Emergency Alert Mobile Application

Version <1.0>

* [Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]
* [To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 08/03/2017 | 1.0 | Initial Version | H.D.Witharana |
|  |  |  |  |

Note: When you include figures

* Use figure numbers and figure captions.
* Use diagrams/ images/ screen shots with high resolution to get a clear figure
* Use the figure captions in the form of Figure 1. <<caption>> and when explain it in the text, use the abbreviation “Figure 1,” even at the beginning of a sentence.
* When you use a tool to draw diagrams, change the font settings of the diagram; It is better to have font in black colour/ large in size (12pt) and if possible do not fill the objects/ elements in the diagram with a colour (keep the background colour white, for clear visibility)
* Also, try to re-locate the objects closely in a way that take less space (you may drag elements/objects close to each other).
* This will be useful to get a clear image; in order to make the diagram small in size without reducing the resolution quality.
* When you include images in your reports, make them “in line with text” ( picture tool bar 🡪 wrap text 🡪 in line with text) and include the caption accordingly. If you include two or more images together ( in a row), group them.
* Describe each diagram with few sentences.

When you draw diagrams (eg. Sequence diagram) do not include only two object called “user” and “system”. Include all the internal objects within the system, without considering the system as a black box. For example: for a mobile application the main system may consists of sub objects such as , <<UI>>:main\_Interface, :controller, <<UI>>:analysis\_Interface, :local\_DB, etc. (this is only an example; use your own terms).

References:

* Indicate the tools you have used to draw the diagrams

Useful theory for design diagrams : (relationships in Class diagram)

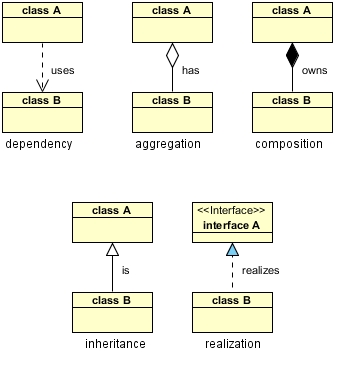


Table of Contents

1. Introduction 2

1.1 Purpose 2

1.2 Scope 2

1.3 Definitions, Acronyms, and Abbreviations 2

1.4 References 2

1.5 Overview 2

2. Architectural Representation 2

3. Architectural Goals and Constraints 2

4. Use-Case View 2

4.1 Use-Case Realizations 2

5. Logical View 2

5.1 Overview 2

5.2 Architecturally Significant Design Packages 2

6. Process View 2

7. Deployment View 2

8. Implementation View 2

8.1 Overview 2

8.2 Layers 2

9. Data View (optional) 2

10. Size and Performance 2

11. Quality 2

# Introduction

This software architecture document provides higher level overview of the architecture of Emergency Alert Mobile system and different views of the system design will be provided.

## Purpose

This document provides an architectural overview of the system. To identify different aspects of the system different architectural views are used. This explains the design views under various categories like use-case view, logical view, process view and etc. Further this document explains about architectural designs and constrains.

## Scope

This is a representation of the architecture of emergency alert system which will be used to design and implement the system. Developers of the system can clearly get an idea about the implementation of system by referring to the given designs. More over stakeholders and other parties interested on the system can get a clear idea about whole system and its’ functionalities.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Term | Description |
| Critical emergency | An emergency where help is needed within few seconds to prevent damages.  E.g. Murder, Robbery, Kidnapping |
| Non-critical emergency | An emergency where help is not needed within few seconds.  E.g. Car broke down |
| User | Person who register to application and use it in an emergency. |
| Nearby users | People who have been registered to the system and be near to an emergency and have on GPS. |
| Trusted contact | Contact that is pre-entered by a user to inform a critical emergency.  E.g. Mother |
| Helper | Person or organization registered in the system to provide services in a non-critical emergency situation. |

## References

* [This subsection provides a complete list of all documents referenced elsewhere in the **Software Architecture Document**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

- Indicate the tool you have used to draw the diagrams

## Overview

The rest of the software architecture document is designed as follows to give different perspectives to the emergency alerting system.

* Architectural representation

This section describes what software architecture is for the current system, and how it is represented. Of the **Use-Case**, **Logical**, **Process**, **Deployment**, and **Implementation Views**, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.

* Architectural goals and constraints

This section describes the software requirements and objectives that have some significant impact on the architecture; for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.

* Use-Case view

This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage- they exercise many architectural elements, or if they stress or illustrate a specific, delicate point of the architecture.

* Logical view

This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.

* Process view

This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.

* Deployment view

This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the **Process View** onto the physical nodes.

* Implementation

This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.

* Data view

A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.

* Size and performance

A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.

* Quality

A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.

# Architectural Representation

* [This section describes what software architecture is for the current system, and how it is represented. Of the **Use-Case**, **Logical**, **Process**, **Deployment**, and **Implementation Views**, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]

# Architectural Goals and Constraints

* [This section describes the software requirements and objectives that have some significant impact on the architecture; for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]

There are some software requirements and objectives that have some significant impact on the architecture of the system.

## Technical platform

# Use-Case View

Use-case view illustrate detail descriptions of functional requirements.

## Use-Case Realizations

### Use-Case diagramC:\Users\Hasini\AppData\Local\Microsoft\Windows\INetCache\Content.Word\HelpMe_Usecase (1).png

**Figure 1**

### Use-Case Detailed Descriptions

|  |  |  |
| --- | --- | --- |
| **Use case name** | Set trusted parties | |
| **Actor** | User | |
| **Description** | User needs to add at least 5 trusted contacts in order to inform an emergency. | |
| **Preconditions** | User should register to the system first. | |
| **Main flow** | **Actor** | **System** |
| 1.User enter name and mobile number of trusted contact | 1.1 Validated mobile number |
| 2.Set priority to the entered contact number |  |
| 3.Confirm the record | 3.1 Save data |
| **Successful end/post condition** | Save trusted contact. | |
| **Fail end/post condition** | 1.1 If mobile number is invalid send error message to user. | |
| **Extensions** | N/A | |

|  |  |  |
| --- | --- | --- |
| **Use case name** | Rate other users | |
| **Actor** | User | |
| **Description** | User can rate other users by their services given in an emergency. After an emergency, user will be given a form with informed users and user can give stars according to the help given or blacklist unwanted users. | |
| **Preconditions** | User should ask for help in an emergency. | |
| **Main flow** | **Actor** | **System** |
|  | 1.Send a feedback form after 24 hours of an emergency. |
| 2.Choose the users and rate them | 2.1 Save data |
| 3.Choose the users and block them. | 3.1 Blacklist contacts.  3.2 Ask the users to inform police about bad behavior of the contact. |
| **Successful end/post condition** | Rate users or blacklist users | |
| **Fail end/post condition** | 1. If user does not fill the feedback form display feedback form until it is filled by the user | |
| **Extensions** | N/A | |

|  |  |  |
| --- | --- | --- |
| **Use case name** | Inform trusted party | |
| **Actor** | User | |
| **Description** | User can press a button in a critical emergency to inform trusted party. Then the system will send information about the emergency to the trusted contacts according to the priority. | |
| **Preconditions** | User should register to the system first.  GPS should be on. | |
| **Main flow** | **Actor** | **System** |
| 1.User press the contact trusted party emergency button. | 1.1 Take the GPS location.  1.2 Take two pictures from front and back camera.  1.3 Select the most trusted contact from the priority list.  1.4 Call that contact.  1.5 Send the GPS location, two pictures and time to all trusted contacts. |
| 2.Inform the emergency. | 2.1 Save the emergency details. |
| **Successful end/post condition** | Inform trusted parties.  Save emergency details. | |
| **Fail end/post condition** | * 1. If GPS is not on inform it.   2. If camera is not working do not interrupt the process.   1.4 If the contact is busy call another trusted contact. | |
| **Extensions** | N/A | |

|  |  |  |
| --- | --- | --- |
| **Use case name** | Inform nearby users. | |
| **Actor** | User | |
| **Description** | User can press a button in a critical emergency to inform nearby users and then the system will send information about the emergency to the nearby contacts according to the ranks and the distance. | |
| **Preconditions** | User should register to the system first.  GPS should be on. | |
| **Main flow** | **Actor** | **System** |
| 1.User press the contact nearby users’ emergency button. | 1.1 Take the GPS location.  1.2 Take two pictures from front and back camera.  1.3 Select the closest 5 users with good ranking.  1.4 Send the GPS location, two pictures and time to those contacts. |
| 2.Inform the emergency. | 2.1 Save the emergency details. |
| **Successful end/post condition** | Inform nearby users.  Save emergency details. | |
| **Fail end/post condition** | * 1. If GPS is not on inform it.   2. If camera is not working do not interrupt the process. | |
| **Extensions** | N/A | |

|  |  |  |
| --- | --- | --- |
| **Use case name** | Get nearby police contacts. | |
| **Actor** | User | |
| **Description** | User can press a button in a critical emergency to get contacts of nearby police. This will provide the number of nearby police station and user can call if needed. | |
| **Preconditions** | User should register to the system first.  GPS should be on. | |
| **Main flow** | **Actor** | **System** |
| 1.User press the contact police button. | 1.1 Take the GPS location.  1.2 Search nearby police stations  1.3 Give details of the police station |
| 2.Inform the emergency to the police. | 2.1 Save the emergency details. |
| **Successful end/post condition** | Give nearby police details.  Save emergency details. | |
| **Fail end/post condition** | * 1. If GPS is not on inform it. | |
| **Extensions** | N/A | |

|  |  |  |
| --- | --- | --- |
| **Use case name** | Notify nearby dangerous places. | |
| **Actor** | System | |
| **Description** | When a user on GPS system will search for places where any emergency happened before and send the location to the users so users can avoid such places. | |
| **Preconditions** | User should register to the system first.  GPS should be on. | |
| **Main flow** | **Actor** | **System** |
| 1.Switch on GPS. | 1.1 Take the GPS location.  1.2 Search for nearby dangerous places.  1.3 Inform users. |
| 2.Confirm notification. |  |
| **Successful end/post condition** | Save emergency details. | |
| **Fail end/post condition** | * 1. If GPS is not on inform it. | |
| **Extensions** | N/A | |

|  |  |  |
| --- | --- | --- |
| **Use case name** | Register service | |
| **Actor** | Helper | |
| **Description** | User can register organization or help to use in an non critical emergency. | |
| **Preconditions** | Helper should register to the system first. | |
| **Main flow** | **Actor** | **System** |
| 1.Helper register the organization. | 1.1 Validate information  1.2 Search for category |
| 1. Add new category | 2.1 Save the category. |
| **Successful end/post condition** | Save emergency helps or organizations. | |
| **Fail end/post condition** | * 1. Send error message | |
| **Extensions** | Register users. | |

|  |  |  |
| --- | --- | --- |
| **Use case name** | Inform non-critical emergency | |
| **Actor** | User | |
| **Description** | User can search helpers according to the service they provide. | |
| **Preconditions** | User should register to system. | |
| **Main flow** | **Actor** | **System** |
| 1.User search help according to the given categories. | 1.1 Find nearby organizations  1.2 Give those contacts. |
| **Successful end/post condition** | Give organization contacts. | |
| **Fail end/post condition** | * 1. If no helpers inform. | |
| **Extensions** | N/A | |

# Logical View

* [This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

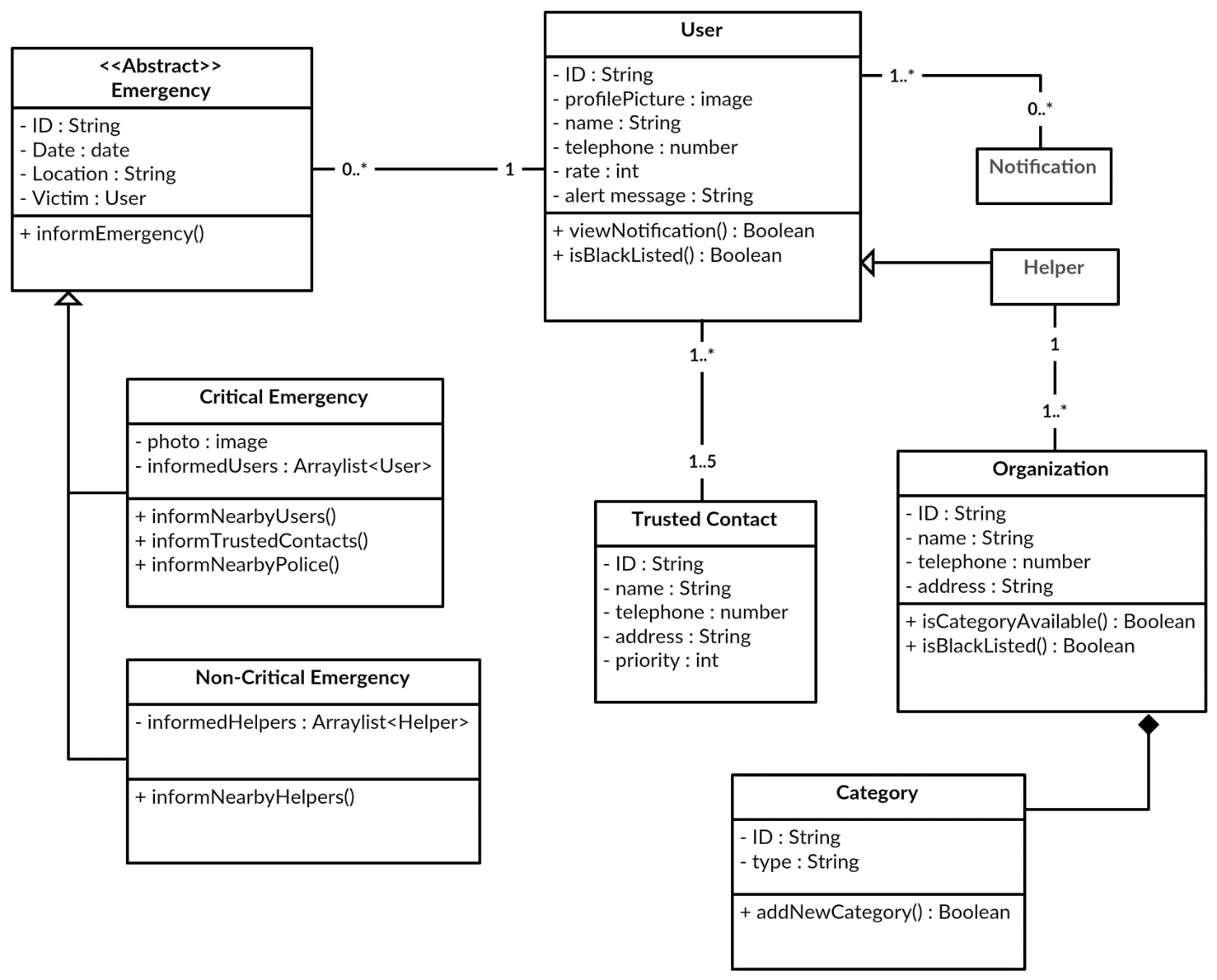
## Overview

* [This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]

## Architecturally Significant Design Packages

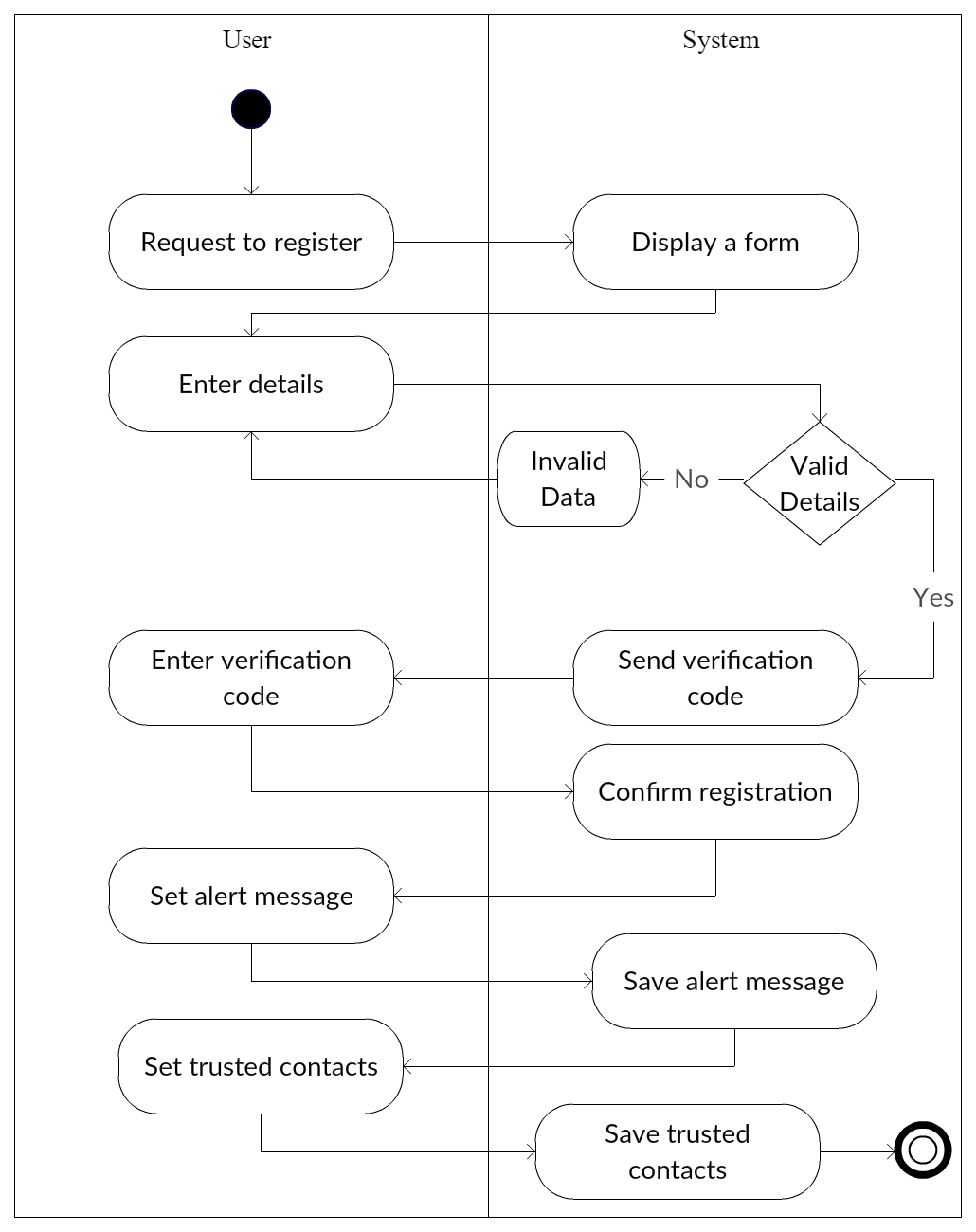
* [For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.
* For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

**Include the Class diagram and describe it**

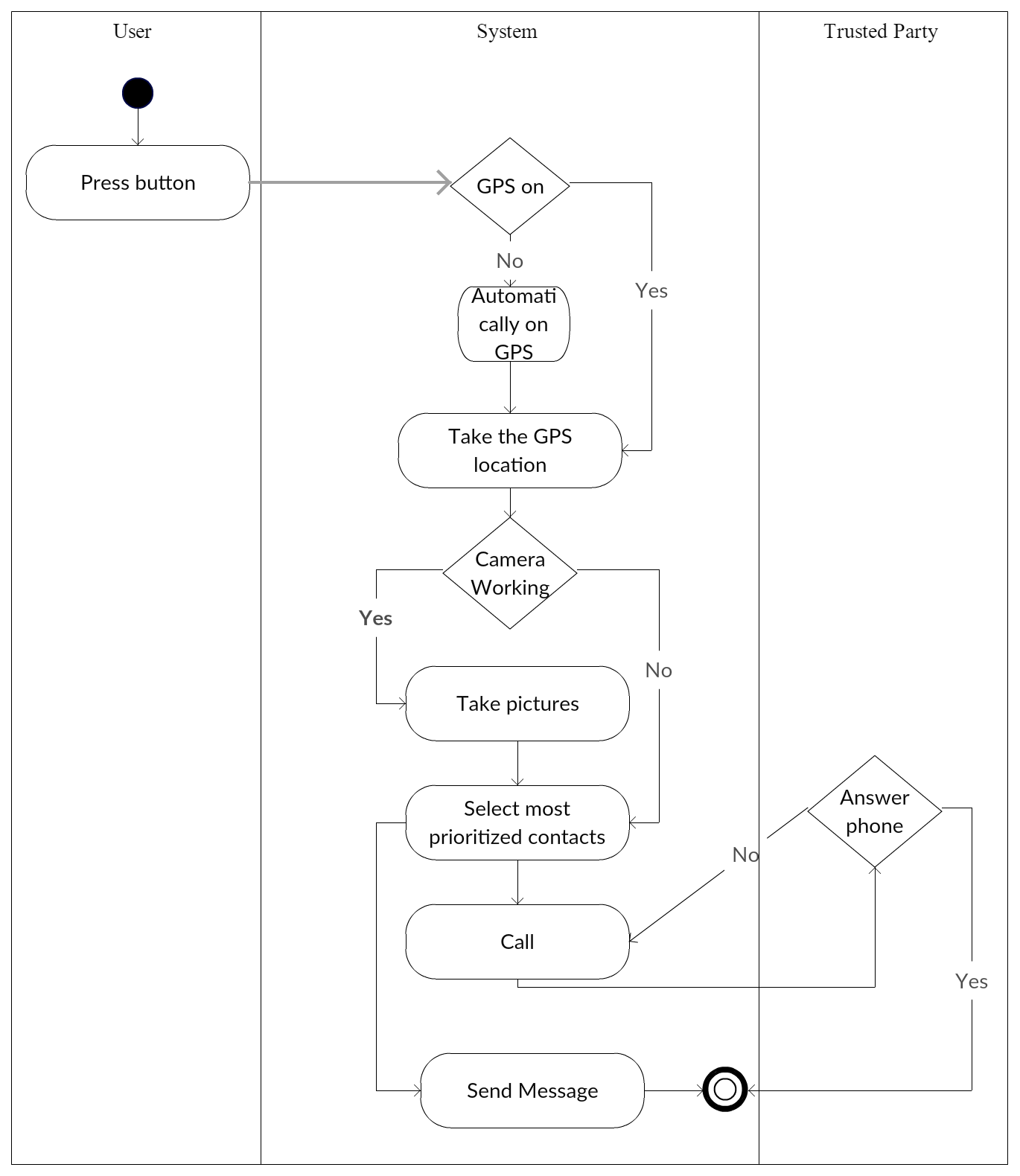


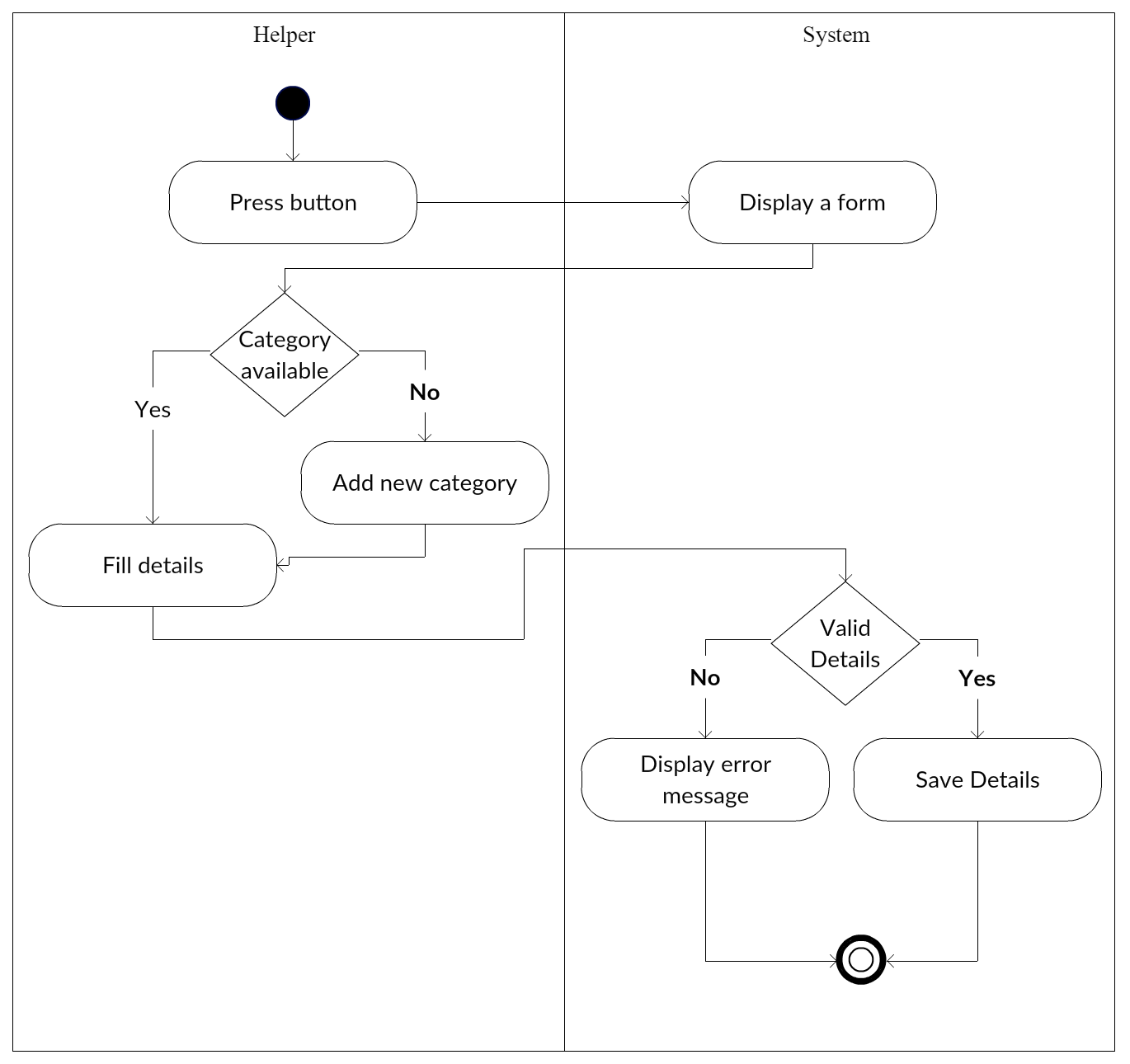
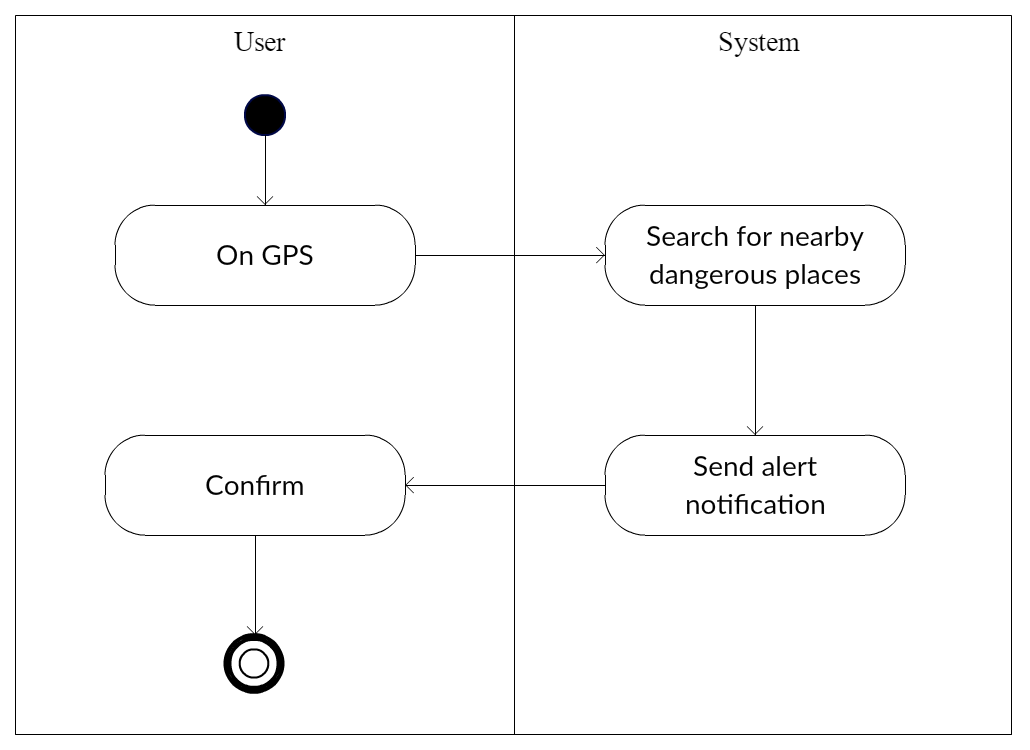
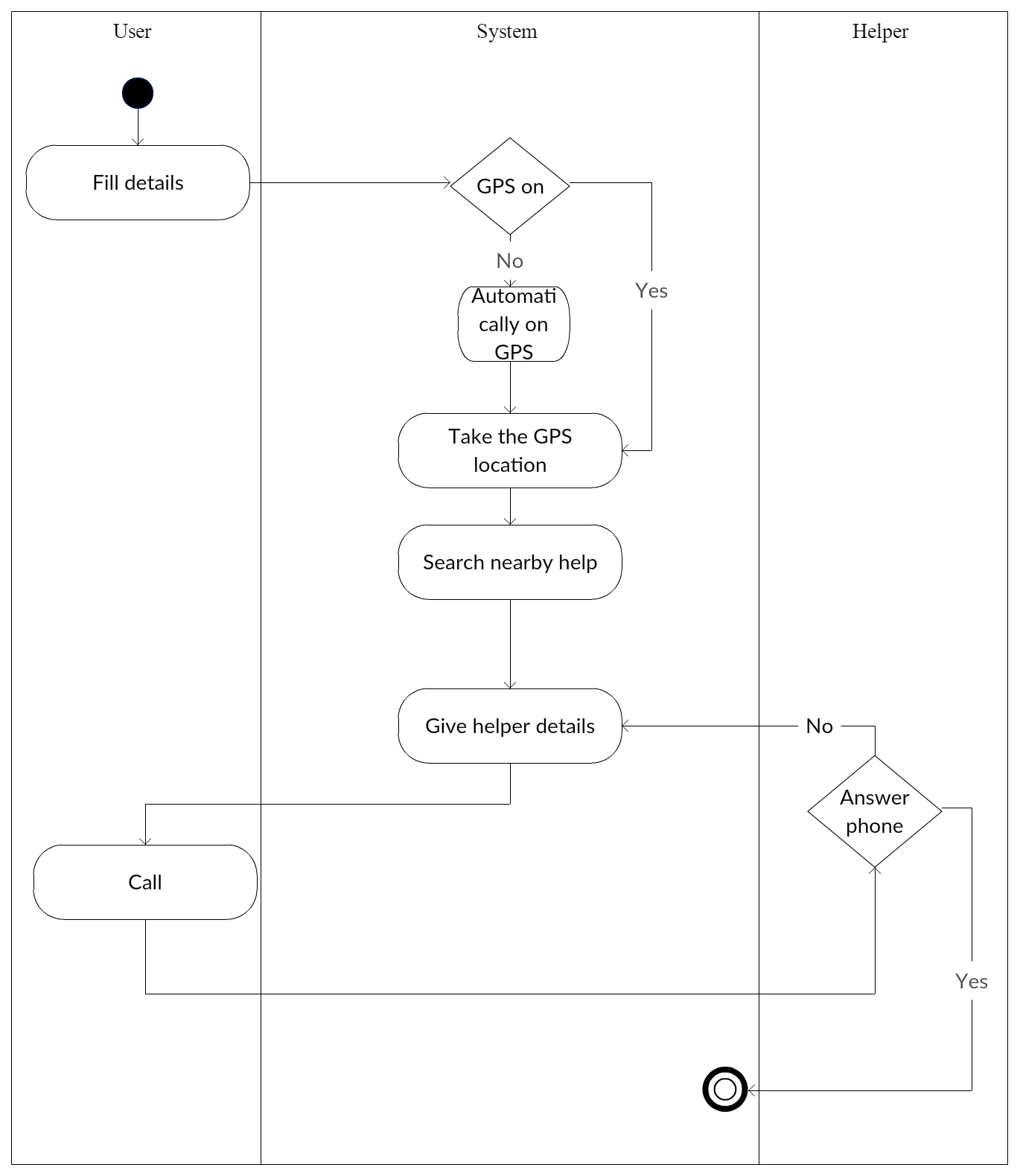
**Figure 3**

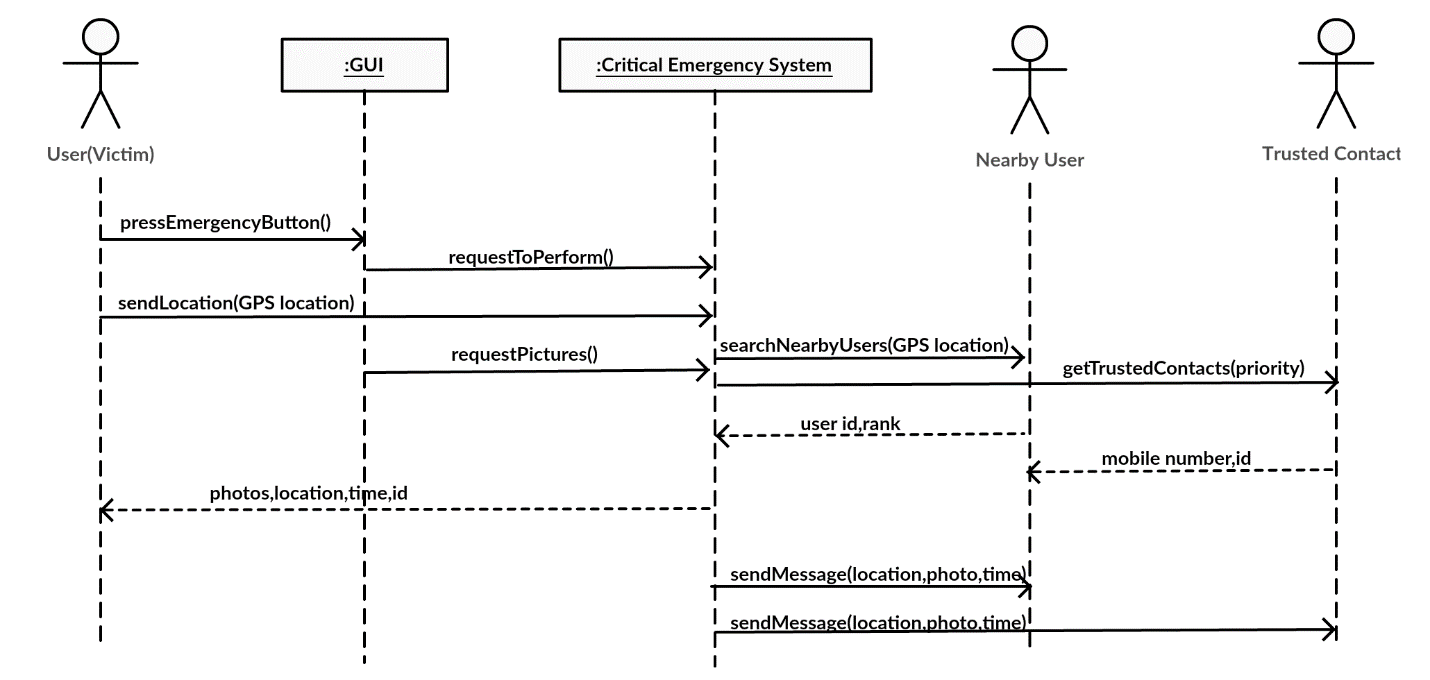
# Process View



**Include the Activity diagram and Sequence diagram and describe**







# Deployment View

* [This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the **Process View** onto the physical nodes.]

**Include the Deployment diagram if available and describe**

# Implementation View

* [This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]

## Overview

* [This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers. ]

## Layers

* [For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

**Include the Package diagram and describe**

# Data View (optional)

* [A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.]

# Size and Performance

* [A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

# Quality

* [A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]

**12. References**

Refer any data/ information in a standard format (eg. IEEE referencing style)

For different algorithms/ techniques/ theories you can refer text books.

For tools you can refer web pages.

For similar work you can refer research paper articles that describe the work.

You may include white paper articles for the description of technologies; web URL for the tool references. When you refer such a web page, you have to indicate the (Accessed on <<date>>)