SOFTWARE DESIGN DOCUMENT

**(SDD)**

**For**

**Flood Management System**

**Prepared for:**

**FACULTY SCIENCE COMPUTER AND SOFTWARE ENGINEERING**

**(FSKKP)**

**Generated By:**

The Insight TechAware Sdn Bhd

|  |  |
| --- | --- |
| **Name** | **ID** |
| Khairul Anwar Bin Halmin | CB14006 |
| Julian Johnny | CB14053 |
| Tengku Nazihah Binti Ungku Setea | CB14054 |
| Razanalia Erma Binti Razali | CB14055 |
| Nurul Azwani Binti Marwani | CB14104 |

**March 2016**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Approval List Table** | | | | | | |
| Index | 01 | 02 | 03 | 04 | 05 | 06 |
| Writed by:  Name: Razanalia Erma Binti Razali | Date: | Date: | Date: | Date: | Date: | Date: |
| Verified by:  Quality Manager  Name: Tengku Nazihah Binti Ungku Setea | Date: | Date: | Date: | Date: | Date: | Date: |
| Check by:  Configuration Manager  Name: Julian Johnny | Date | Date: | Date: | Date: | Date: | Date: |
| Approved by:  Work Package Manager  Name: Nurul Azwani Binti Marwani | Date | Date: | Date: | Date: | Date: | Date: |
| Authenticated by:  Project Manager  Name: Khairul Anwar Bin Halmin | Date | Date: | Date: | Date: | Date: | Date: |

|  |  |
| --- | --- |
| **Revision History** | |
| **Revision** | **Description** |
| 01 |  |
| 02 |  |
| 03 |  |
| 04 |  |
| 05 |  |
| 06 |  |

Contents

[1. SCOPE 1](#_Toc446484638)

[1.1. Identification 1](#_Toc446484639)

[1.2. Overview of the System 1](#_Toc446484640)

[1.3. Overview of the Document 2](#_Toc446484641)

[2. REFERENCED DOCUMENTS 3](#_Toc446484642)

[2.1. Overview of the Documents 3](#_Toc446484643)

[2.2. Reference Documents 3](#_Toc446484644)

[3. PRELIMINARY DESIGN 5](#_Toc446484645)

[3.1. System Overview 5](#_Toc446484646)

[3.1.1 System Architecture 7](#_Toc446484647)

[3.1.1.1 Static Organization 7](#_Toc446484648)

[3.1.2. System States and Modes 10](#_Toc446484649)

[3.2. CSCI Design Description 11](#_Toc446484650)

[3.2.1 Manage Centre 11](#_Toc446484651)

[3.2.2 Manage Victim 12](#_Toc446484652)

[3.2.3 Manage Volunteer 13](#_Toc446484653)

[4. DETAILED DESIGN 16](#_Toc446484654)

[4.1 Manage Centre (CMS01) 16](#_Toc446484655)

[4.1.1 Centre 17](#_Toc446484656)

[4.2 Manage Victim (CMS02) 34](#_Toc446484657)

[Figure 15 shows Manage Victim Module Subsystem 34](#_Toc446484658)

[4.2.1 Manage Victim Package 34](#_Toc446484659)

[4.3 Manage Volunteer Module (CMS03) 44](#_Toc446484660)

[Figure 4.3 shows Manage Volunteer Module subsystem. 44](#_Toc446484661)

[4.3.1 Manage Volunteer Package 44](#_Toc446484662)

[4.4 FMS-Inventory (CMS04) 58](#_Toc446484663)

[4.4.1 Inventory Module Package 58](#_Toc446484664)

[4.5 Generate Report Module (CMS05) 67](#_Toc446484665)

[Figure 4.5 shows Generate Report Module subsystem. 67](#_Toc446484666)

[4.5.1 Generate Report Package 67](#_Toc446484667)

[5. DATA DICTIONARY 74](#_Toc446484668)

[6. NOTES 78](#_Toc446484669)

# SCOPE

## Identification

System name: Flood Management System

Abbreviation: FMS

System ID No.: FMS-2016-01

## Overview of the System

The aim FMS is to introduce an efficient system that is manageable and user friendly system to our client. This system specifically design for *Jabatan Kebajikan Masyrakat* (JKM) to manage all the things when flood occur.

* + 1. **Problem Statement**

Flooding is the most significant natural hazard in Malaysia in terms of populations affected, frequency, area extent, flood duration and social economic damage. Therefore, The Insight TechAware Sdn Bhd has develop a web based system named Flood Management System (FMS).

* + 1. **Purpose of The System**

The purpose of the project is to analyze the requirements of design, implement, and maintain the system according to the requirements specified by the client and also manage the data systematically. Flood operation requires close cooperation and understanding among various parties involved at the flood plan to be efficient and successful in rescuing the victims and reducing number of property losses, including at district level coordination. This system involve victims, volunteers, sponsors and external users.

## Overview of the Document

This paragraph summarizes the purpose and contents of this Software Design Document (SDD). It specifies the requirement aspects related to the Flood Management System.

In general, this SDD is divided into 8 sections as the following:

|  |  |
| --- | --- |
| **Chapter 1** | Describes the scope identification, system overview and the document overview. |
| **Chapter 2** | Referenced documents, contractual documents and non-contractual documents. |
| **Chapter 3** | Describes the preliminary design for the CSCI. The preliminary design will identify CSC, the description of each CSC design, the characteristics of each CSC and the traceability of requirements set forth in SRS and IRS. |
| **Chapter 4** | Describes the detail design. |
| **Chapter 5** | Describes the global data elements within the CSCI. |
| **Chapter 6** | Describes each of the shared data files of the CSCI. |
| **Chapter 7** | Requirements traceability to SRS and IRS documents. |
| **Chapter 8** | Notes and abbreviations. |

# 2. REFERENCED DOCUMENTS

This section lists the document number and document name referenced in this document. Any discrepancies of this document in describing the software development process should be covered by the documents listed in this chapter. The following documents were referred as the basis for this SDD preparation.

Copies of specification, standards, drawings and publication requested by suppliers in contact with the specified supplying functions may be obtained by contacting the agency or directly through the contracting office.

## 2.1. Overview of the Documents

Below is the standard that has been used in this document.

1. IEEE Std 1016-1998, “IEEE Recommended Practice for Software Design Descriptioms” , 1998 Edition, IEEE, 1998

<http://standards.ieee.org/findstds/standard/1016-1998.html>

|  |
| --- |
|  |
|  |

## 2.2. Reference Documents

This section consists :

2.2.1 Contractual documents

Not applicable.

2.2.2 Non-contractual documents

1. **Software Requirements Specification (SRS) for Flood Management System (FMS)**

Version : 1.0

Date : Disember 2015

Author : The Insight TechAware Sdn Bhd

Publisher : -

1. **Software Development Plan (SDP) for Flood Management System (FMS)**

Version : 2.0

Date : November 2015

Author : The Insight TechAware Sdn Bhd

Publisher : -

1. **Software Design:From Programming To Architecture**

Version : -

Date : 2004

Author : Eric Braude

Publisher : John Wiley & Sons, Inc.

# 3. PRELIMINARY DESIGN

## 3.1. System Overview

Figure 3.1 shows the relationship between external interfaces and Flood Management System (FMS).

|  |
| --- |
|  |

Figure 3.1: System Design Overview

This project consists of five main modules which are the center registration, manage victim, manage volunteer, manage item and report.

The registration center module will register the center. Center will allow the flood victims and volunteer to check in and check out in the center. The information of the flood victims and volunteer will be recorded. The status of the center will be informed to the public.

In manage victim module, once they have check in, victim manager will place them in the center. The staff center will provide the necessities such as water, food for them.

In manage volunteer module, once they have check in, they can choose types of volunteer they would like to do based on the categories provided such as accommodation, equipment, food and water, transport, labor specialist. They can also keep track their activities from time to time.

In manage item or inventory module, it is used to store items that will be distributed to the victims at the evacuation center. It consists of stock in, stock out and disposal items. The status of the inventory will be recorded in report.

In news and report, administrator will retrieve the report about center, number of victim being evacuate and item has been distribute to victims. It also will be used by the user to get the information about the center and number of victims being evacuate.

## 3.1.1 System Architecture

This paragraph identifies the internal organizational structure of the system. The relationship among system subsystem will be described.

## 3.1.1.1 Static Organization

|  |
| --- |
|  |

Figure 3.1.1.1: Static Organization of FMS

Figure above shows that FMS consists of five packages. This section describes the detail for each subsystem/package .

* 1. **CentreRegistration**

This package responsible to manage the registration of centre. This package consists of the following classes :

* + 1. Centre
    2. Address
    3. PicDetail
  1. **ManageVictim**

The role of this subsystem is to manage victim. This package consists of the following classes or unit :

* + 1. Victim
    2. MedicalInfo
    3. CheckIn
    4. CheckOut
  1. **ManageVolunteer**

The role of this subsystem is to manage volunteer. This package consists of the following classes or unit :

* + 1. Volunteer
    2. Distribution
    3. Address
    4. CheckOut
  1. **ManageInventory**

The role of this subsystem is to manage inventory information. This package consists of the following classes or unit :

* + 1. Items
    2. Food
    3. Clothes
    4. Others
  1. **ManageReport**

The role of this subsystem is to manage report of the system. This package consists of the following classes or unit :

* + 1. Report
    2. Daily
    3. Weekly
    4. Monthly

**3.1.1.2. Dynamic Organization**

Figure 3.1.1.2 diagram shows components and their relationships between each other in system.

|  |
| --- |
| C:\Users\ka_style\Desktop\Workshop\component.JPG |

Figure 2.1.1.2: Component Diagram of FMS

**3.1.1.3. CSCs Interfaces**

Figure 3.1.1.3 shows all the interaction between package or subsystem and non-development external subsystem/unit.

|  |
| --- |
|  |

Figure 3.1.1.3: Package/Subsystem Interfaces

#### 3.1.1.4. External Interfaces

Not applicable.

### 3.1.2. System States and Modes

Figure 3.1.2 describes states diagrams of FMS.

|  |
| --- |
|  |

Figure 3.1.2: State Diagram for your system

## 3.2. CSCI Design Description

FMS consists of 5 subsystem which are manage centre, manage victim, manage volunteer, manage inventory and generate report.

### 3.2.1 Manage Centre

|  |
| --- |
|  |

Figure 3.2.1 : Visibility of Manage Centre

### 

### 3.2.2 Manage Victim

|  |
| --- |
| C:\Users\ka_style\Desktop\julian.JPG |

Figure 3.2.2 : Visibility of Manage Victim

### 3.2.3 Manage Volunteer

|  |
| --- |
|  |

Figure 3.2.3 : Visibility of Manage Volunteer

**3.2.4 Manage Inventory**

|  |
| --- |
|  |

Figure 3.2.4 : Visibility of Manage Volunteer

**3.2.5 Generate Report**

|  |
| --- |
| **report class diagram** |

Figure 3.2.5 : Visibility of Generate Report

# 

# 4. DETAILED DESIGN

This section divided into the following paragraphs and subparagraphs to describe the detailed design.

## 4.1 Manage Centre (CMS01)

Figure 4.1 show Manage Centre Module Subsystem

|  |
| --- |
|  |

Figure 4.1: Manage Centre Module Detail Design

### 4.1.1 Centre

The purpose of this class is to allow admin to register,update and delete centre information

This package consists of:

1. Centre
2. Address
3. PicDetail
4. CentreController
5. RegisterCentreView
6. UpdateCentreView
7. Main

#### 4.1.1.1 Class Centre Design

This subparagraph specifies the design of centre

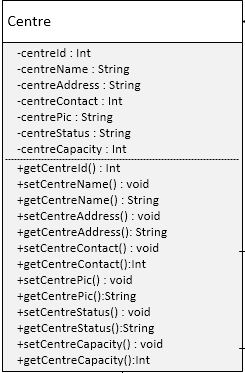


Figure 4.1.1 Centre Class

1. **Input/Output data elements**

List of input and output data elements:

Input : centreId,centreName,centreAddress,centreContact,centrePic,

centreStatus,centreCapacity

Output : Display all the detail of centre

1. **Local data elements**

|  |  |
| --- | --- |
| Name | centreId |
| Description | This variable hold the centre’s id |
| Data Type | Integer |
| Precision/resolution | Number |

Table 4.1.1: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | centreName |
| Description | This variable hold centre’s name |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.1.2: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | centreAddress |
| Description | This variable hold centre’s address |
| Data Type | - |
| Precision/resolution | - |

Table 4.1.3: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | centreContact |
| Description | This variable hold centre’s contact |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.1.4: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | centrePic |
| Description | This variable hold centre’s pic |
| Data Type | - |
| Precision/resolution | - |

Table 4.1.5: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | centreStatus |
| Description | This variable hold centre’status |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.1.6: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | centreCapacity |
| Description | This variable hold centre’s capacity |
| Data Type | integer |
| Precision/resolution | Number |

Table 4.1.7: Local Data Definition for data elements

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this Slass

Class Type : Model Class

Responsibility : Holds the details of centre information

Attributes : id,name, address, contact, pic,status, capacity

Type: centreId, centreName,centreAddress,centreContact,

centrePic,centreStatus,centreCapacity

**Methods:**getCentreId,setName,getName,setAddress,getAddress,setContact,getContact,setPic,getPic, setStatus, getStatus, setCapacity, getCapacity

**1. *getCentreId***

**Responsibility : get id for centre information**

**Input Parameter :** None

**Output Parameter :** None

**Algorithm :**

BEGIN

GET centreId

END

**1. *setName***

**Responsibility : set name centre information**

**Input Parameter :** txtName

**Output Parameter :** None

**Algorithm :**

BEGIN

setName

END

**2.*setAddress***

**Responsibility : set address centre information**

**Input Parameter :** txtAddress

**Output Parameter :** None

**Algorithm :**

BEGIN

setAddress

END

**3.*setContact***

**Responsibility : set contact centre information**

**Input Parameter :** txtContact

**Output Parameter :** None

**Algorithm :**

BEGIN

setContact

END

**4.*setPic***

**Responsibility : set pic centre information**

**Input Parameter :** txtPic

**Output Parameter :** None

**Algorithm :**

BEGIN

setPic

END

***5. setCentreStatus***

**Responsibility : set status centre information**

**Input Parameter :** txtCentreStatus

**Output Parameter :** None

**Algorithm :**

BEGIN

setCentreStatus

END

***6. setCentreCapacity***

**Responsibility : set capacity centre information**

**Input Parameter :** txtCentreCapacity

**Output Parameter :** None

**Algorithm :**

BEGIN

setCentreCapacity

END

**7. *getName***

**Responsibility : get name centre information**

**Input Parameter :** txtName

**Output Parameter :** None

**Algorithm :**

BEGIN

GET picName

END

**8.*getAddress***

**Responsibility : get address centre information**

**Input Parameter :** txtAddress

**Output Parameter :** None

**Algorithm :**

BEGIN

GET centreAddress

END

**9.*getContact***

**Responsibility : get contact centre information**

**Input Parameter :** txtContact

**Output Parameter :** None

**Algorithm :**

BEGIN

GET centreContact

END

**10.*getPic***

**Responsibility : get pic centre information**

**Input Parameter :** txtPic

**Output Parameter :** None

**Algorithm :**

BEGIN

GET centrePic

END

***11. getCentreStatus***

**Responsibility : get status centre information**

**Input Parameter :** txtCentreStatus

**Output Parameter :** None

**Algorithm :**

BEGIN

GET centreStatus

END

***12. getCentreCapacity***

**Responsibility : get capacity centre information**

**Input Parameter :** txtCentreCapacity

**Output Parameter :** None

**Algorithm :**

BEGIN

GET centreCapacity

END

#### 4.1.1.2 Class Address Design

This subparagraph specifies the design of address

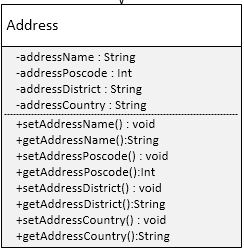


Figure 4.1.2 Address Class

1. **Input/Output data elements**

List of input and output data elements:

Input : addressName, addressPoscode, addressDistrict,

addressCountry

Output : Display all the detail of address

1. **Local data elements**

|  |  |
| --- | --- |
| Name | addressName |
| Description | Contain name for address detail. |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.1.8: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | addressPoscode |
| Description | Contain poscode for address detail. |
| Data Type | Integer |
| Precision/resolution | Number |

Table 4.1.9: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | addressDistrict |
| Description | Contain district for address detail. |
| Data Type | String |
| Precision/resolution | Alpabet |

Table 4.1.10: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | addressCountry |
| Description | Contain poscode for address detail. |
| Data Type | String |
| Precision/resolution | Alpabet |

Table 4.1.11: Local Data Definition for data elements

c) **Algorithms**

This section states the purpose and describes in detail the algorithms of this Slass

Class Type : Model Class

Responsibility : Holds the details of address information

Attributes : name, poscode, instinct, country

Type : addressName, addressPoscode, addressDistrict, addressCountry

**Methods:**setAddressName,setAddressPoscode,setAddressInstict, setAddressCountry,getAddressName,getAddressPoscode,getAddressInstict,

getAddressCountry

**1. *setAddressName***

**Responsibility : set name for address details**

**Input Parameter :** txtAddressName

**Output Parameter :** None

**Algorithm :**

BEGIN

setAddressName

END

**2. *setAddressPoscode***

**Responsibility : set poscode for address information**

**Input Parameter :** txtAddressPoscode

**Output Parameter :** None

**Algorithm :**

BEGIN

setAddressPoscode

END

**3. *setAddressDistrict***

**Responsibility : set district for address information**

**Input Parameter :** txtAddressDistrict

**Output Parameter :** None

**Algorithm :**

BEGIN

setAddressDistrict

END

**4. *setAddressCountry***

**Responsibility : set country for address information**

**Input Parameter :** txtAddressCountry

**Output Parameter :** None

**Algorithm :**

BEGIN

setAddressCountry

END

**5. *getAddressName***

**Responsibility : get name for address information**

**Input Parameter :** txtAddressCountry

**Output Parameter :** None

**Algorithm :**

BEGIN

GET AddressName

END

**6. *getAddressPoscode***

**Responsibility : get poscode for address information**

**Input Parameter :** txtAddressPoscode

**Output Parameter :** None

**Algorithm :**

BEGIN

GET AddressPoscode

END

**7. *getAddresDistrict***

**Responsibility : get district for address information**

**Input Parameter :** txtAddressDistrict

**Output Parameter :** None

**Algorithm :**

BEGIN

GET addressDistrict

END

**8. *getAddresCountry***

**Responsibility : get country for address information**

**Input Parameter :** txtAddressCountry

**Output Parameter :** None

**Algorithm :**

BEGIN

GET addressCountry

END

#### 4.1.1.3 Class PicDetail Design

This subparagraph specifies the design of PicDetail

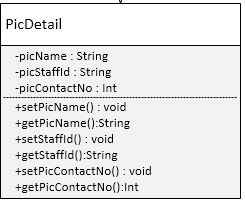


Figure 4.1.3 Centre Class

1. **Input/Output data elements**

List of input and output data elements:

Input : picName, picStaffId,picContactNo

Output : Display all the detail of person in charge (PIC)

1. **Local data elements**

|  |  |
| --- | --- |
| Name | picName |
| Description | Contain name for pic detail. |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.1.12: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | picStaffId |
| Description | Contain staff id for pic detail. |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.1.13: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | picContactNo |
| Description | Contain contact number for pic detail. |
| Data Type | Integer |
| Precision/resolution | Number |

Table 4.1.14: Local Data Definition for data elements

c) **Algorithms**

This section states the purpose and describes in detail the algorithms of this Class

Class Type : Model Class

Responsibility : Holds the details of person in charge information

Attributes : name, staffId,contactNo

Type : picName,picStaffId,picContactNo

**Methods:** setPicName, setPicStaffId, setPicContactNo, getPicName, getPicStaffId,getPicContactNo

**1. *setPicName***

**Responsibility : set name for person in charge information details**

**Input Parameter :** txtPicName

**Output Parameter :** None

**Algorithm :**

BEGIN

setPicName

END

**2. *setPicStaffId***

**Responsibility : set staff id for person in charge information details**

**Input Parameter :** txtPicStaffId

**Output Parameter :** None

**Algorithm :**

BEGIN

setPicStaffId

END

**3. *setPicContactNo***

**Responsibility : set contactNo for person in charge information details**

**Input Parameter :** txtPicContactNo

**Output Parameter :** None

**Algorithm :**

BEGIN

setPicContactNo

END

**4. *getPicName***

**Responsibility : get name for person in charge information details**

**Input Parameter :** txtPicName

**Output Parameter :** None

**Algorithm :**

BEGIN

GET picName

END

**5. *getPicStaffId***

**Responsibility : get staffId for person in charge information details**

**Input Parameter :** txtPicStaffId

**Output Parameter :** None

**Algorithm :**

BEGIN

GET picStaffId

END

**6. *getPicContactNo***

**Responsibility : get contact number for person in charge information details**

**Input Parameter :** txtPicContactNo

**Output Parameter :** None

**Algorithm :**

BEGIN

GET picContactNo

END

## 4.2 Manage Victim (CMS02)

## Figure 15 shows Manage Victim Module Subsystem

|  |
| --- |
| **C:\Users\ka_style\Desktop\julian.JPG** |

Figure 3: Manage Victim Detail Design

### 4.2.1 [Manage Victim Package]

The purpose of this package is to show how victim is managed in the Flood Management System which consists of:

(a)4.2.1.1 Victim

(b)4.2.2.1 medicalInfo

(c)4.2.3.1 checkIn

(d)4.2.4.1 checkOut

#### 4.2.1.1 Class [Victim] Design

This subparagraph specifies the design of [Victim] which shows how victim’s credential is inserted into the Flood Management System.

1. **Input/Output data elements**

List of input and output data elements:

Input : [name

ic

age

gender

address

phoneno

id]

Output : [null]

1. **Local data elements**

|  |  |
| --- | --- |
| Name | name |
| Description | Name of flood victim |
| Data Type | String |
| Precision/resolution | Alphabets |

Table 1: Local Data Definition for data elements victimName

|  |  |
| --- | --- |
| Name | ic |
| Description | Victim’s identity card number |
| Data Type | Integer |
| Precision/resolution | Numbers |

Table 2: Local Data Definition for data elements victimIC

|  |  |
| --- | --- |
| Name | age |
| Description | Victim’s age |
| Data Type | Integer |
| Precision/resolution | Numbers |

Table 3: Local Data Definition for data elements victimAge

|  |  |
| --- | --- |
| Name | gender |
| Description | Victim’s gender |
| Data Type | String |
| Precision/resolution | Alphabets |

Table 4: Local Data Definition for data elements victimGender

|  |  |
| --- | --- |
| Name | address |
| Description | Victim’s address |
| Data Type | String |
| Precision/resolution | Alphabets |

Table 5: Local Data Definition for data elements victimAddress

|  |  |
| --- | --- |
| Name | phoneno |
| Description | Victim’s phone number |
| Data Type | Integer |
| Precision/resolution | Numbers |

Table 6: Local Data Definition for data elements victimPhoneno

|  |  |
| --- | --- |
| Name | id |
| Description | Victim’s id |
| Data Type | Integer |
| Precision/resolution | Numbers |

Table 7: Local Data Definition for data elements victimID

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class.

Class Type : Victim

Responsibility : Victim is the main class for Manage Victim subsystem of the Flood Management System. It serves the function of adding victim’s credential into the system.

Attributes : name:String

ic:Integer

age:Integer

gender:String

address:String

phoneno:Integer

id:Integer

Methods :

**4.2.1.1.1 setName**

Responsibility: To add victim’s name into the system

Input Parameters: name

Output Parameters: null

Pre Conditions: name is instantiated inside Victim class

Algorithm:

BEGIN

setName(String victimName){

this.name=victimName;

}

END

Post Conditions: input name is inserted

**4.2.1.1.2 getName**

Responsibility: To add victim’s name into the system

Input Parameters: name

Output Parameters: null

Pre Conditions: name is instantiated inside Victim class

Algorithm:

BEGIN

getName(){

return name;

}

END

Post Conditions: input name is inserted

**4.2.1.1.3 setIC**

Responsibility: To add victim’s IC number to the system

Input Parameters: ic

Output Parameters: null

Pre Conditions: ic is instantiated inside Victim class

Algorithm:

BEGIN

setIC(int victimIC){

this.ic=victimIC;

}

END

Post Conditions: input ic is inserted

**4.2.1.1.4 getIC**

Responsibility: To add victim’s IC number to the system

Input Parameters: ic

Output Parameters: null

Pre Conditions: ic is instantiated inside Victim class

Algorithm:

BEGIN

getIC(){

return ic;

}

END

Post Conditions: input ic is inserted

**4.2.1.1.5 setAge**

Responsibility: To add victim’s age to the system

Input Parameters: age

Output Parameters: null

Pre Conditions: age is instantiated inside Victim class

Algorithm:

BEGIN

setAge(int victimAge){

this.age=victimAge;

}

END

Post Conditions: input age is inserted

**4.2.1.1.6 getAge**

Responsibility: To add victim’s age to the system

Input Parameters: age

Output Parameters: null

Pre Conditions: age is instantiated inside Victim class

Algorithm:

BEGIN

getAge(){

return age;

}

END

Post Conditions: input age is inserted

**4.2.1.1.7 setGender**

Responsibility: To add victim’s gender to the system

Input Parameters: gender

Output Parameters: null

Pre Conditions: gender is instantiated inside Victim class

Algorithm:

BEGIN

setGender(String victimGender){

this.gender=victimGender;

}

END

Post Conditions: input gender is inserted

**4.2.1.1.8 getGender**

Responsibility: To add victim’s gender to the system

Input Parameters: gender

Output Parameters: null

Pre Conditions: gender is instantiated inside Victim class

Algorithm:

BEGIN

getGender(){

return gender;

}

END

Post Conditions: input gender is inserted

**4.2.1.1.9 setAddress**

Responsibility: To add victim’s address to the system

Input Parameters: address

Output Parameters: null

Pre Conditions: address is instantiated inside Victim class

Algorithm:

BEGIN

setAddress(String victimAddress){

this.address=victimAddress;

}

END

Post Conditions: input address is inserted

**4.2.1.1.10 getAddress**

Responsibility: To add victim’s address to the system

Input Parameters: address

Output Parameters: null

Pre Conditions: address is instantiated inside Victim class

Algorithm:

BEGIN

getAddress(){

return address;

}

END

Post Conditions: input address is inserted

**4.2.1.1.11 setPhoneno**

Responsibility: To add victim’s phone number to the system

Input Parameters: phoneno

Output Parameters: null

Pre Conditions: phoneno is instantiated inside Victim class

Algorithm:

BEGIN

setPhoneno(int victimPhoneno){

this.phoneno=victimPhoneno;

}

END

Post Conditions: input phoneno is inserted

**4.2.1.1.12 getPhoneno**

Responsibility: To add victim’s phone number to the system

Input Parameters: phoneno

Output Parameters: null

Pre Conditions: phoneno is instantiated inside Victim class

Algorithm:

BEGIN

getPhoneno(){

return phoneno;

}

END

Post Conditions: input phoneno is inserted

**4.2.1.1.13 setID**

Responsibility: To generate victim’s in the system

Input Parameters: id

Output Parameters: null

Pre Conditions: id is instantiated inside Victim class

Algorithm:

BEGIN

setID (int victimID){

this.id=victimID;

}

END

Post Conditions: id is generated

**4.2.1.1.14 getID**

Responsibility: To generate victim’s in the system

Input Parameters: id

Output Parameters: null

Pre Conditions: id is instantiated inside Victim class

Algorithm:

BEGIN

getID(){

return id;

}

END

Post Conditions: id is generated

#### 4.2.2.1 Class [medicalInfo] Design

This subparagraph specifies the design of [medicalInfo] which shows how victim’s disease type is inserted into the Flood Management System.

1. **Input/Output data elements**

List of input and output data elements:

Input : [diseaseType]

Output : [null]

1. **Local data elements**

|  |  |
| --- | --- |
| Name | diseaseType |
| Description | Name of disease of flood victim |
| Data Type | String |
| Precision/resolution | Alphabets |

Table 1: Local Data Definition for data elements diseaseType

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class.

Class Type : medicalInfo

Responsibility : medicalInfo is a class in Manage Victim subsystem of the Flood Management System. It serves the function of adding victim’s diseasae type into the system.

Attributes : diseaseType:String

Methods :

**4.2.2.1.1 setMedicalinfo**

Responsibility: To add victim’s disease type into the system

Input Parameters: diseaseType

Output Parameters: null

Pre Conditions: diseaseType is instantiated inside Victim class

Algorithm:

BEGIN

setMedicalinfo(String diseaseType){

this.diseaseType=diseaseType;

}

END

Post Conditions: input diseaseType is inserted

**4.2.2.1.2 getMedicalinfo**

Responsibility: To add victim’s disease type into the system

Input Parameters: diseaseType

Output Parameters: null

Pre Conditions: diseaseType is instantiated inside Victim class

Algorithm:

BEGIN

getMedicalinfo(){

return diseaseType;

}

END

Post Conditions: input diseaseType is inserted

#### 4.2.3.1 Class [checkIn] Design

This subparagraph specifies the design of [checkIn] which shows how flood victims are registered into the Flood Management System.

1. **Input/Output data elements**

List of input and output data elements:

Input : [null]

Output : [null]

1. **Local data elements**

|  |  |
| --- | --- |
| Name | - |
| Description | - |
| Data Type | - |
| Precision/resolution | - |

Table 1: Local Data Definition for data elements in checkIn

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class.

Class Type : checkIn

Responsibility : checkIn is a class in Manage Victim subsystem of the Flood Management System. It serves the function of registering flood victim into the system.

Attributes : null

Methods :

**4.2.3.1.1 addVictim**

Responsibility: To register flood victim into the system

Input Parameters: victimIC

Output Parameters: null

Pre Conditions: Victim class is called

Algorithm:

BEGIN

Get victimIC;

IF(victimIC=null)

Victim victim[i];

while(i<centreCapacity)

{

victim[i]=new Victim();

victim[i].setName(victimName);

victim[i].setIC (victimIC);

victim[i].setAge(victimAge);

victim[i].setGender(victimGender);

victim[i].setAddress(victimAddress);

victim[i].setPhoneno(victimPhoneno);

victim[i].setID(victimID);

}

ELSE

{system.out.println(“Victim is already in the system!”);}

END

Post Conditions: Flood victim is registered into the system

#### 4.2.4.1 Class [checkOut] Design

This subparagraph specifies the design of [checkOut] which shows how flood victims are dropped out of the Flood Management System.

1. **Input/Output data elements**

List of input and output data elements:

Input : [null]

Output : [null]

1. **Local data elements**

|  |  |
| --- | --- |
| Name | - |
| Description | - |
| Data Type | - |
| Precision/resolution | - |

Table 1: Local Data Definition for data elements in checkOut

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class.

Class Type : checkOut

Responsibility : checkOut is a class in Manage Victim subsystem of the Flood Management System. It serves the function of dropping out flood victim of the system.

Attributes : null

Methods :

**4.2.4.1.1 dropVictim**

Responsibility: To drop flood victim out of the system

Input Parameters: victimIC

Output Parameters: null

Pre Conditions: Victim class is called

Algorithm:

BEGIN

Get victimIC;

IF(victimIC=null)

{system.out.println(“Victim is not in the system!”);}

ELSE

Get victimName;

{system.out.println(“Check out of victim ”+victimName+” with

Identity card number ”+victimIC+” has checked out successfully”);}

END

Post Conditions: Flood victim is dropped out of the system

## 4.3 Manage Volunteer Module (CMS03)

## Figure 4.3 shows Manage Volunteer Module subsystem.

|  |
| --- |
|  |

Figure 4.3: Manage Volunteer Module Detail Design

### 4.3.1 Manage Volunteer Package

The purpose of this class is to describe the volunteer details and relationship between other classes in the Manage Volunteer Module.

This package consists of:

1. Volunteer class
2. Distribution class
3. Address class
4. Check In Page
5. Check Out Page
6. Distribution Page
7. Check Out Controller
8. Volunteer Manager Controller

#### 4.3.1.1 Class Volunteer Design

This subparagraph specifies the design of volunteer check in.

1. **Input/Output data elements**

List of input and output data elements:

Input : volunteerName, volunteerIc\_no, volunteerAddress,

volunteerPhone, volunteerEmail, volunteerGender,

volunteerDistribution

Output : display volunteer information

1. **Local data elements**

|  |  |
| --- | --- |
| Name | volunteerName |
| Description | Contain name user information |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.3.1: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | volunteerIc\_no |
| Description | Contain IC number user information |
| Data Type | Integer |
| Precision/resolution | Number |

Table 4.3.2: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | volunteerAddress |
| Description | Contain address of user information |
| Data Type | Address |
| Precision/resolution | - |

Table 4.3.3: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | volunteerPhone |
| Description | Contain phone number of user information |
| Data Type | Integer |
| Precision/resolution | Number |

Table 4.3.4: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | volunteerEmail |
| Description | Contain email of user information |
| Data Type | String |
| Precision/resolution | Alphaber and number |

Table 4.3.5: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | volunteerGender |
| Description | Contain gender user information |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.3.6: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | volunteerDistribution |
| Description | Contain distribution of task information |
| Data Type | Distribution |
| Precision/resolution | - |

Table 4.3.7: Local Data Definition for data elements

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class.

Class Type : Display volunteer info

Responsibility : Accept and send information to database

Attributes : name, ic\_no, address, phone, email, gender.

Methods : setVolunteerName, getVolunteerName, setVolunteerIc\_no,

getVolunteerIc\_no, getVolunteerAddress, setVolunteerPhone,

getVolunteerPhone, setVolunteerGender, getVolunteerGender,

getVolunteerDistribution

**1. getVolunteerName ( )**

**Responsibility :** get the name user information

**Input Parameter :** None

**Output Parameter :** Volunteer name

**Algorithm :**

BEGIN

getVolunteerName ( );

END

**2. getVolunteerIc\_no ( )**

**Responsibility :** get the I/C number user information

**Input Parameter :** None

**Output Parameter :** Volunteer I/C

**Algorithm :**

BEGIN

getVolunteerIc\_no ( );

END

**3. getVolunteerPhone( )**

**Responsibility :** get the phone user information

**Input Parameter :** None

**Output Parameter :** Volunteer phone number

**Algorithm :**

BEGIN

getVolunteerPhone ( );

END

**4. getVolunteerGender ( )**

**Responsibility :** get the gender user information

**Input Parameter :** None

**Output Parameter :** Volunteer gender

**Algorithm :**

BEGIN

getVolunteerGender ( );

END

**5. setVolunteerName (vName)**

**Responsibility :** set the name user information

**Input Parameter :** vName

**Output Parameter :** None

**Algorithm :**

BEGIN

setVolunteerName (vName);

END

**6. setVolunteerIc\_no (IcNo)**

**Responsibility :** set the I/C number user information

**Input Parameter :** IcNo

**Output Parameter :** None

**Algorithm :**

BEGIN

setVolunteerIc\_no (IcNo);

END

**7. setVolunteerPhone (phone)**

**Responsibility :** set the phone user information

**Input Parameter :** phone

**Output Parameter :** None

**Algorithm :**

BEGIN

setVolunteerPhone (phone);

END

**8. setVolunteerGender (gender)**

**Responsibility :** set the gender user information

**Input Parameter :** gender

**Output Parameter :** None

**Algorithm :**

BEGIN

setVolunteerGender (gender);

END

**9. setVolunteerAddress (address)**

**Responsibility :** set the address user information

**Input Parameter :** address

**Output Parameter :** None

**Algorithm :**

BEGIN

setVolunteerAddress (address);

END

#### 4.3.1.2 Class Address Design

This subparagraph specifies the design of volunteer address.

1. **Input/Output data elements**

List of input and output data elements:

Input : volunteerHouse\_no, volunteerStreet\_name, volunteerCity

Output : display address details

1. **Local data elements**

|  |  |
| --- | --- |
| Name | volunteerHouse\_no |
| Description | Contain house number user information |
| Data Type | Integer |
| Precision/resolution | - |

Table 4.3.8: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | volunteerStreet\_name |
| Description | Contain street name of user address |
| Data Type | String |
| Precision/resolution | - |

Table 4.3.9: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | volunteerCity |
| Description | Contain city of user address |
| Data Type | String |
| Precision/resolution | - |

Table 4.3.10: Local Data Definition for data elements

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this

class

Class Type : Address class

Responsibility : Accept and send address information to volunteer database

Attributes : volunteerHouse\_no, volunteerStreet\_name, volunteerCity

Methods : getVolunteerHouse\_no, setVolunteerHouse\_no,

getVolunteerStreet\_name, setVolunteerStreet\_name,

setVolunteerCity, getVolunteerCity

**1. getVolunteerHouse\_no ( )**

**Responsibility :** get the house number user information

**Input Parameter :** None

**Output Parameter :** House no

**Algorithm :**

BEGIN

getVolunteerHouse\_no ( );

END

**2. getVolunteerStreet\_name()**

**Responsibility :** get the street name of user address

**Input Parameter :** None

**Output Parameter :** Street name

**Algorithm :**

BEGIN

getVolunteerStreet\_name ( );

END

**3. getVolunteerCity ()**

**Responsibility :** get the city of user address

**Input Parameter :** None

**Output Parameter :** Volunteer name

**Algorithm :**

BEGIN

getVolunteerCity ( );

END

**5. setVolunteerHouse\_no ( houseNo)**

**Responsibility :** set the house number user information

**Input Parameter :** houseNo

**Output Parameter :** None

**Algorithm :**

BEGIN

setVolunteerHouse\_no ( houseNo);

END

**6. setVolunteerStreet\_name (streetName)**

**Responsibility :** set the street name of user address

**Input Parameter :** streetName

**Output Parameter :** None

**Algorithm :**

BEGIN

setVolunteerStreet\_name (streetName);

END

**7. setVolunteerCity (city)**

**Responsibility :** set the city of user address

**Input Parameter :** city

**Output Parameter :** None

**Algorithm :**

BEGIN

setVolunteerCity (city);

END

#### 4.3.1.3 Class Distribution Design

This subparagraph specifies the design of volunteer address.

1. **Input/Output data elements**

List of input and output data elements:

Input : task\_id, task\_name, centre\_name, category

Output : distribute task and centre name to volunteer

1. **Local data elements**

|  |  |
| --- | --- |
| Name | task\_id |
| Description | Contain id of task |
| Data Type | Integer |
| Precision/resolution | - |

Table 4.3.11: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | task\_name |
| Description | Contain name of task |
| Data Type | String |
| Precision/resolution | - |

Table 4.3.12: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | centre\_name |
| Description | Contain name of centre |
| Data Type | String |
| Precision/resolution | - |

Table 4.3.13: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | Category |
| Description | Contain category of task |
| Data Type | String |
| Precision/resolution | - |

Table 4.3.14: Local Data Definition for data elements

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class

Class Type : Distribution class

Responsibility : Accept and send information distribution of task

Attributes : task\_id, task\_name, centre\_name, category

Methods : getTask\_id, setTask\_id, getTask\_name, setTask\_name,

getCentre\_name, setCentre\_name, getCategory, setCategory

**1. getTask\_id ( )**

**Responsibility :** get the id of task

**Input Parameter :** None

**Output Parameter :** task id

**Algorithm :**

BEGIN

getTask\_id ( );

END

**2. getTask\_name ()**

**Responsibility :** get the name of task

**Input Parameter :** None

**Output Parameter :** Task name

**Algorithm :**

BEGIN

getTask\_name ( );

END

**3. getCentre\_name( )**

**Responsibility :** get the name of centre

**Input Parameter :** None

**Output Parameter :** centre name

**Algorithm :**

BEGIN

getCentre\_name ( );

END

**4. getCategory ( )**

**Responsibility :** get the category of task

**Input Parameter :** None

**Output Parameter :** Category

**Algorithm :**

BEGIN

getCategory ( );

END

**5. setTask\_id ( taskID)**

**Responsibility :** set the id of task

**Input Parameter :** taskID

**Output Parameter :** None

**Algorithm :**

BEGIN

setTask\_id (taskID);

END

**6. setTask\_name (taskName)**

**Responsibility :** set the name of task

**Input Parameter :** taskName

**Output Parameter :** None

**Algorithm :**

BEGIN

setTask\_name (taskName);

END

**7. setCentre\_name (centreName)**

**Responsibility :** set the name of centre

**Input Parameter :** centreName

**Output Parameter :** None

**Algorithm :**

BEGIN

setCentre\_name (centreName);

END

**8. setCategory (category)**

**Responsibility :** set the category of task

**Input Parameter :** category

**Output Parameter :** None

**Algorithm :**

BEGIN

setCategory (category);

END

## 4.4 FMS-Inventory (CMS04)

This paragraph is divided into the following subparagraphs to identify and describe Flood Management System.

|  |
| --- |
|  |

Figure 4.4.1: Inventory Design

### 4.4.1 Inventory Module Package

The purpose of this class is to describe the items’ details and relationship between other classes in the inventory module package.

This package consists of:

1. Items class
2. Food class
3. Clothes class
4. Others class
5. ItemsController
6. ManageInventory class

#### 4.4.1.1 Class Items Design

This subparagraph specifies the design of items details

1. **Input/Output data elements**

List of input and output data elements:

Input : ItemsName, ItemsQuantity, ItemsType

Output : Display the details of the items

1. **Local data elements**

|  |  |
| --- | --- |
| Name | ItemsName |
| Description | Name of the items for the victim |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.4.1: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | ItemsQuantity |
| Description | Quantity of the items |
| Data Type | Integer |
| Precision/resolution | Number |

Table 4.4.2: Local Data Definition fordata elements

|  |  |
| --- | --- |
| Name | ItemsType |
| Description | Type of the items |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.4.3: Local Data Definition fordata elements

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class

Class Type : Items class

Responsibility : Accept and send the information of the items

Attributes : ItemsName, ItemsQuantity, ItemsType

Methods : getItemsName, getItemsQuantity, getItemsType, setItemsName,

setItemsQuantity, setItemsType

**1. *getItemsName()***

**Responsibility :** get the name of items

**Input Parameter :** None

**Output Parameter :** String

**Algorithm :**

BEGIN

getItemsName();

END

**2. *getItemsQuantity()***

**Responsibility :** get the quantity of the items

**Input Parameter :** None

**Output Parameter :** Integer

**Algorithm :**

BEGIN

getItemsQuantity();

END

**3. *getItemsType()***

**Responsibility :** get the quantity of the items

**Input Parameter :** None

**Output Parameter :** String

**Algorithm :**

BEGIN

getItemsType();

END

**4. *setItemsName()***

**Responsibility :** set the name of the items

**Input Parameter :** itemname

**Output Parameter :** None

**Algorithm :**

BEGIN

setItemsName(itemname);

END

**5. *setItemsQuantity()***

**Responsibility :** set the quantity of the items

**Input Parameter :** quantityitem

**Output Parameter :** None

**Algorithm :**

BEGIN

setItemsQuantity(quantityitem);

END

**6. *setItemsType()***

**Responsibility :** set the type of the items

**Input Parameter :** itemtype

**Output Parameter :** None

**Algorithm :**

BEGIN

setItemsQuantity(itemtype);

END

#### 4.4.1.2 Class Food Design

This subparagraph specifies the design of food details

1. **Input/Output data elements**

List of input and output data elements:

Input : ItemsExpiryDate

Output : Display the details of the food items

1. **Local data elements**

|  |  |
| --- | --- |
| Name | ItemsExpiryDate |
| Description | Expiry date of the food |
| Data Type | String |
| Precision/resolution | Alphabet and number |

Table 4.4.4: Local Data Definition for data elements

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class

Class Type : Food class

Responsibility : Accept and send the information of the food items

Attributes : ItemsExpiryDate

Methods : getItemsExpiryDate, setItemsExpiryDate

**1. *getItemsExpiryDate()***

**Responsibility :** get the expiry date of food items

**Input Parameter :** None

**Output Parameter :** String

**Algorithm :**

BEGIN

getItemsExpiryDate();

END

**2. *setItemsExpiryDate()***

**Responsibility :** set the expiry date of food items

**Input Parameter :** itemexpirydate

**Output Parameter :** None

**Algorithm :**

BEGIN

setItemsExpiryDate(itemexpirydate);

END

#### 4.4.1.3 Class Clothes Design

This subparagraph specifies the design of clothes details

1. **Input/Output data elements**

List of input and output data elements:

Input : ItemsMaterial

Output : Display the details of the clothes items

1. **Local data elements**

|  |  |
| --- | --- |
| Name | ItemsMaterial |
| Description | Material of the clothes |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.4.5: Local Data Definition for data elements

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class

Class Type : Clothes class

Responsibility : Accept and send the information of the clothes items

Attributes : ItemsMaterial

Methods : getItemsMaterial, setItemsMaterial

**1. *getItemsMaterial()***

**Responsibility :** get the material of clothes items

**Input Parameter :** None

**Output Parameter :** String

**Algorithm :**

BEGIN

getItemsMaterial();

END

**2. *setItemsMaterial()***

**Responsibility :** set the material of clothes items

**Input Parameter :** itemmaterial

**Output Parameter :** None

**Algorithm :**

BEGIN

setItemsMaterial(itemmaterial);

END

#### 4.4.1.4 Class Others Design

This subparagraph specifies the design of others details

1. **Input/Output data elements**

List of input and output data elements:

Input : ItemsDescription

Output : Display the details of the other items

1. **Local data elements**

|  |  |
| --- | --- |
| Name | ItemsDescription |
| Description | Description of the items |
| Data Type | String |
| Precision/resolution | Alphabet |

Table 4.4.6: Local Data Definition for data elements

1. **Algorithms**

This section states the purpose and describes in detail the algorithms of this class

Class Type : Others class

Responsibility : Accept and send the information of the other items

Attributes : ItemsDescription

Methods : getItemsDescription, setItemsDescription

**1. *getItemsDescription()***

**Responsibility :** get the description of other items

**Input Parameter :** None

**Output Parameter :** String

**Algorithm :**

BEGIN

getItemsDescription();

END

**2. *setItemsDescription()***

**Responsibility :** set the description of other items

**Input Parameter :** itemdescription

**Output Parameter :** None

**Algorithm :**

BEGIN

getItemsDescription(itemdescription);

END

## 4.5 Generate Report Module (CMS05)

## report class diagramFigure 4.5 shows Generate Report Module subsystem.

Figure 4.5: Generate Report Module Detail Design

### 4.5.1 Generate Report Package

The purpose of this class is to enable public and admin to view and print reports according to their needs.

This package consists of :

1. ReportController class

* Center class
* Victim class
* Items class

1. Report class

* Daily
* Weekly
* Monthly

1. ReportView class

**a) Input/Output data elements**

List of input and output data elements:

Input: type of report, category of report, report date

Output: Display the details of report

**b) Local data elements**

|  |  |
| --- | --- |
| Name | type |
| Description | type of report |
| Data Type | string |
| Precision/resolution | Example: Center report/ Victim Report/ Items Report |

Table 4.5.1: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | category |
| Description | category of report |
| Data Type | string |
| Precision/resolution | Example: Daily report/ Weekly report/ Monthly report |

Table 4.5.1: Local Data Definition for data elements

|  |  |
| --- | --- |
| Name | date |
| Description | date of required date |
| Data Type | string |
| Precision/resolution | Alphabet and number |

Table 4.5.1: Local Data Definition for data elements

#### 4.5.1.1Class Report Model Design

This subparagraph specifies the design of Report Model Class

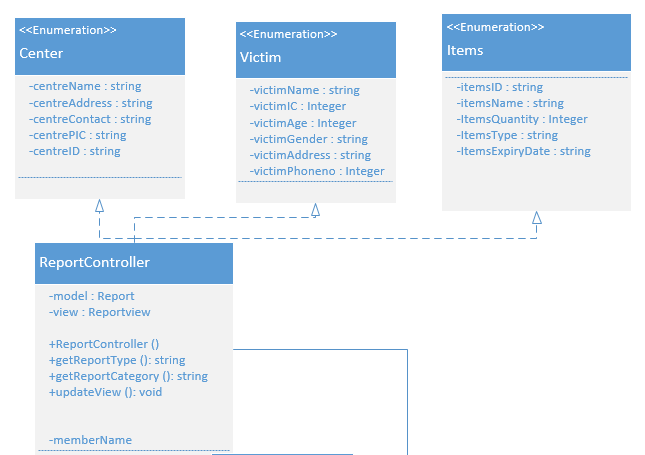
ReportController class

Figure 4.5.2.1 :ReportController Class

**Report Controller**

Class Type : Controller class

Responsibility : Determine and control the report management for the system.

Attributes : report, reportView

**Methods :** reportController, getType, getCategory, getDate

**1. *ReportController()***

**Responsibility :** to control which methods to call and return

value

**Input Parameter :** none

**Output Parameter :** none

**Algorithm :**

BEGIN

reportController()

END

**2. *getType()***

**Responsibility :** toprompt user to choose for type of report

**Input Parameter :** type

**Output Parameter :** string

**Algorithm :**

BEGIN

getType()

END

**3.*getCategory()***

**Responsibility :** toprompt user to choose for category of report

**Input Parameter :** category

**Output Parameter :** string

**Algorithm :**

BEGIN

getCategory()

END

**4.*getDate()***

**Responsibility :** toprompt user to enter for needed date of report

**Input Parameter :** date (string)

**Output Parameter :** string

**Algorithm :**

BEGIN

getDate()

END

Report class

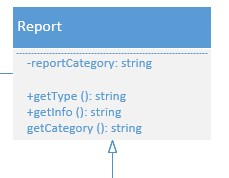


Figure 4.5.2.1 :Report Class

**Report**

Class Type : Report class

Responsibility : to get the category, report details and date of report

Attributes : getInfo, getdate, getCategory

**Methods :** reportController, getType, getCategory, getDate

**1.*getInfo()***

**Responsibility :** toget details from chosen type and category of report from database

**Input Parameter :** type, category

**Output Parameter :** string

**Algorithm :**

BEGIN

getInfo()

END

**2.*getDate()***

**Responsibility :** toprompt user to enter for needed date of report

**Input Parameter :** date

**Output Parameter :** string

**Algorithm :**

BEGIN

getDate()

END

**3.*getCategory()***

**Responsibility :** toprompt user to choose for category of report

**Input Parameter :** category

**Output Parameter :** string

**Algorithm :**

BEGIN

getCategory()

END

ReportView class

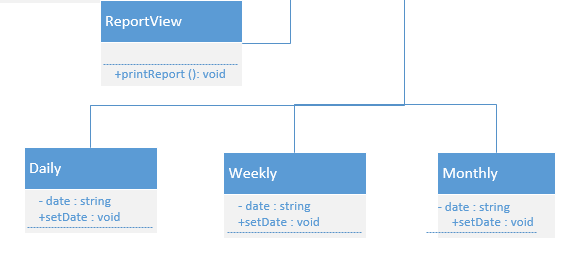


Figure 4.5.2.1 :ReportView Class

**ReportView**

Class Type : ReportView class

Responsibility : to get the date and to print report

Attributes : printReport, setDate

**Methods :** printReport, setDate

***1. getDate()***

**Responsibility :** toprompt user to enter for needed date of report

**Input Parameter :** date

**Output Parameter :** string

**Algorithm :**

BEGIN

getDate()

END

# 5. GLOBAL DATA OF CSCI

Class Name : Centre

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | KEY | REFERENCE | DESCRIPTION |
| centreId | number | Primary Key | No | Centre id |
| centreName | Varchar | N/A | No | Centre name |
| centreAddress | Varchar | N/A | No | Centre address |
| centreContact | number | N/A | No | Centre contact number |
| centrePic | Varchar | N/A | No | Centre person in charge |
| centreStatus | Varchar | N/A | No | Centre status |
| centreCapacity | number | N/A | No | Centre capacity |

Class Name : Address

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | KEY | REFERENCE | DESCRIPTION |
| centreId | number | Primary Key | No | Centre id |
| centreName | Varchar | N/A | No | Centre name |
| centreAddress | Varchar | N/A | No | Centre address |
| centreContact | number | N/A | No | Centre contact number |
| centrePic | Varchar | N/A | No | Centre person in charge |
| centreStatus | Varchar | N/A | No | Centre status |
| centreCapacity | number | N/A | No | Centre capacity |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | KEY | REFERENCE | DESCRIPTION |
| addressName | Varchar | N/A | No | Address name |
| addressPoscode | Number | N/A | No | Address poscode |
| addressDistrict | Varchar | N/A | No | Address district |
| addressCountry | Varchar | N/A | No | Address country |

Class Name : PicDetail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | KEY | REFERENCE | DESCRIPTION |
| picName | Varchar | N/A | No | Person in charge name |
| picStaffId | Varchar | N/A | No | Person in charge staff id |
| picContactNo | Number | N/A | No | Person in charge contact number |

Victim Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | KEY | REFERENCE | DESCRIPTION |
| victimName | Varchar | N/A | No | Victim name |
| victimIc | Number | N/A | No | Victim ic |
| victimAge | number | N/A | No | Victim age |
| victimGender | Varchar | N/A | No | Victim gender |
| victimAddress | Varchar | N/A | No | Victim address |
| victimPhoneNo | number | N/A | No | Victim phone number |

Volunteer Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | KEY | REFERENCE | DESCRIPTION |
| name | Varchar | N/A | No | Volunteer name |
| ic\_no | number | N/A | No | Volunteer ic number |
| address | Varchar | N/A | No | Volunteer address |
| phone | number | N/A | No | Volunteer phone |
| email | Varchar | N/A | No | Volunteer email |
| gender | Varchar | N/A | No | Volunteer gender |

Address Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | KEY | REFERENCE | DESCRIPTION |
| house\_no | number | N/A | No | Address number |
| street\_name | Varchar | N/A | No | Address street name |
| city | Varchar | N/A | No | Address city |
| category | Varchar | N/A | No | Address category |

Item Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | KEY | REFERENCE | DESCRIPTION |
| itemsId | number | N/A | No | Items id |
| itemsName | Varchar | N/A | No | Items name |
| itemsQuantity | number | N/A | No | Items quantity |
| itemsType | Varchar | N/A | No | Items type |
| itemsExpiryDate | Varchar | N/A | No | Items expity date |

# 

# 6. SHARED DATA OF CSCI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | KEY | REFERENCE | DESCRIPTION |
| donId | Int | Primary Key | No | Donation id number |
| donItem | Varchar |  |  | Name of donation item |
| donType | Varchar |  |  | Type of donation item |
| donQuantity | number |  |  | Quantity of donation item |
| donDate | Varchar |  |  | Date of receiving item |
| Itemid | Varchar | PK | SDD-REQ-300 | The id is use as reference. |
| ItemName | Varchar |  | SDD-REQ-300 | The name is to make sure that the item are known. |
| Itemexpired\_date | Varchar |  | SDD-REQ-300 | To make sure the item still in good condition. |
| Itemtype | Varchar |  | SDD-REQ-300 | To classify the item into group. |
| Itemquantity | number |  | SDD-REQ-300 | To inform the value of item available. |
| quantity | number | PK | SDD-REQ-300 | To know the value of flood victim. |
| place | Varchar |  | SDD-REQ-300 | To know the specific area for distribution. |

**7. REQUIREMENT TRACEABILITY**

|  |  |  |
| --- | --- | --- |
| Requirement | | Description |
| Source | Allocated |
| FMS-FCM-01 |  | Center Manager is allowed to register the new center by entering its information. |
| FMS-FCM-02 |  | Center Manager are able to check the status of the center whether it is available or not based on the listed center that have been registered. |
| FMS-FSC-01 |  | Staff Center are allowed to register the victim by entering their personal information. The registration shows that the victim has successfully check-in at the center. |
| FMS-FSC-02 |  | Staff Center are allowed to do the check-out based on the recorded information of the victims. |
| FMS-FVM-01 |  | Volunteer Manager are allowed to register the volunteer by entering their information. The volunteer will successfully check-in at the center. |
| FMS-FVM-02 |  | Volunteer Manager are allowed to do the check-out for the volunteer. |
| FMS-FVM-03 |  | Volunteer Manager are allowed to update volunteer information. |
| FMS-FVM-04 |  | Volunteer Manager are allowed to allocate workplace for the volunteer based on the type of task that they have selected. |
| FMS-FIM-01 |  | Inventory Manager are allowed to add new items in the inventory (Stock-in) by entering the details and also its category. |
| FMS-FIM-02 |  | Inventory Manager are allowed to update items that have been distributed or deposed. |
| FMS-FA-01 |  | The Administrator will generate report base on the recorded information about center registration, victims, volunteer and activities of inventory. |
| FMS-FP-01 |  | The user are allowed to view the report that have been generated by the administrator |

# 8. NOTES

Abbreviation used:

UMP - Universiti Malaysia Pahang

CDRL - Context Data Requirement List

CSCI - Computer Software Configuration Items

IRS - Interface Requirement Specification

OOAD - Object Oriented Analysis Design

SDD - Software Development Document

SDP - Software Development Plan

SRS - Software Requirement Specification

SPM - Software Programmer Manual

SUM - Software User’s Manual

SOW - Statement of Work

UML - Unified Modelling Language

FMS - Flood Management System