test case files, there is also a data dictionary file which contains description of all input types that are derived from use cases.

Test Case folder contains all of above mentioned files and can be seen for further information.

## Test Report

The Test Report document summarizes the results of the tests which performed during development and at the end of the project before delivering to stakeholders. A brief description of each test has been provided here. For more details about each of them please refer to “TestReport” document in “QualityAssurance” folder within deliverables.

### Function Test

Function test has been performed in three stages during the project using test-cases.

1. Immediately after development by developers.
2. At the end of iterations by testers.
3. In transition phase of the project as a part of overall system testing.

During the overall system test 164 records of test-case execution logged in TestLog.xls. The following table shows the result.

|  |  |
| --- | --- |
| Result | Frequency |
| Passed | 139 |
| Failed | 25 |
| Total | 164 |

All the bugs found during function test have been reported into bug tracking system.

### User interface test

The inspection meetings were held by quality assurance team using those guidelines and checklists, for the finished functionalities at the end of each iteration.

### Load, stress, and performance test

Because of the shortage in time and resources, this test didn’t perform and skip.

### Security and access control test

This test is considered to protect the system from unauthorized access. For this reason test cases were defined and were executed at the system test level.

### Code coverage analysis

To assess the sufficiency and adequacy of the test cases, runtime code coverage analysis is performed. NCover is the tool that is used for this purpose. Here is the total coverage analysis result summary:

|  |  |
| --- | --- |
| Item | Value |
| Symbol Coverage | 70.64% (2413 of 3416) |
| Branch Coverage | 71.23% (1067 of 1498) |
| Method Coverage | 68.72% (512 of 745) |
| Cyclomatic Complexity Avg | 1.53 Max:18 |
| Modules | 4 |
| Namespaces | 7 |
| Classes | 94 |
| Methods | 745 |
| Documents | 63 |

### Code static analysis

It performed by using a tool called “JetBrain Resharper” during development. It was the responsibility of developer to consider the warning of the tool for potential defects and fix them.

### Software qualification test

The quality assurance team of the project held the code inspection meeting to check the quality of the code according to the guidelines defined in Quality Assurance Plan of the project.

### Inspection white box Testing

In order to have confidence for some algorithmically-complex stored procedures and also functions, manual inspection and some level of code tracing are used.

### Unit testing

In addition to all functionality which have been tested as functionality units using stubs and drivers, The “Login” method of the web service have also been unit tested using .net unit test tools.

## Use-cases

For the use-case driven development methodology like RUP the use-cases are the functional requirements. The “Use-case” folder within the deliverable contains all the final use-cases and “Estimations of use cases and completion status” excel file which contains the estimations of the use-cases.

The structure of the use-case was created during the Industrial Challenge course. It has the following structure:

1. Use-case name
2. Description

Description of the use-case.

1. Actor

The external user of the system who perform the actions in the use-case.

1. Identifier

Unique identifier of the use-case.

1. Traceability

The requirement number that use-case had been created.

1. Pre-condition

The conditions which are mandatory to perform the use-case in the beginning of the use-case.

1. Post-condition

The conditions of the system after performing the use-case.

1. Main path

Main path of the use-case. Steps explain the user’s action and the system response. Main path is marked with the unique number which is the functional requirement number.

1. Alternate path

Alternate path of the use-case. The use-case can be without alternate path as well as with one or more alternate paths. Every alternate path is marked with the unique number which is the functional requirement number.

1. Non-functional requirements

The quality attributes for which the use-case is important and satisfies the quality attributes partially of completely.

1. Issues

Issues connected with the use-case.

“Estimations of use cases and completion status” excel file shows the estimations of the use-cases. It shows the difficulty of the use-case from the development point of view which can be simple, average or difficult, the weight of the use-case in the system and the size of the use-case in overall system in percentage.

According to those estimations during the project and in the end of the project it was possible to calculate the percentage of the system’s developed part.

## Software Architecture Document

SAD was produced by the architects based on quality attributes and functional requirements. The document is located in folder “ArchitectureDesign” within the deliverables under the name “SAD”.

SAD explains the goals and constrains of the VMS architecture. The document contains the important stakeholders of the VMS architecture and explains why those stakeholders are important for the architecture. The important stakeholders for the architecture of VMS system are volunteers, crisis managers and developers.

The key requirements are discussed in details in the document. The key requirements are those which are very important for the VMS and the architects had to realize these requirements in the architecture.

SAD explains the architecture of VMS for different stakeholders. For this purpose, VMS architecture has four different views for each stakeholder. Those are Logical view, Process view, Deployment view and Implementation view.

The Logical view explains the logical model the VMS architecture. It shows the architectural patter of the VMS which is layered architecture, the layers of the architecture, order and communication between the layers and explanation of the layers. Figure 2 shows the Logical view of VMS.

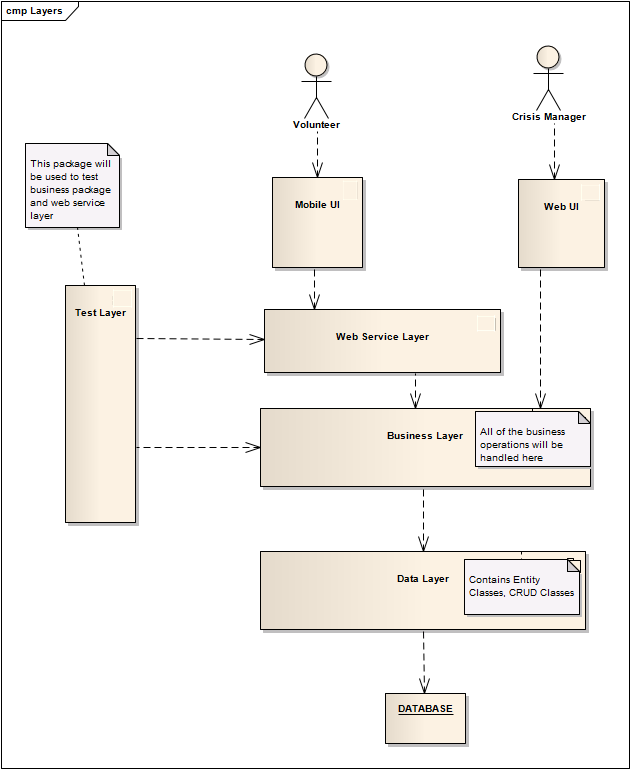


Figure . Logical View of VMS

More about the Logical view and detailed explanation of other views can be found in “SAD” document.

## Software Design Document

SDD document contains the detailed design based on the functional requirements, quality attributes and VMS architecture. It is guideline for developers who developed the VSM system. “SDD” document is located in “ArchitectureDesign” folder.

Structural design section of the SDD explains the layers, important components and classes inside the layers. Behavioural design explains the most important behaviours of the system and explains how these behaviours should be implemented by the developers. For explanation of some behaviours of the system sequence diagrams had been used. For others only textual explanation had been written.

## Manager’s manual

In order to provide the manager with the ability of using the features of the system in and efficient way, a Manager’s Manual is provided. The manager, who interacts with the system through web portal, will learn how to register and log in to the system by this manual. He also can find how to work with a crisis, with the incidents, with the requests for incidents, and how to search volunteers and send them alerts.

## Volunteer’s manual

Volunteers interact with VMS Mobile application and should have enough knowledge of how to work with that in an efficient way even though the application is made quite self-descriptive. For this purpose a volunteer’s manual has been prepared which describes the system functionalities and the way they can use them in a quick, easy way. By reading this manual they will learn how to log in to the system, report a new incident happened, receive alerts from manager, receive requests and accept/reject them, and report progress of the incident which they’re assigned to.

## Web Portal Installation Guide

The Web Portal Installation Guide explains the steps needed to deploy the web portal on the server. As deploying the web portal is not a very frequent action for each organization, it’s using technical vocabulary. The content of this guide shows how to deploy web application, database and connecting them.

# Conclusion

During the Technical Challenge course Apollo group worked on the VMS project. Based on the Industrial Challenge course outputs this project had been continued and finished with the worling prototype. The prototype was developed by the Apollo group. 84% of the functionality of the system was developed and tested by different testing strategies and techniques which are mentioned in the Test plan section.  Section Deliverables explains all the documents which have been produced during the Technical Challenge course by Apollo group.

The “SourceCode” folder within the deliverables contains two zip files. “VMSClient.zip”  file contains the source code of the VMS mobile application and “WebPortal.zip” file contains the source code for web part of the system.

1. <http://www.unv.org/en/current-highlight/focus-on-emergency-relief/doc/volunteers-vital-before-and.html> [↑](#footnote-ref-1)
2. D. Galin. Software Quality Assurance. Pearson, p.194, 2004 [↑](#footnote-ref-2)