

**Software Requirement Specification**

**for**

**Crisis Management System**

**Prepared by**

**Team Xtreme**

|  |  |  |
| --- | --- | --- |
| **Matric Number** | **Student Name** | **Student Email** |
| U1420448A | HO SONG YAN | HOSO0003@E.NTU.EDU.SG |
| U1421315F | KONG ZHONG HAN | KONG0110@E.NTU.EDU.SG |
| U1422579L | CHUA WEE HANG | CHUA0724@E.NTU.EDU.SG |
| U1422381G | TAN YAN LING | YTAN070@E.NTU.EDU.SG |
| U1421356F | TAN TIAN WEN | TTAN021@E.NTU.EDU.SG |
| U1420552H | GOH KHAI HONG | GOHK0062@E.NTU.EDU.SG |
| U1422074E | GAYLE NATALIE ANG AI TING | GAYL0002@E.NTU.EDU.SG |
| U1422041J | NG HAN BENG | HNG013@E.NTU.EDU.SG |
| U1420239G | ANDREAS CHRISNA MAYONG | C140067@E.NTU.EDU.SG |
| U1422110J | GOH PEI SHAN | PGOH006@E.NTU.EDU.SG |

Table of Contents

|  |  |
| --- | --- |
| [Introduction](#h.4a9g3lac9hnz) | 4 |
| [Purpose](#h.lsl061l2kets) | 4 |
| [Product Scope](#h.2zvv2lq7ygy3) | 4 |
| [Users and Stakeholders](#h.rrxbq5zd9to1) | 5 |
| [Assumptions and Constraints](#h.qefpt0bt7qqf) | 5 |
| [References](#h.fuxt4hn2gd3z) | 6 |
| [Overall Description](#h.phnl5jv36wdh) | 7 |
| [Product Perspective](#h.c0temvxeal79) | 7 |
| [Product Functions](#h.8uonm6mal4rv) | 7 |
| [User Classes and Characteristics](#h.wu8ef52nqr0c) | 7 |
| [Operating Environment](#h.4jim1ofayllz) | 7 |
| [Design and Implementation Constraints](#h.8mqvdyod6mmn) | 7 |
| [Use Case Model](#h.hrspy9dasgfn) | 7 |
| [Use Case Descriptions](#h.d3up8ucqwjs7) | 8 |
| [Class Diagram](#h.nxck25nui02t) | 24 |
| [Context Diagram](#h.gr4zbr7knr4s) | 24 |
| [Data Flow Diagram](#h.rojcztwrhcaz) | 25 |
| [Architectural Diagram](#h.dtgh4d4lmv2t) | 26 |
| [Entity-Relation Diagram](#h.5eqrm8ikf6dl) | 27 |
| [Component Diagram](#h.n2sx7u917o9q) | 28 |
| [Sequence Diagram](#h.eber1u6ql1qe) | 29 |
| [State Transition Diagram](#h.2e406q6mo0gy) | 31 |
| [Activity Diagram](#h.5vepnv71itep) | 32 |
| [User Documentation](#h.71oqn990ne4y) | 32 |
| [Assumptions and Dependencies](#h.jcu0o3foocsv) | 32 |
| [External Interface Requirements](#h.473ldmkc1jik) | 33 |
| [User Interfaces](#h.wq1psfkbx3g0) | 33 |
| [Map View](#h.7dxbid1os11m) | 33 |
| [Map View - Crisis Overview](#h.xk56zui3djjd) | 34 |
| [Incident List](#h.fztgcstcimc0) | 35 |
| [Incident Status Update](#h.ryyfii4by2hb) | 36 |
| [Incident Submission Form](#h.xvaf8rxshtv8) | 37 |
| [Request Form](#h.lhvav4buuwb) | 38 |
| [Incident Summary](#h.dxdnzt5jnqgf) | 39 |
| [Hardware Interfaces](#h.ez89ph3jkaa0) | 39 |
| [Software Interfaces](#h.8b456q60qnib) | 39 |
| [Communication Interfaces](#h.34lmuiikshb9) | 40 |
| [System Features](#h.nwqmbtoyclds) | 42 |
| [Functional Requirement](#h.mhfzudqn0vwy) | 42 |
| [Software Requirements](#h.cbttmxy99os) | 44 |
| [Other Non-functional Requirements](#h.btgxxh6bajgg) | 45 |
| [Performance Requirements](#h.n7b5hodo9c31) | 45 |
| [Security Requirements](#h.himd68dbrde1) | 45 |
| [Software Quality Attributes](#h.t2o5iublo639) | 45 |
| [Testing](#h.5hakhnzg2ga4) | 47 |
| [Stub Driver Test](#h.ksrxohmku272) | 47 |
| [Database Script](#h.evr1xbanowm1) | 48 |
| [Load Test by ApacheBench](#h.oyk7gogygr8a) | 52 |
| [Testing strategies and techniques](#h.vsgpkv1f6h50) | 53 |
| [System Level Functional Tests](#h.4tbjv6keu5ip) | 54 |
| [Recommendations](#h.2ni4dtlhi4wv) | 67 |
| [Conclusion](#h.7sj0z2lvpdr) | 67 |
| [Appendix A: Glossary](#h.i7zk5n6bcm8z) | 68 |
| [Data Dictionary](#h.sxqpk3vtynlo) | 68 |
| [Appendix B: Analysis Models](#h.9vgxvxk8941h) | 70 |
| [Architecture Design Analysis](#h.ptjp7471duur) | 70 |
| [Layered Architecture](#h.n61iawkhy7d8) | 70 |
| [Main Program/Subroutines with Shared Data](#h.w5rs6y244j2l) | 70 |
| [Pipe and Filter](#h.o5jkr51qoosb) | 71 |
| [Abstract Data Type](#h.nlhnfppw6mke) | 71 |
| [Design Principles](#h.63vwr0rqjjnh) | 72 |
| [Scenario Evaluations](#h.3zue2r46841w) | 72 |
| [Evaluation of CMS Architectures](#h.5t0h5pcr3xs9) | 74 |
| [Requirements Priority Table](#h.qy5jcjz2h5x) | 76 |
| [Appendix C: Prototype (User Interface)](#h.olmn1c19qpik) | 78 |

# Introduction

## Purpose

The purpose of this document is to describe the requirements specification for the Crisis Management System (CMS), a product which allows for seamless collaboration between government agencies and updating the public in terms of emergency. The CMS will serve as a command-control platform, enabling quick and proper response to any dangerous situations within the country.

## Product Scope

A Crisis Management System (CMS) will be developed with the aim of being a platform for seamless collaborations between agencies in the times of impending crisis. It allows for crisis handling by providing timely responses of emergency and rescue assistance to the civilians. When an incident is being reported by a member of the public, call centre operators will input it into the CMS and request dispatch from relevant agencies to deal with the incident. The CMS allows for the respective authorities to be on alert as they will be able to observe the conditions and when necessary, convey to both public and the cabinet ministers of the crisis.

On a web graphical use interface, the system will provide real-time status updates on a map of Singapore, integrated with weather conditions, dengue hot spot, haze information. With these features, users of the CMS can conveniently observe the conditions of Singapore.

Furthermore, useful information will be sent to the public periodically through social media platforms, i.e. SMS, Twitter and Facebook. As crisis can hit at anytime and in any form, this prepares the public with knowledge on what necessary precautions or steps they should take when such dangerous situations occur. This periodical update of information to the public also proves the CMS’s functionality of being able to update the respective social media platforms automatically. In addition, other than allowing for the prime minister to observe the conditions on the CMS, a status report summarizing key indicators and trends will be sent over to the Prime Minister’s Office every 30 minutes.

Hence, with the functionalities of the CMS, risks can be managed efficiently as it is a system that updates the relevant agency bodies and allows for seamless interaction between the them. Also, credible data will be processed into the system, allowing for the public to be well informed.

## Users and Stakeholders

Users of the system includes call centre operators, government agencies, key decision makers and ministers. Call centre operators will input cases into the CMS after receiving reports from the public of incidents witnessed. Agency governments include the Singapore Police Force and Singapore Civil Defence Force (SCDF) as they can access the CMS and request for other resources and also update the status of the incidents. Key decision makers can follow the states of the incidents reported and when there it is observed to be very severe, declare the incident as a crisis, alerting the other users of the CMS.

For the stakeholders, other than the users mentioned, it includes the public, which comprises of the residents of Singapore. The public will want to be informed immediately should there be any crisis occurring in Singapore that might put their lives at risk, and hence the possible steps and precautions to take.

## Assumptions and Constraints

**Assumptions**

1. Agencies must dispatch immediately upon receiving the SMS request.
2. NEA must be able to provide latest PSI and weather data.
3. All the users of the CMS have received proper training in using the CMS such that they are aware of how they should navigate the system.

**Constraints**

A design constraint refers to some limitation on the conditions under which the system is developed, or on the requirements of the system. As the system serves as a crucial channel to inform various government agencies on prevailing emergency situations and crisis management, it is a necessity to ensure constant reliability to the system. As such, complex algorithms to handle faults have to be implemented to improve reliability and quality assurance of the system, albeit being costlier than simpler algorithms. This allows the system to be more responsive to a stimuli at a given interval of time.

Also, due to the implementation of various frameworks- such as Propel and Angular JS Framework- that helps to support the three layers of the Model-View-Controller Architecture, it is a requirement that developers be restricted with the applied programming language- such as PHP and Java- that are bounded by these frameworks.

Further, the system is supported by various communication and informative channels such as Facebook, Twitter, SMS and Google Map. Any changes in the client’s regulations and usage extensibility will also affect the system. For example, high usage of a particular channel may result in an accumulated charge of use.

Lastly, as the architectural is grouped according to its modules in accordance to its purpose, it is subjected with inheritance within classes. Therefore, it is easy to move this dependent code to a new environment, but is not feasible if major change would be made within a particular module. Therefore, this effect will make it difficult should a transitional shift be made in the architecture style.

# Overall Description

## Product Features

1. Incident submission by call centre operator
2. Real-time incident updates and monitors on map
3. Crisis declaration by key decision maker
4. Automatic updates to social media platforms (Facebook and Twitter)
5. Direct communication to government agencies for dispatchment request
6. Automatic report generations for Prime Minister

## User Classes and Characteristics

The CMS has two user classes, active users and passive users. Active users are people or bodies that interacts with CMS directly and actively inputting information to the CMS. On the other hand, passive users are people or bodies that interacts with CMS directly or indirectly, but only viewing the information displayed by CMS.

**Active Users**

1. Administrator
   1. Create and remove user accounts from CMS database
   2. Modify user accounts information and limitation of access to CMS system
2. Call centre operator
   1. Create incident based on information reported by the caller
   2. Create category of incident
   3. Modify incident information
   4. Request dispatchment
   5. View incident list and information
3. Government agencies
   1. Add follow up information for the distinct incident
   2. Request dispatchment
   3. View incident list and information
4. Key decision maker
   1. Modify crisis status for the distinct category or incident
   2. View incident list and information

**Passive Users**

There is only one type of passive user, which is member of public. Member of public can only interact indirectly with CMS through their reports to call centre and view the information posted by CMS on CMS Facebook page and Twitter account.

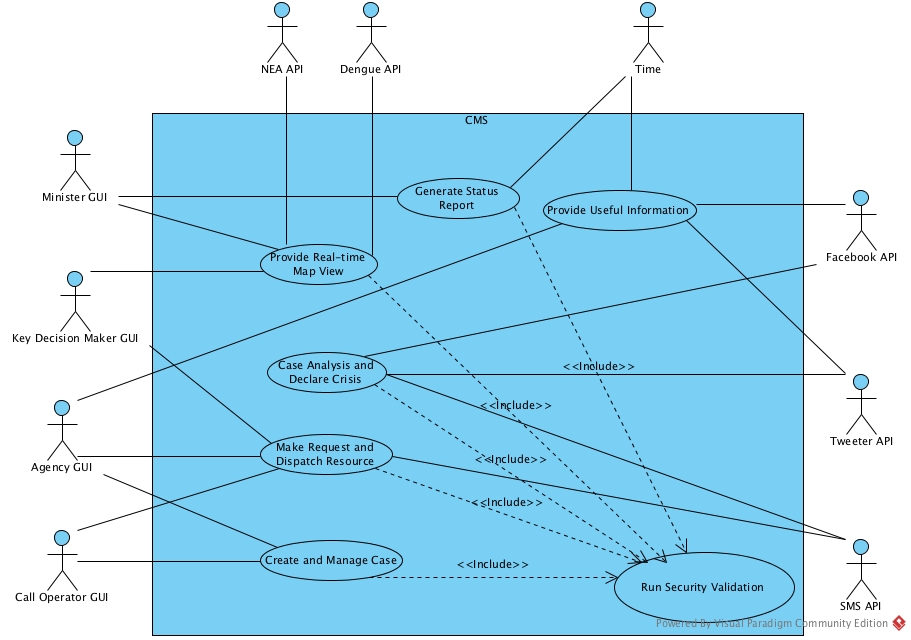
## 

## Operating Environment

## Design and Implementation Constraints

The some of the CMS features are dependent on information provided by its associated APIs, therefore its design and implementation is limited by how each API works, and how much and how the information is presented by the APIs. The CMS is also build upon multiple framework integrated together. Therefore, the design and implementation is also depends on how the framework can interact with each other and what features and constraints that the frameworks have.

### Use Case Model



### Use Case Descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.001 | | |
| Use Case Name: | Run Security Validation | | |
| Created By: | Tan Tian Wen | Last Updated By: | Tan Yan Ling |
| Date Created: | 18 Feb 2016 | Date Last Updated: | 14 Apr 2016 |

|  |  |
| --- | --- |
| Actor: | Call operator, Agency, Key decision maker, Administrator |
| Description: | Call operator has to log in before they can take any action. |
| Preconditions: | Call operator must have an existing account for the CMS. |
| Postconditions: | A pop-up message will show the call operator that they have successfully login to the CMS. |
| Priority: | High |
| Frequency of Use: | Frequent |
| Flow of Events: | 1. When call operator accesses the CMS, the home page will display a login page. 2. The login page requests call operator to input username and password. 3. System will validate the username and password input entered by the call operator. 4. Username and password have been successfully verified and call operator is being logged in into the CMS. The use case ends. |
| Alternative Flow: | 1. Call operator enters invalid username and password. 2. The system displays an error message. |
| Exceptions: | **Account Does Not Exist Error**  No account with the username input by the call operator can be found. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.002 | | |
| Use Case Name: | Generate Status Report | | |
| Created By: | Tan Tian Wen | Last Updated By: | Tan Yan Ling |
| Date Created: | 18 Feb 2016 | Date Last Updated: | 14 Apr 2016 |

|  |  |
| --- | --- |
| Actor: | Minister GUI, Time |
| Description: | The CMS will generate a status report and send it to the minister. |
| Preconditions: |  |
| Postconditions: | The minister follows up to see if any necessary actions are to be taken. |
| Priority: | High |
| Frequency of Use: | Every 30 minutes |
| Flow of Events: | 1. Time clocks 30 minutes and alerts system to update the minister by generating a status report. 2. System views incident database and collates statistics for all happenings. 3. System compares current statistics with the previous statistics. 4. System generates a report to show changes, displaying key indicators and trends. 5. System sends generated report to the minister. 6. System then notifies the office that the report is sent via email. |
| Alternative Flow: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.003 | | |
| Use Case Name: | Create and Manage CIncident | | |
| Created By: | Goh Pei Shan | Last Updated By: | Goh Pei Shan |
| Date Created: | 18 Feb 2016 | Date Last Updated: | 18 Feb 2016 |

|  |  |
| --- | --- |
| Actor: | Call Operator |
| Description: | Call operator will enter assisted request information from public into the system. |
| Preconditions: | Call operator has logged into CMS and has already received a call from the public. |
| Postconditions: | Call operator receives acknowledgement for submitted entry and response from selected government agencies. The request is updated and reflected on the map. If the case is not valid, call operator will receive explanation specifying why submission could not be processed (e.g. broken connection). |
| Priority: | High |
| Frequency of Use: | High |
| Flow of Events: | 1. Call operator chooses ‘new case’ option and an entry form appears. 2. Call operator chooses the category (emergency ambulance, rescue and evacuation, fire-fighting, gas leak control) and level of urgency for the incident. The system pre-selects the relevant government agencies. 3. Call operator must enter the name and mobile number of caller. 4. Call operator enters location of the incident: postal code, building name, unit number provided by the public. The system displays a list of incidents happening around the location. Call operator maps to existing case(s) (if applicable). 5. Call operator enters description (e.g. cause, casualty number, hazardous content) of the incident concisely and comprehensively. 6. Call operator enters the name of the incident’s category. 7. Call operator modifies or confirms the relevant government agencies selected. 8. Once completed and confirmed, call operator submits the entry, the system dispatches the request to selected government agencies through SMS. |
| Alternative Flow: |  |
| Exceptions: | **Mandatory Field Incomplete**  System highlights incomplete field(s) to call operator and prompts for a response. |
| Includes: | Login |
| Special Requirements: | Call operator receives acknowledgement for entry submission within 30 seconds. |
| Assumptions: | Assume the only source of case input is through calls.  Assume call operator cannot file a crisis. |
| Notes and Issues: | Case = Request |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.004 | | |
| Use Case Name: | View Map | | |
| Created By: | Tan Tian Wen | Last Updated By: | Ho Song Yan |
| Date Created: | 16 Feb 2016 | Date Last Updated: | 18 Feb 2016 |

|  |  |
| --- | --- |
| Actor: | Call operator, Agency |
| Description: | Call operators has an overview of cases on map |
| Preconditions: | User has to login to access the map view |
| Postconditions: |  |
| Priority: | High |
| Frequency of Use: | Every time after call operator has logged in to main page |
| Flow of Events: | 1. Call operator views the Map View. 2. Call operator filter the cases that are displayed on the map.    1. Call operator chooses to show all, dengue, haze, riot or viruses. 3. Call operator zooms to particular region in map to have detailed view. 4. Call operator clicks the pin on map to show details of the case. 5. Call operator clicks button to show or update the selected case. 6. Call operator click reset button to back to default map view. |
| Alternative Flow: |  |
| Exceptions: | **CMS.EX.1** – Dengue Singapore API has downtime.  System Response – Dialog Box appears informing user that Dengue Singapore API is facing downtime and the issue is being resolved.  **CMS.EX.2** – Google Map API has downtime.  System Response – Dialog Box appears informing user that Google Map API is facing downtime and the issue is being resolved.  **CMS.EX.3** – NEA API has downtime.  System Response – Dialog Box appears informing user that NEA API is facing downtime and the issue is being resolved.  **System Response for unexpected failure of use case execution** – A dialog box will pop up asking the user to send feedback to the system administrator and the application will close. |
| Includes: | Login, Map Renderer, NEA API Request, Dengue API Request |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.005 | | |
| Use Case Name: | Send Request | | |
| Created By: | Goh Khai Hong | Last Updated By: | Goh Khai Hong |
| Date Created: | 16 Feb 2016 | Date Last Updated: | 18 Feb 2016 |

|  |  |  |
| --- | --- | --- |
| Actor: | Call Operator, Agency | |
| Description: | The request will be sent by call operator to relevant agencies upon the submission of the particular case to check the severity of the particular case. In case there is a need, call operator will send the request for the help from the relevant agencies. | |
| Preconditions: | Call operator has to submit the case into the system first. Upon the submission of the case, the request will be sent to the relevant agencies. | |
| Postconditions: | The request is received and processed by relevant agencies. | |
| Priority: | High | |
| Frequency of Use: | Every time after call operator has input the case into the system. | |
| Flow of Events: | **Actor Step**  **Step 1.** Call operator input the case, indicate the relevant agencies and submit it into the system.  **Step 3.** Agency updates the incident’s status according to the real-world situation. If the status is “completed”, go to **Step 4**. If not, go to **Alt Step 4.** | **System Step**  **Step 2.** System send the request to the relevant agencies via SMS.  **Step 4.** System changes the status of the request to “completed” |
| Alternative Flow: | **Alt Step 4.** Agency send the request to relevant agencies for help if there is a need. | |
| Exceptions: | 1. **System Failed to Send The Request**     1. The request is unable to reach the relevant agencies. 2. **Request is Sent to The Irrelevant Agencies**    1. The request is sent to the irrelevant agencies and has been ignored. | |
| Includes: | Update Status, Crisis and Case Input | |
| Special Requirements: | The system will notify the call operator whether the request is received by the relevant agencies within a couples of second. | |
| Assumptions: | Every request is sent successfully(no request is failed to send) and acknowledged by relevant agencies. | |
| Notes and Issues: | **Issues**  The system will prompt the call operator if the request is failed to send and has being processed. How we can know that the request is failed to send or is being processed by relevant agencies? | |

|  |
| --- |
| 1. This use case begins after the call operator has submits the case into the system 2. Upon the submission of the case, system will send a request to the relevant agencies for sending someone down to check on the incident based on the assistance required. 3. The status of the particular case is now pending and waiting for the confirmation from the personnel of the relevant agencies 4. After the severity of the particular case is confirmed,either it is crisis or not, if needed, case operator could send the request to the relevant agencies for help. 5. The cases will be updated immediately on the map after the check request is granted and finished. 6. The status of the particular case is now changed based on the severity of the case. |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.006 | | |
| Use Case Name: | Update Status | | |
| Created By: | Ng Han Beng | Last Updated By: | Ng Han Beng |
| Date Created: | 18 Feb 2016 | Date Last Updated: | 18 Feb 2016 |

|  |  |
| --- | --- |
| Actor: | Call operator, Agency |
| Description: | Call operator will update information on the CMS map when new updates are received |
| Preconditions: | There is an existing case on the map |
| Postconditions: | Civil Defence shelters and status report for Prime Minister must be updated on the changes. |
| Priority: | High |
| Frequency of Use: | High |
| Flow of Events: | 1. Call operator receives information to change the case from “active” to “inactive” or vice versa, or just any normal updates on the severity of the case. 2. Call operator updates the status of the relevant case. 3. Information of the change is released to the public through Facebook, Twitter and SMS, if its change is from normal to crisis or crisis to normal. |
| Alternative Flow: | 1. Call operator receives information to change the case from “active” to “inactive” or vice versa, or just any normal updates on the severity of the case. 2. Call operator sends request to the relevant agencies if the case’s situation requires more assistance/backup. 3. Call operator updates the status of the relevant case. 4. Information of the change is released to the public through Facebook, Twitter and SMS, if its change is from normal to crisis or crisis to normal. |
| Exceptions: |  |
| Includes: | Crisis, Release Information |
| Special Requirements: |  |
| Assumptions: | The update is about an existing case. |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.007 | | |
| Use Case Name: | Dengue API Request | | |
| Created By: | Kong Zhong Han | Last Updated By: | Kong Zhong Han |
| Date Created: | 16 Feb 2016 | Date Last Updated: | 18 Feb 2016 |

|  |  |  |
| --- | --- | --- |
| Actor: | Dengue Singapore | |
| Description: | System requests from Dengue Singapore API for updated information of current Dengue incidents in Singapore. | |
| Preconditions: | The View Map use case is being called upon. | |
| Postconditions: | API returns with the requested information of Dengue Incidents in Singapore. | |
| Priority: | High | |
| Frequency of Use: | High | |
| Flow of Events: | **Actor Step**  **Step 2.** The API returns the requested information to the system. | **System Step**  **Step 1.** The system requests for the updated information of current dengue incidents in Singapore.  **Step 3.** The system parses the information and shows the incidents on the map. |
| Alternative Flow: |  | |
| Exceptions: | **CMS.EX.1** – Dengue Singapore API has downtime.  System Response – Dialog Box appears informing user that Dengue Singapore API is facing downtime and the issue is being resolved.  **System Response for unexpected failure of use case execution** A dialog box will pop up asking the user to send feedback to the system administrator and the application will close. | |
| Includes: | View Map | |
| Special Requirements: | 1. The system is able to correctly parse the information received from the API. 2. Dengue Singapore API must not have downtime. | |
| Assumptions: |  | |
| Notes and Issues: |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.008 | | |
| Use Case Name: | Map Renderer | | |
| Created By: | Kong Zhong Han | Last Updated By: | Kong Zhong Han |
| Date Created: | 16 Feb 2016 | Date Last Updated: | 18 Feb 2016 |

|  |  |  |
| --- | --- | --- |
| Actor: | Google Map API | |
| Description: | System requests from Google Map for the Singapore Map. | |
| Preconditions: | The View Map use case is being called upon. | |
| Postconditions: | API returns with the map of Singapore. | |
| Priority: | High | |
| Frequency of Use: | High | |
| Flow of Events: | **Actor Step**  **Step 2.** The API returns the requested information to the system. | **System Step**  **Step 1.** The system requests for the Singapore Map.  **Step 3.** The system parses the information and renders the map on the browser. |
| Alternative Flow: |  | |
| Exceptions: | **CMS.EX.2** – Google Map API has downtime.  System Response – Dialog Box appears informing user that Google Map API is facing downtime and the issue is being resolved.  **System Response for unexpected failure of use case execution**  A dialog box will pop up asking the user to send feedback to the system administrator and the application will close. | |
| Includes: | View Map | |
| Special Requirements: | 1. The system is able to correctly parse the information received from the API. 2. Google Map API must not have downtime. | |
| Assumptions: |  | |
| Notes and Issues: |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.009 | | |
| Use Case Name: | NEA API Request | | |
| Created By: | Kong Zhong Han | Last Updated By: | Kong Zhong Han |
| Date Created: | 16 Feb 2016 | Date Last Updated: | 18 Feb 2016 |

|  |  |  |
| --- | --- | --- |
| Actor: | NEA | |
| Description: | System requests from NEA for the Weather and Haze information of Singapore. | |
| Preconditions: | The View Map use case is being called upon. | |
| Postconditions: | API returns with the requested information of the Weather and Haze in Singapore. | |
| Priority: | High | |
| Frequency of Use: | High | |
| Flow of Events: | **Actor Step**  **Step 2.** The API returns the requested information to the system. | **System Step**  **Step 1.** The system requests for the Weather and Haze information in Singapore currently.  **Step 3.** The system parses the information and renders the information on the browser. |
| Alternative Flow: |  | |
| Exceptions: | **CMS.EX.3** – NEA API has downtime.  System Response – Dialog Box appears informing user that NEA API is facing downtime and the issue is being resolved.  **System Response for unexpected failure of use case execution**  A dialog box will pop up asking the user to send feedback to the system administrator and the application will close. | |
| Includes: | View Map | |
| Special Requirements: | 1. The system is able to correctly parse the information received from the API. 2. NEA API must not have downtime. | |
| Assumptions: |  | |
| Notes and Issues: |  | |

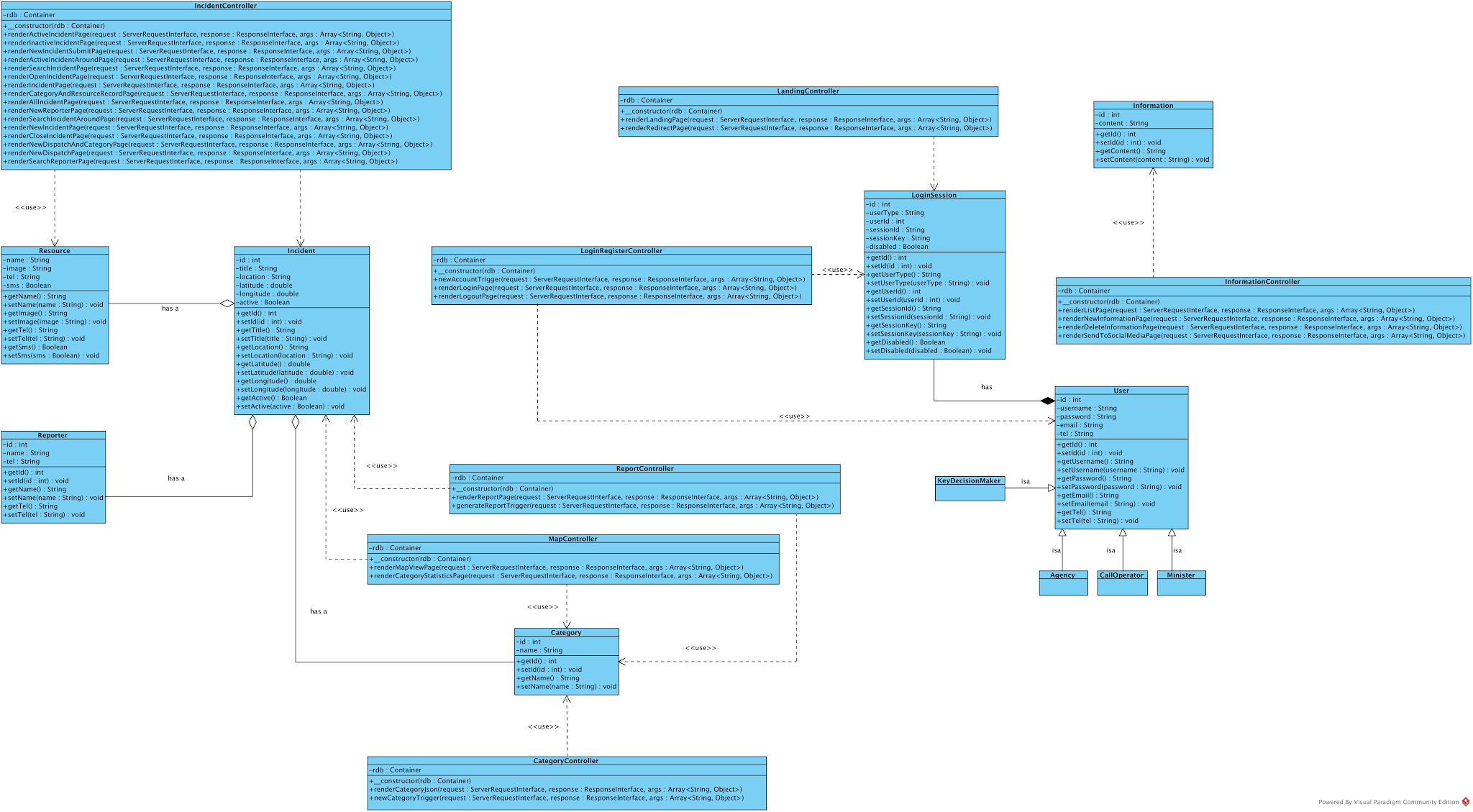
|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.010 | | |
| Use Case Name: | Crisis | | |
| Created By: | Gayle Natalie Ang | Last Updated By: | Ho Song Yan |
| Date Created: | 18 Feb 2016 | Date Last Updated: | 18 Feb 2016 |

|  |  |  |
| --- | --- | --- |
| Actor: | Key decision maker | |
| Description: | System indicates the crisis on the UI with blinking red color. | |
| Preconditions: | The Update Status use case will indicate if a situation is now a crisis | |
| Postconditions: | The system will post crisis report on Facebook and Twitter and send report to Prime Minister | |
| Priority: | High | |
| Frequency of Use: | Frequent | |
| Flow of Events: | **Actor Step**  **Step 1.** The key decision maker toggle the crisis status of the respective category or incident. | **System Step**  **Step 2.** The system parses the information and change to color of the UI. |
| Alternative Flow: |  | |
| Exceptions: |  | |
| Includes: | Send Request, Release Information | |
| Special Requirements: |  | |
| Assumptions: | 1. Crisis status is fully based on the judgement of key decision maker | |
| Notes and Issues: |  | |

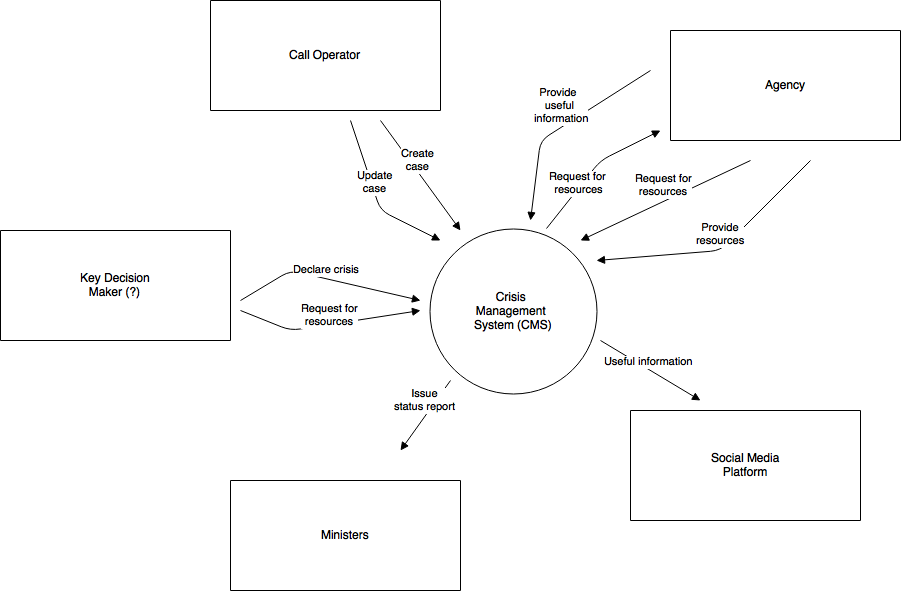
|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CMS.011 | | |
| Use Case Name: | Release information | | |
| Created By: | Gayle Natalie Ang | Last Updated By: | Chua Wee Hang |
| Date Created: | 18 Feb 2016 | Date Last Updated: | 19 Feb 2016 |

|  |  |
| --- | --- |
| Actor: | Call operator, Facebook , Twitter, SMS |
| Description: | System sends SMS to social media agencies to update the status of the crisis to the public via platforms, which include Facebook, Twitter and SMS. |
| Preconditions: | Call operator updates the status of the crisis. |
| Postconditions: | Public will be updated with the latest details of the crisis. |
| Priority: | High |
| Frequency of Use: | Frequent |
| Flow of Events: | 1. Once the status of the crisis is “Confirmed”, the system sends SMS to social media agencies to update the crisis. 2. The agencies release the information to public via social media platforms, which include Facebook, Twitter and SMS. 3. Public is being updated with the latest details of the crisis. The use case ends. |
| Alternative Flow: | 1. Once the status of the crisis is updated to “Resolved”, and the icon of that particular crisis is removed from the map, the system sends SMS to social media agencies to update the crisis. 2. The agencies release the resolved information to public via social media platforms, which include Facebook, Twitter and SMS. 3. Public is being updated with the latest details of the crisis. The use case ends. |
| Exceptions: | **System Response for failure to send message**  A dialog box will pop up telling the call operator that the message has failed to send through. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

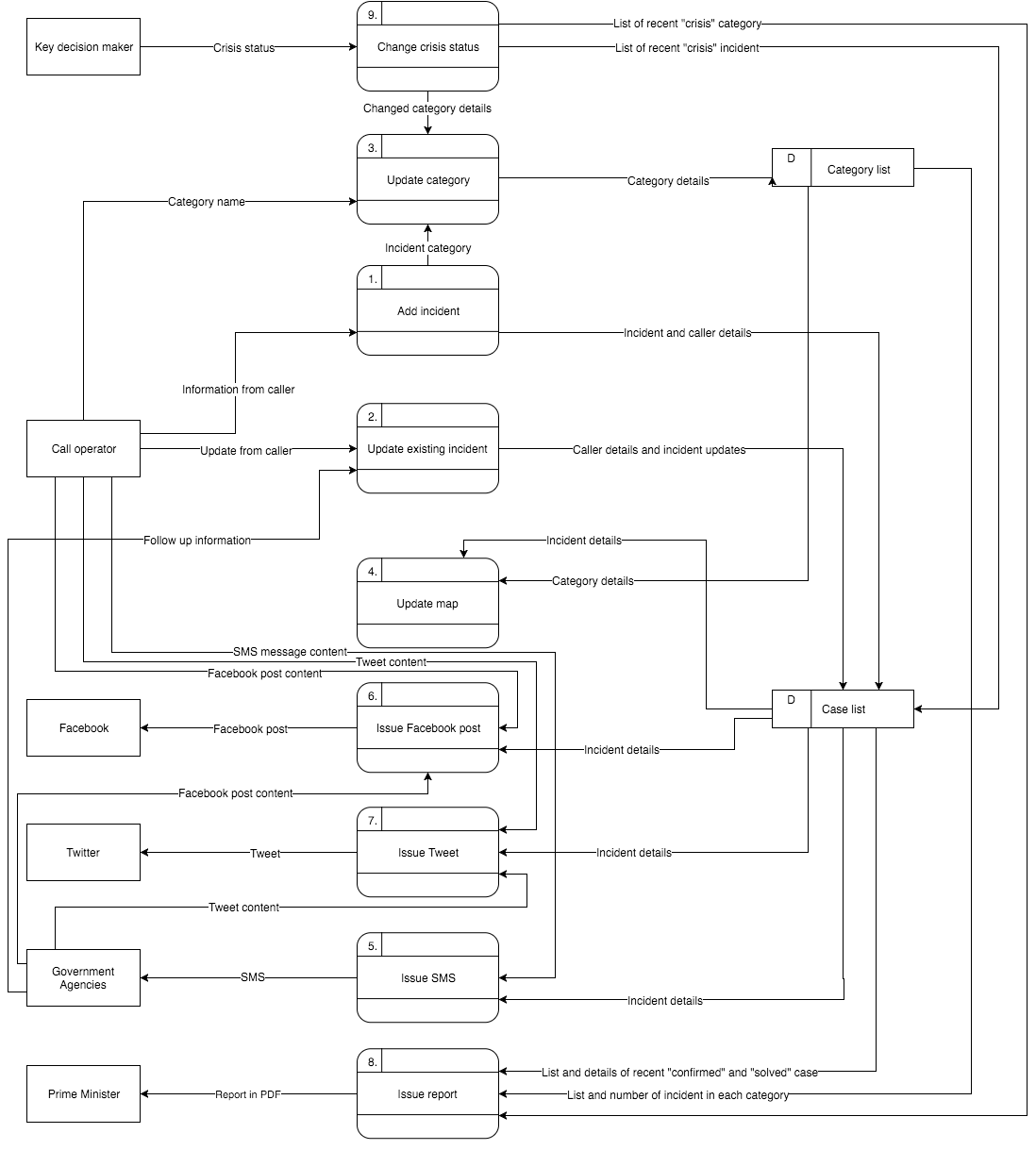
### Class Diagram



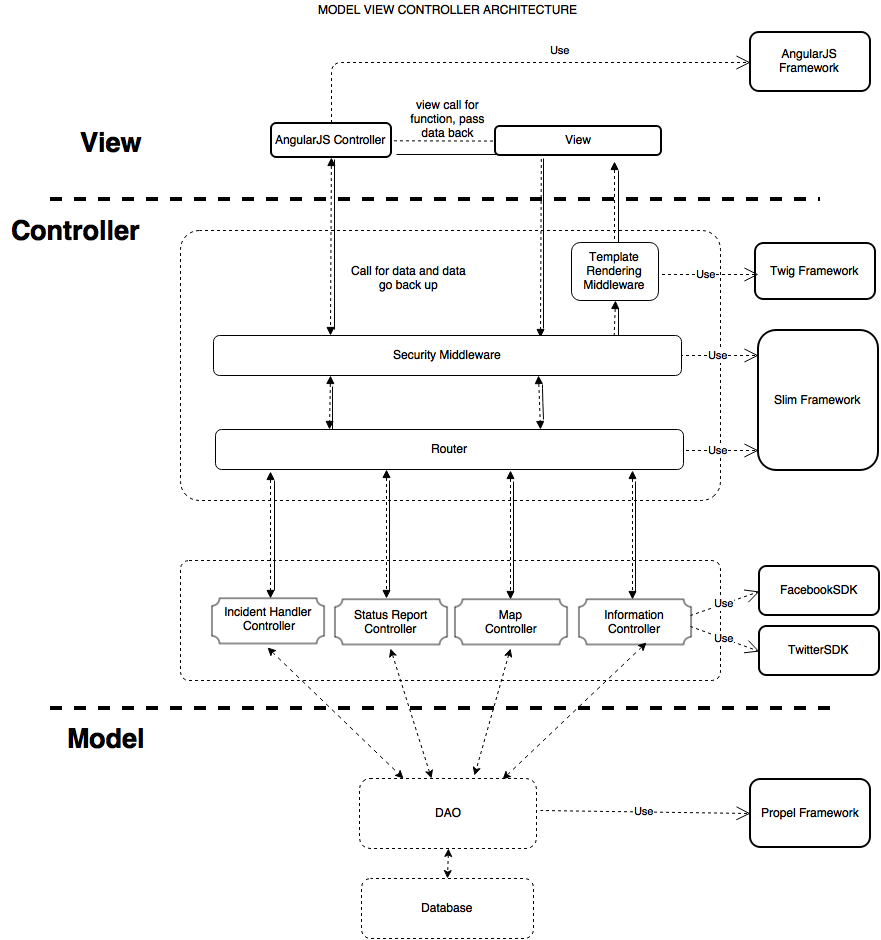
### Context Diagram



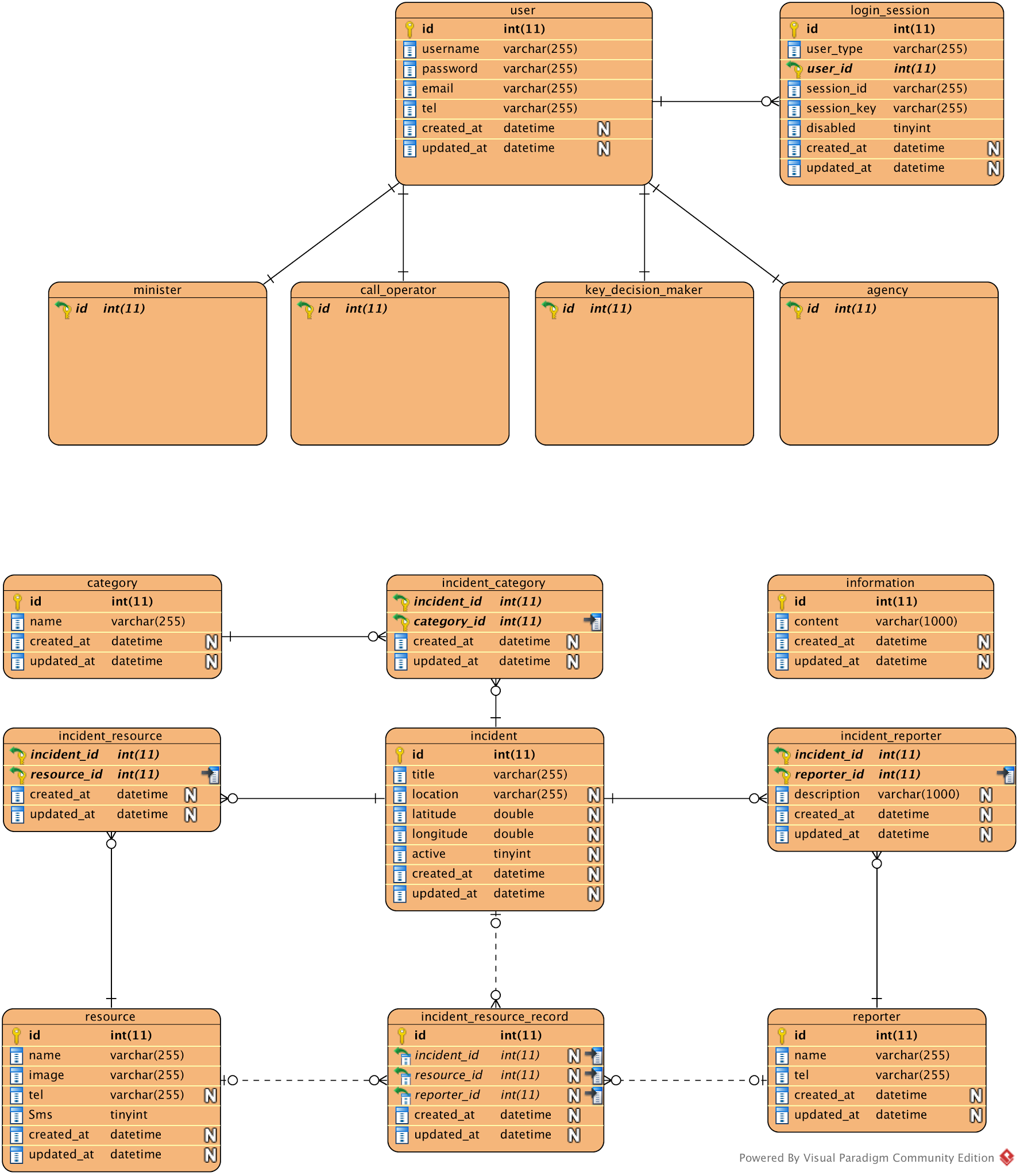
### Data Flow Diagram



### Architectural Diagram



### Entity-Relation Diagram

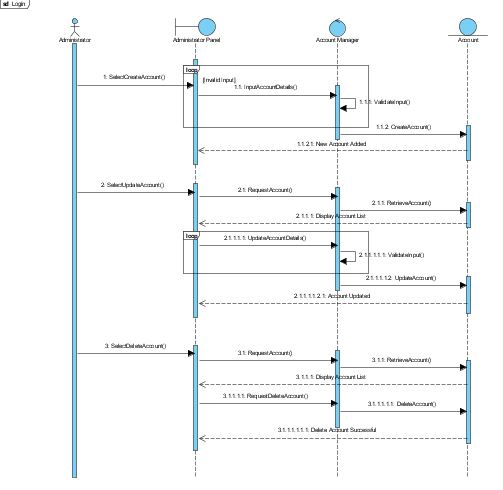


### Component Diagram

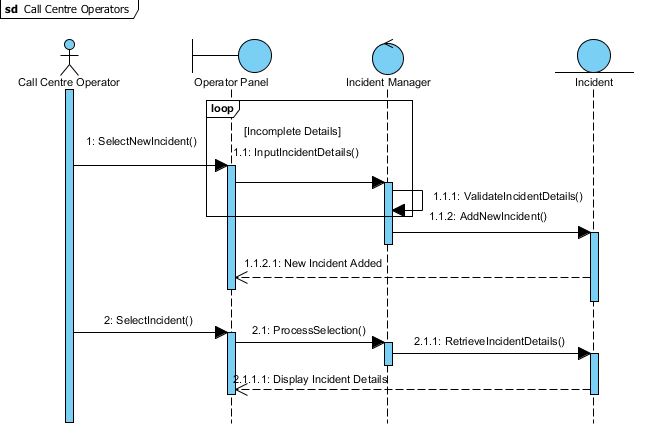


### Sequence Diagram

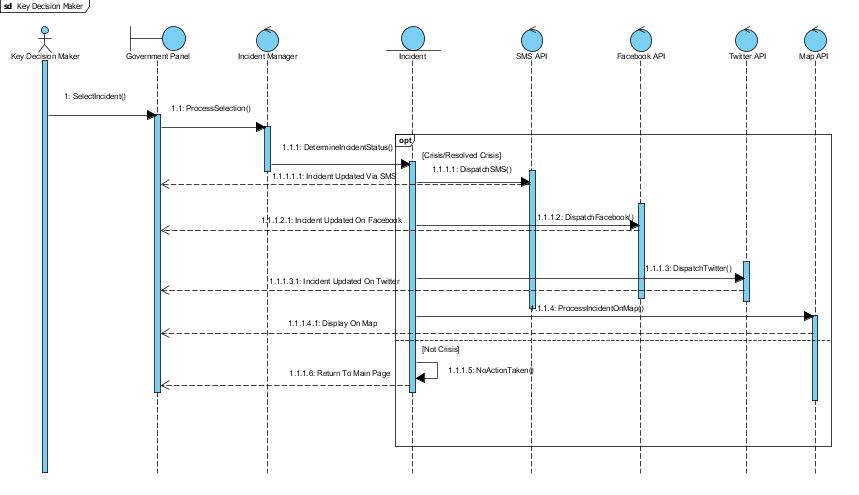
**Login**



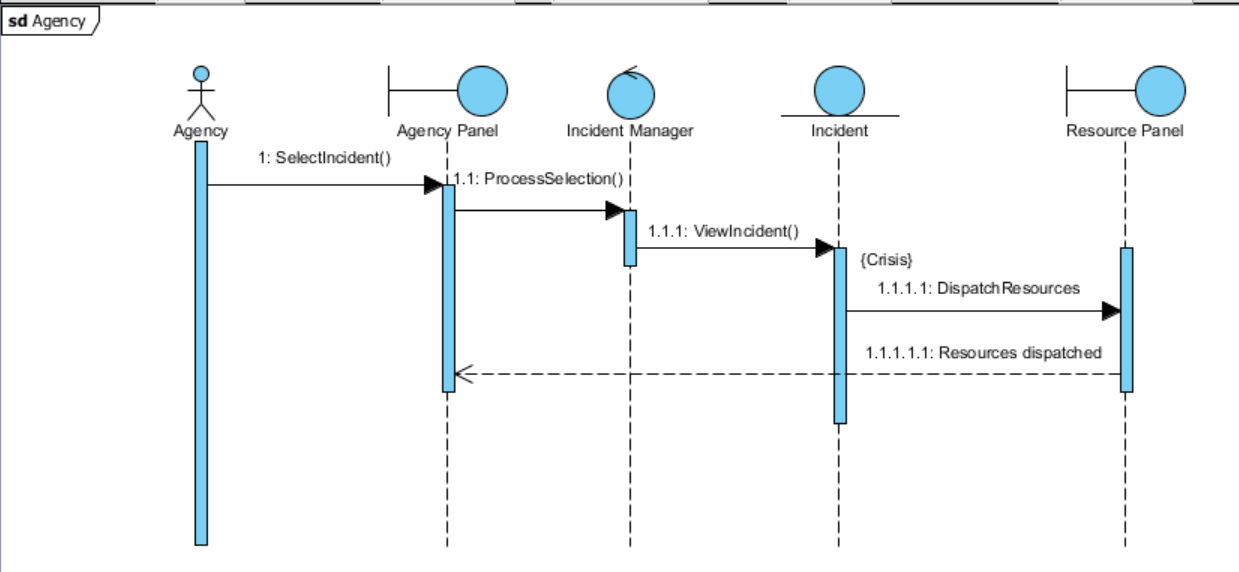
**Call Centre Operator**



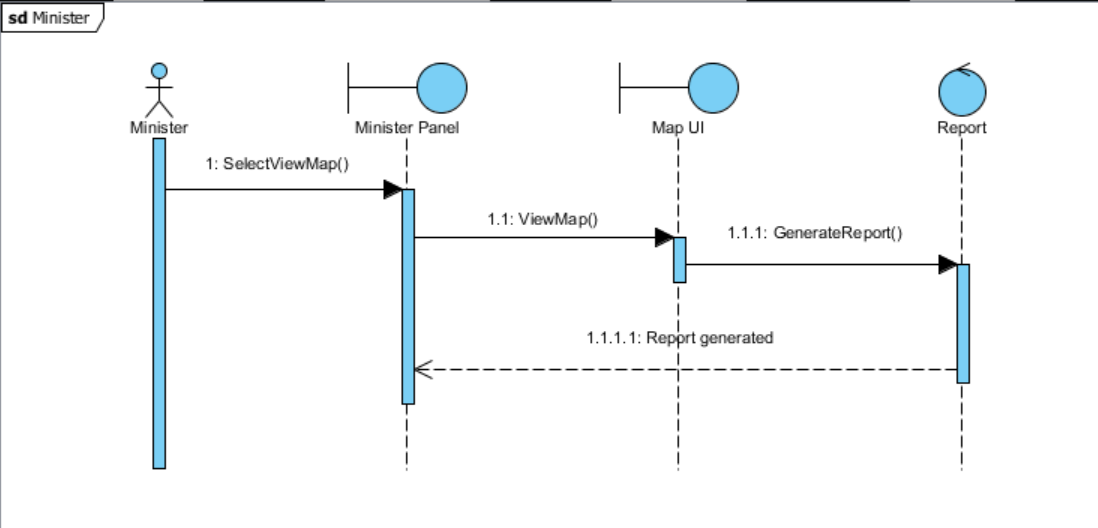
**Key Decision Maker**



**Agency**



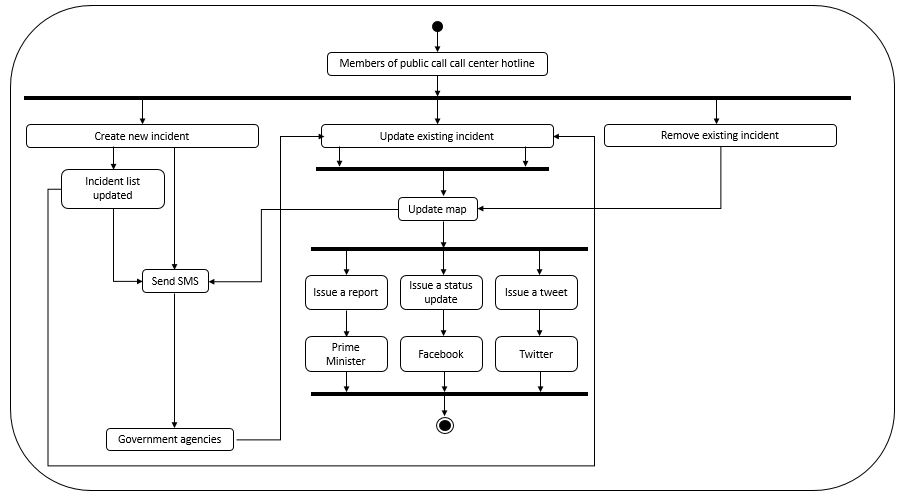
**Minister**



### State Transition Diagram

Dialog Map .png

### Activity Diagram



## Assumptions and Dependencies

1. Agencies must dispatch immediately upon receiving the dispatchment request through SMS.
2. Agencies must update the incident based on real-world situation.
3. Google Map API must be available to provide latest map information.
4. NEA API must be available to provide latest PSI and weather information.
5. Dengue Singapore API must be available to provide latest dengue information.
6. Twitter and Facebook API must be online to post incident and crisis information.

SMS API gateway must be available to send dispatchment request.



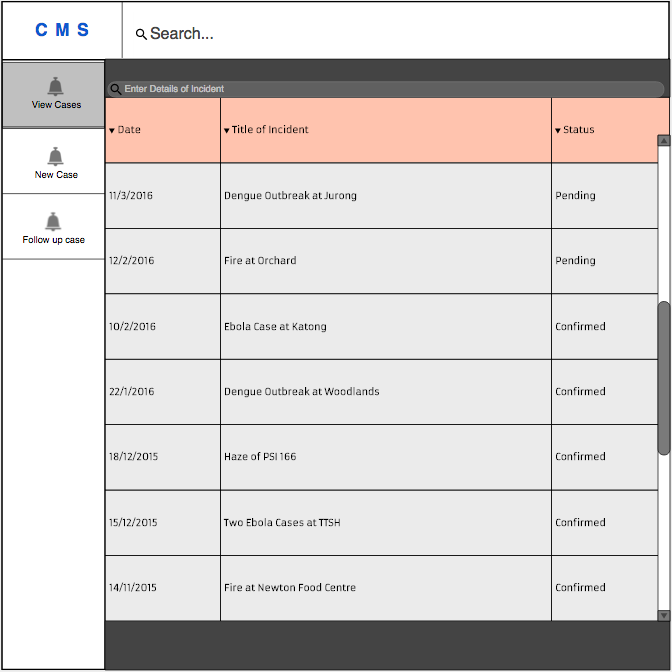
# External Interface Requirements

## User Interfaces

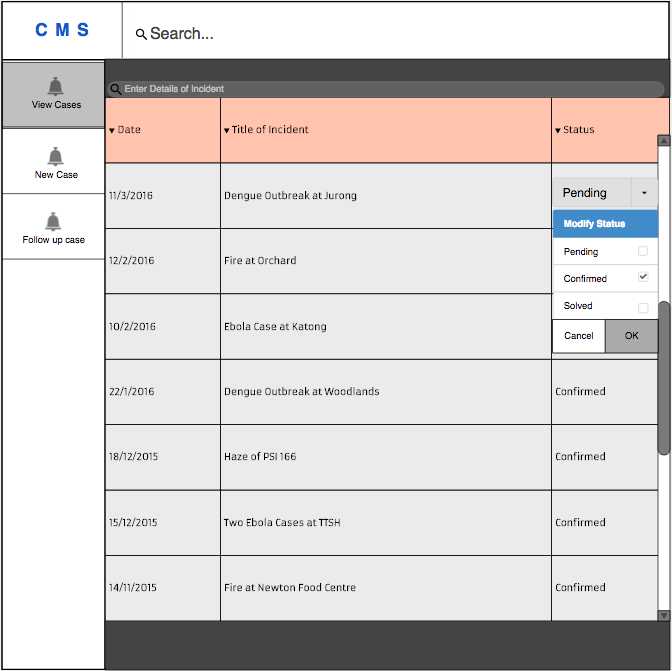
### Map View

### Map View - Crisis Overview

### Incident List

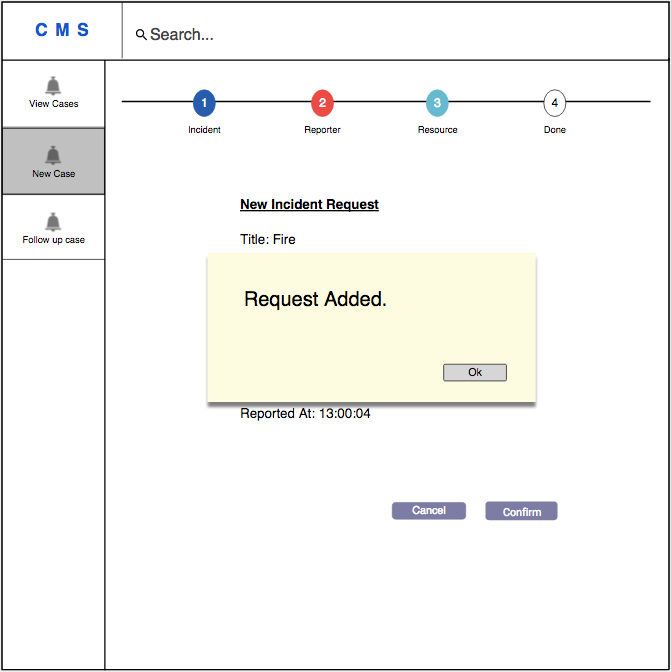


### Incident Status Update

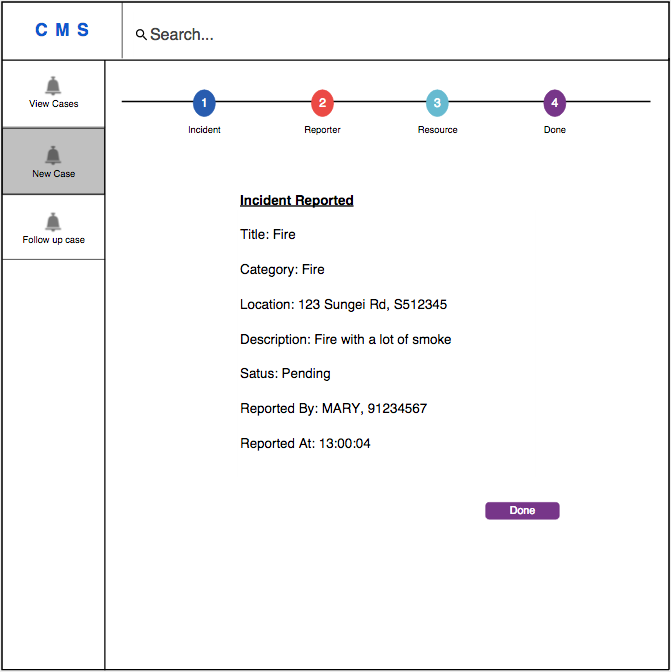


### Incident Submission Form

### Request Form



### Incident Summary



## Hardware Interfaces

The Hardware Interfaces of the system are handled by the Ubuntu with Apache 2. No hardware dependent code will be written.

## Software Interfaces

1. Operating System
   1. The software is being designed to run on most of the operating systems namely Ubuntu, CentOS and Windows Server.
2. Web Server
   1. The software is being designed to run on Apache version 2.4.18.
   2. The software is being designed to run with PHP module version 5.5.
3. Database
   1. The software will access the MySQL database for the following features.
      1. Manage incidents.
      2. Generating status report.
      3. Making request to resources.
      4. Sending useful information periodically.
4. Libraries
   1. The software will be created using the Slim framework version 3 for routing and middleware control.
   2. The software will be created using the Propel Framework version 2.0 for Object-oriented realization.
   3. The software will be created using the jQuery Framework version 2.2.1 for web client animation.
   4. The software will be created using the Google Map Javascript API for map rendering and geocoding services.
   5. The software will be created using the AngularJS framework for 2-way data binding.
5. Page Layout Tools
   1. The software will be created using the Twig Framework version 1.24.0 for template rendering.

## Communication Interfaces

1. Web Interface
   1. The application will be accessed over the Internet. All features will accessible through the web site.
2. Request SMS
   1. Call operators, Agency Staff and Key Decision Makers can send request to other government agencies for help and assistance. Within the SMS, the incident’s location and description will be included.

# System Features

## Functional Requirement

1. The CMS must provide real-time status updates on the map of Singapore by displaying the latest incidents in different ways based on the severity of the incident.
   1. The CMS must display the haze information in PSI.
      1. The data of PSI in Singapore is obtained through NEA API.
   2. The CMS must display current date and time of Singapore.
   3. The CMS must display an up-to date map of Singapore.
      1. The map must be obtained through Google Map API.
2. The CMS must provide a platform for the call centre operators to enter the requested information provided by the caller.
   1. The CMS must allow the call centre operators to add multiple report for the same incident by referencing to the list.
3. The CMS must display the real-time updates on the incidents’ information.
   1. The incidents’ information including the total number of report submitted, status, severity, and location.
      1. The map will show the current total number of report submitted for each incident.
      2. The total number of report for each incident consists of every report that each call operator get for the particular incident.
      3. Each distinct incident is shown by a marker on its location in the map.
      4. The severity is displayed in rating form.
      5. The rating of severity ranges from “A” being the most severe incident, and “E” being the least severe incident, is indicated on the marker.
      6. Each type of severity is denoted by its respective color code, described as follows:
         * “A” : Red
         * “B” : Yellow
         * “C” : Blue
         * “D” : Green
         * “E” : Violet
      7. Incidents can be grouped by category, and in the form of “active” and “inactive”.
      8. Severity of the incidents in the rating form can only be set when the status is “active”
   2. The CMS must display the detailed information of the selected incident on the map.
4. The CMS must allow the call centre operators to update the details of each distinct incident.
5. The CMS must allow the call centre operators to add and edit the category of incidents.
   1. Category is determined based on the type of problem reported and assistance required.
6. The CMS must allow the agencies to change the incident status.
7. The CMS must allow the agencies to input follow up information to the respective incident.
8. The CMS must allow the call centre operators and agencies to request dispatchment of government personnel.
9. The CMS must allow the key decision makers to declare “crisis” situation for a category of incidents.
   1. The CMS must allow the key decision makers to deactivate the “crisis” declaration at any time after the “crisis” has been declared.
10. The CMS must provide updates to the public through SMS and social media, such as Facebook and Twitter, once the incident has been categorised as a crisis or resolved.
11. The CMS must send a status report summarizing key indicators and trends over email to the Prime Minister’s Office every 30 minutes
    1. The report must be sent in the form of PDF document.
    2. The CMS must send an immediate status report when a category is declared as “crisis”.
    3. The CMS must send an immediate status report when a “crisis” is resolved.
12. The CMS must send an SMS to relevant agencies immediately when an incident is reported.
13. The CMS must serve as a command-and-control platform to respond quickly to needs and dangerous conditions, such as gas leaks, hazardous air condition and fires.
14. The CMS shall provide useful information such as location of Civil Defence shelters and updates to the public through social media (SMS, Facebook and Twitter) periodically.

## Software Requirements

1. Web application shall be viewable on:
   1. Google Chrome v46
   2. Safari v7
   3. Firefox v42
2. Web application server shall be working on Apache v2.14.
3. Database server shall be working on MySQL v5.6
4. Development of code shall be done with PHP 5.6
   1. **User Interface Requirements**
5. Interface must be consistent throughout the different web browser.
6. Color theme and layout of the interface must be consistent throughout all pages available.
7. Color theme and layout of the interface must not use more than 4 main colors.
8. Font size must be at least the same size as the browser’s default font size.
9. User must be able to move from section to section at any point in time, using the sidebar.
10. Logout button must always be available at the sidebar.
11. Error messages must be displayed immediately.
12. The content of error messages must be clear and concise.

The content of error messages must contains the error type.

# Other Non-functional Requirements

## Performance Requirements

1. SMS to and from the relevant agencies must be sent within 5 seconds.
2. Twitter and Facebook updates must be viewable within 5 seconds.
3. Incident report file must be constructed and sent within 20 seconds.
4. The map display must be updated periodically after every report submission.
5. The map update process must be done within 10 seconds.

## Security Requirements

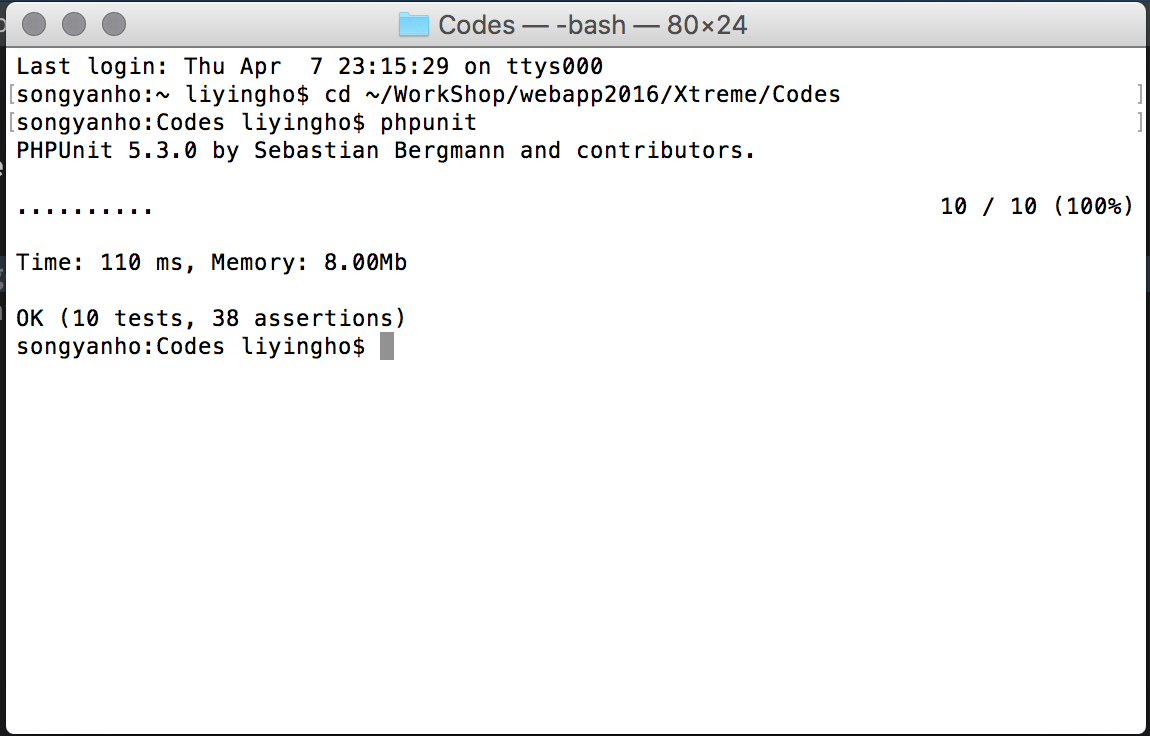
1. The CMS must protect against unauthorized access by requesting login information and user authentication.
2. The CMS must provide additional blockage to some part of its application based on the current user type.
3. The CMS shall not reveal any personal information regarding any user.
4. The CMS shall give each user an unique identification number.
5. The CMS shall only identify each user by their own identification number.

## Software Quality Attributes

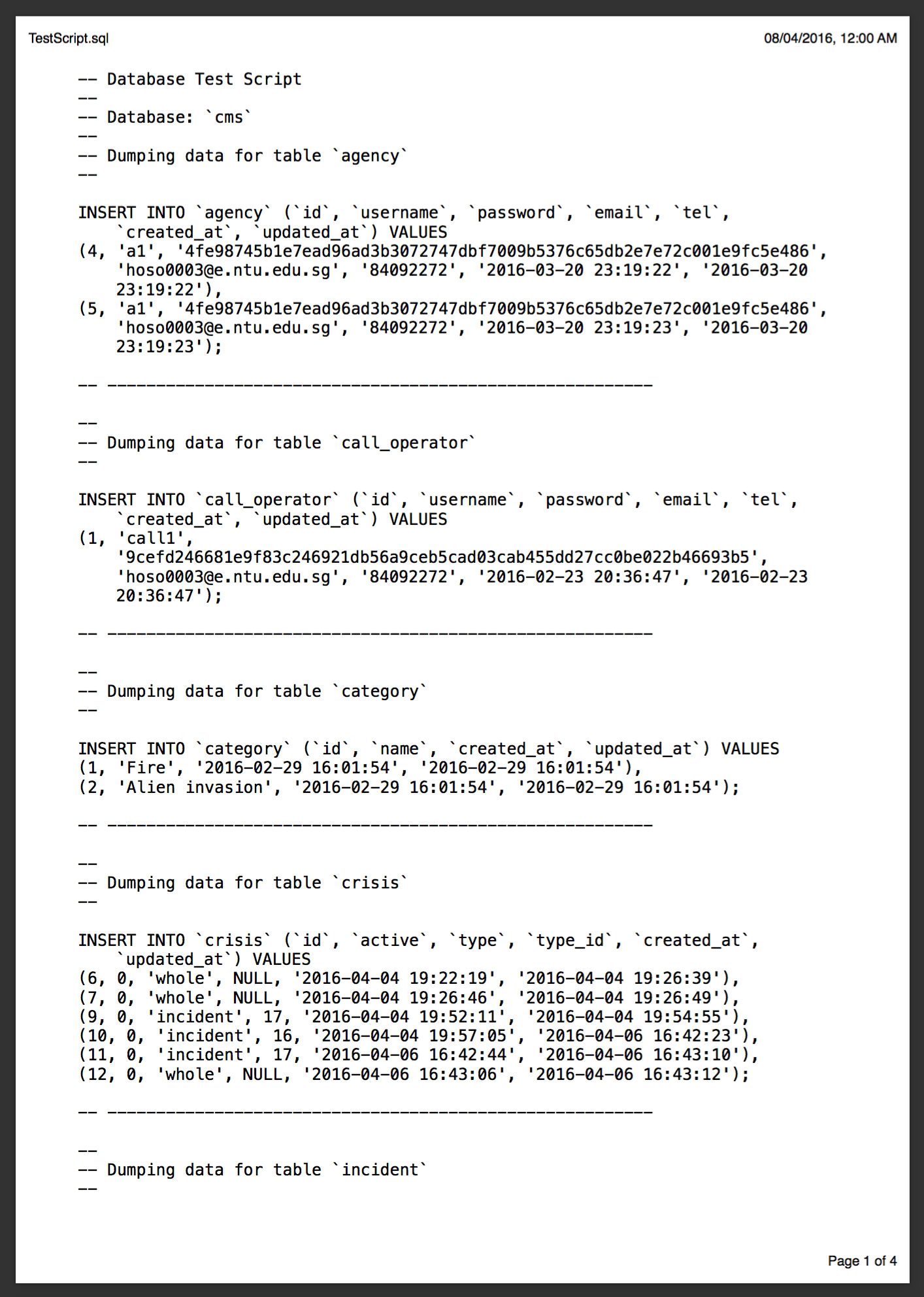
1. Availability
   1. The CMS must be available 99% of the time.
2. Correctness
   1. The CMS UI must be clear and clean enough for the user to understand each of the component function.
   2. The CMS must handle errors made by the user.
   3. The CMS must inform user for any error occurred.
3. Efficiency
   1. The system shall alert the members of the public within 1 minute when a crisis occurs.
   2. The system must be able to start up within 5 seconds
4. Integrity
   1. The system shall protect against unauthorized access by blocking some of the features for some user type.
   2. The system shall request and verify id and password at login page.
5. Portability
   1. The system must be available on multiple web browsers, such as Firefox, Safari, and Google Chrome.
6. Reliability
   1. The system must have proper data management.
      1. The system must retrieve data correctly.
      2. The system must store data correctly.
      3. Error checking must be done by the system.
      4. Exception handling must be done by the system.
      5. The system must ensure data consistency.
   2. The system must have an success rate of 99% as the system deals with life critical situations.
      1. The system must be restored within 1 minute if it crashes.
      2. The system must recover from any down time in less than 5 minutes.
7. Robustness
   1. The system must be able to recover from cyber attack within 1 minute.
   2. The system must be able to handle up to 1,000 users.
8. Usability
   1. The system interface shall use English as its main language, as it is an international language.
   2. User that has a basic proficiency in English shall be able to operate 90% of the system’s function.
9. Safety
   1. The system must be restored within 2 minutes after failure.

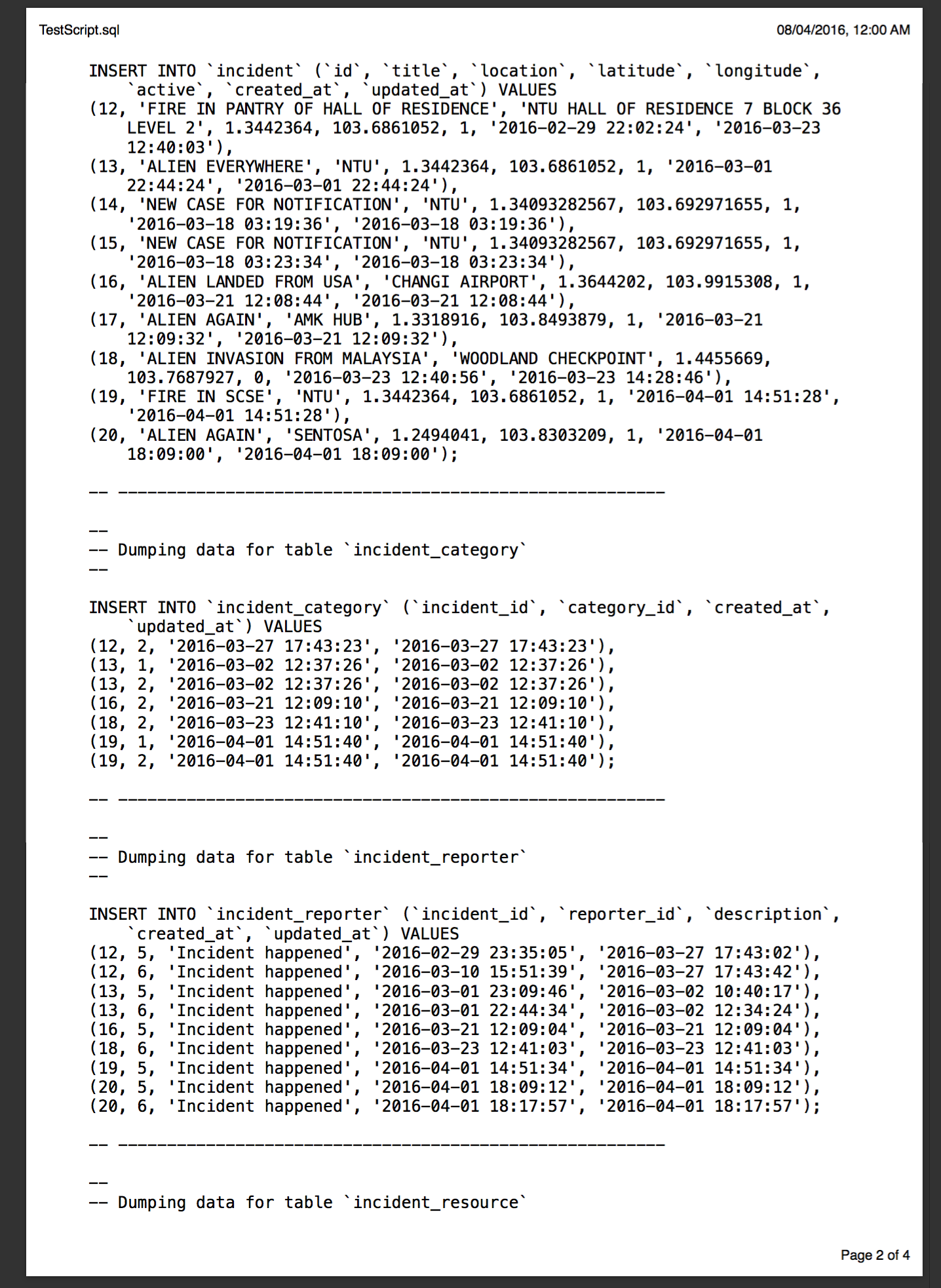
# Testing

## Stub Driver Test

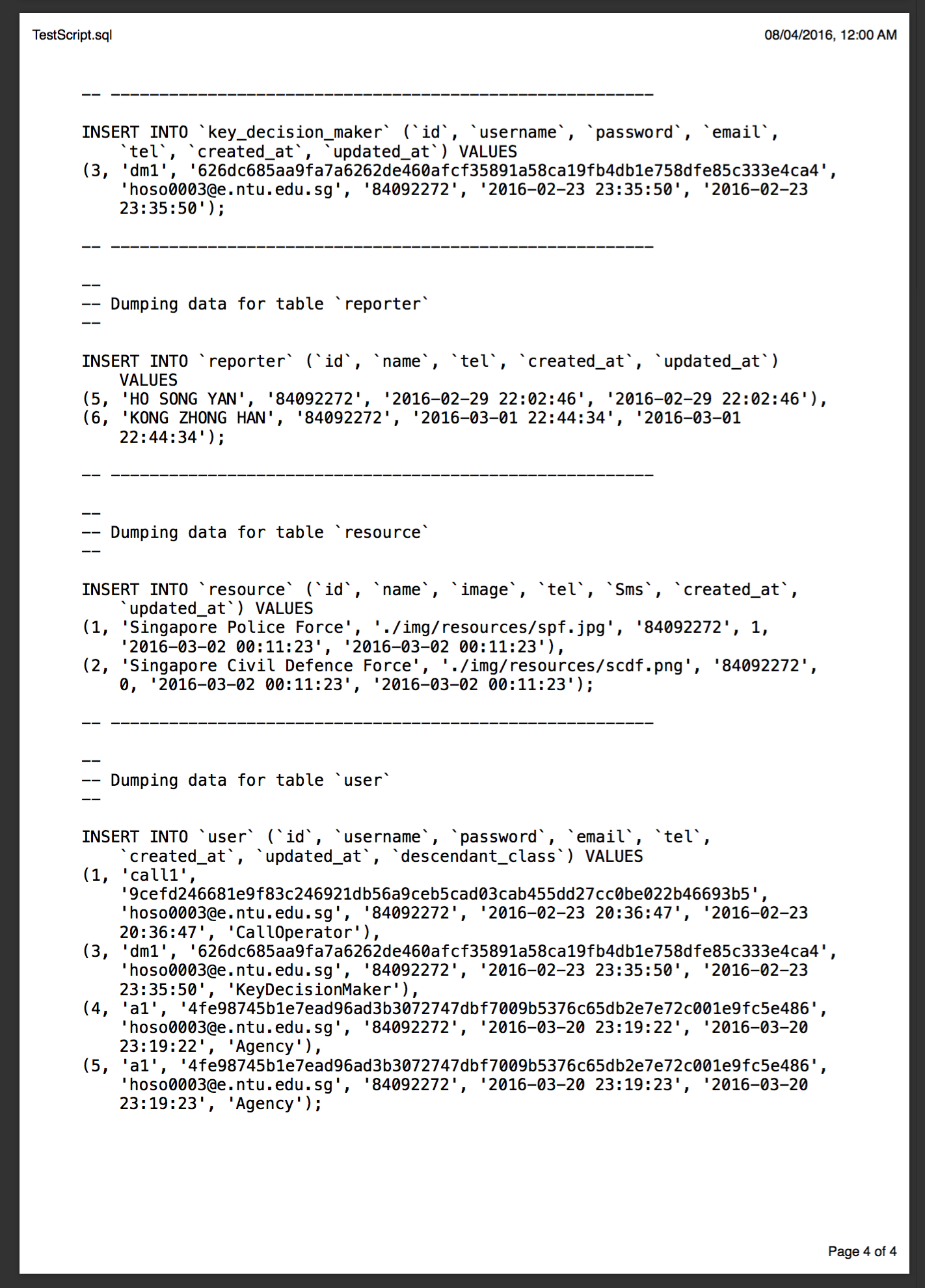


## Database Script

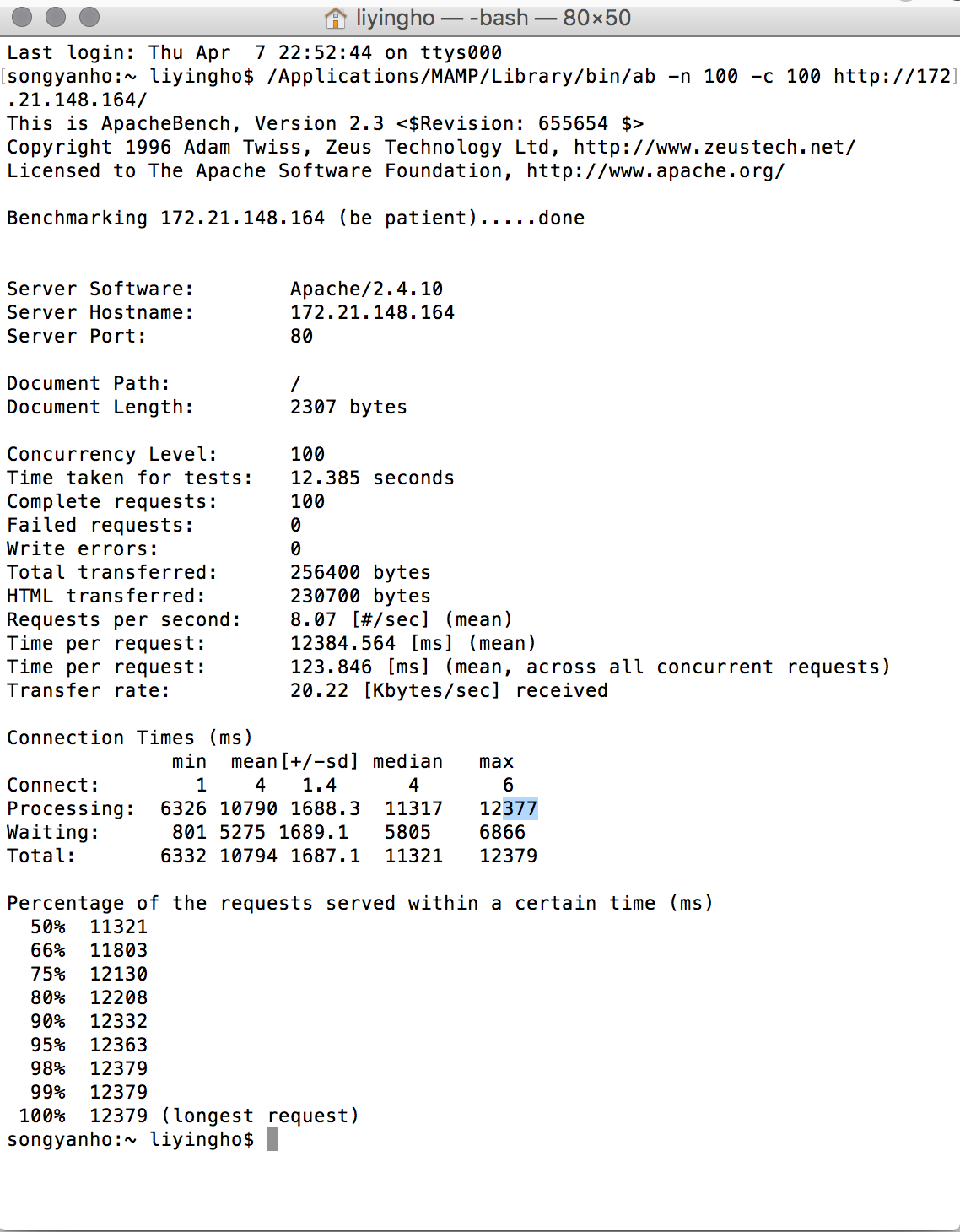








## Load Test by ApacheBench



## Testing strategies and techniques

Several test strategies are planned and carried out during the development of software application, namely unit test, integration test and validation test. First of all, unit test allows developers to identifies bugs or abnormalities efficiently and effectively. Besides that, it demonstrates a concrete development progress and unit test may only involve small part of codes to ensure the correctness of result for particular situation.

In order to ensure an automated, fair and accurate test results, several tools are implemented in software development lifecycle, namely PHPUnit and ApacheBench. PHPUnit is a PHP testing framework mostly for unit testing. The testing is automated and driven by command in command line interface(CLI). It saves time for developers to test the basic functionalities after merging of software components and modules. Furthermore, ApacheBench is implemented to carry out load test or stress test to the software application. It is a tool provided by Apache community to simulate a large number of requests by several concurrent parties. It provides data on the time taken for a request, mean time taken and maximum time taken by a request. It can prove the performance of the software application meets the functional requirements.

## System Level Functional Tests

The functional tests cases will be considered and generated based on the use case diagram done through the analysis of the needs of the software. The following scenarios will be considered:

1. Function: **Generate Status Report**
   1. Test Objective
      1. Ensure that the system is able to generate a report every 30 minutes.
      2. Ensure that the report generated is successfully sent to the Minister’s Office.
      3. The data generated in the report must be updated.
      4. The format of the report generated must be consistent.
   2. Evaluation Criteria
      1. Allow the system to run for a set number of time to allow the system to generate reports of a significant sample size.
      2. Of all the reports which are generated, compare and ensure that the intervals is 30 minutes.
      3. Ensure that all reports which were generated was successfully sent to the Minister’s Office.
      4. Ensure that updated data which are keyed into the system while the functional test is being conducted is shown in the report.
      5. Ensure that all reports are presented in a consistent format.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Pre-condition** | **Steps** | **Test Data** | **Expected Results** | **Post Condition** |
| #01 | Successful Generation of a report every 30 minutes | N.A. | 1. Run the system for 30 minutes | 1. Input of new incident every 30 minutes | The new incident which is created will be shown in the report | The Report is sent to the Minister’s Office |
| #02 | Report Successfully Sent to the Minister’s Office | Successful Generation of a report | 1. Run the system for 30 minutes | N.A. | The report is sent to the Minister’s Office | N.A. |
| #03 | Data Generated in the report must be updated | Successful Generation of a report | 1. Run the system for 2 hours | 1. Input of new incident every 30 minutes | Every report will have any pre-existing incidents, with reports on new incident included into the report. | N.A. |
| #04 | Format of the Report generated must be consistent | Successful Generation of a report | 1. Run the system for 2 hours | 1. Input of new incident every 30 minutes | Details of an incident in every report must be displayed in a consistent manner. | N.A. |

1. Function: **Provide Real-time map View**
   1. Test Objective
      1. Ensure that new incidents which are inputted into the system is updated on the map in real time.
      2. Ensure that incidents which have been closed will be removed from the map in real time.
   2. Evaluation Criteria
      1. Ensure that new incident inputs by call operator will be reflected on Key Decision Maker UI
      2. Ensure that existing incidents which have been updated or closed by the Agency or Call Operators will be reflected on Key Decision Maker UI.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Pre-condition** | **Steps** | **Test Data** | **Expected Results** | **Post Condition** |
| #01 | Ensure that new incidents which are inputted into the system is updated on the map in real time. | Call Operator and Key Decision Maker is logged in to the system. | 1. Call Operator creates a new incident into the CMS | 1. Details of a new incident | A marker is placed on the map where the new incident is taking place.  When the call operator clicks on the marker he will be able to see more information about the incident. | Agency is informed on the situation via SMS. |
| #02 | Ensure that incidents which have been closed will be removed from the map in real time | 1. Call operator and Key Decision maker is logged into the system. 2. There must be an existing active incident in the system. | 1. Call Operator follows up on an existing active case. 2. Call Operator closes the case. | 1. Close the case. | The marker of the existing active case which was shown on the map on the UI of Key Decision Maker would be removed in real time. | N.A. |

1. Function: **Case Analysis and Declare Crisis**
   1. Test Objective
      1. The Key Decision Maker will be able to view statistics of active incidents by its category.
      2. The Key Decision Maker will be able to declare crisis
         * Declare an incident to be a crisis.
         * Declare a Category of incident to be a crisis.
         * Declare the country to be in Crisis.
      3. A Facebook post will be generated and sent once a crisis has been declared.
      4. A Tweet will be generated and sent once a crisis has been declared.
      5. SMS will be generated and sent once a crisis has been declared.
   2. Evaluation Criteria
      1. A sidebar tab allows the Key Decision Maker to view details and statistics of active incidents by its category
      2. Changes will be reflected on the UI once a crisis has been declared.
      3. Ensure that a Facebook post will be generated and sent 5 second after a crisis has been declared.
      4. Ensure that a tweet will be generated and sent 5 second after a crisis has been declared.
      5. Ensure that a SMS will be generated and sent 5 second after a crisis has been declared.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Pre-condition** | **Steps** | **Test Data** | **Expected Results** | **Post Condition** |
| #01 | The Key Decision Maker will be able to view statistics of active incidents by its category | Existing active incidents of different category are in the system. | 1. The Key Decision Maker chooses to view the incidents. | N.A. | Incidents would be shown according to the different category, and within it grouped according to the different severity of an incident. | N.A. |
| #02 | The Key Decision Maker will be able to declare an incident to be a crisis | Existing active incidents are in the system. | 1. The Key Decision Maker views the incidents 2. An incident which is to be declared a crisis would be shown 3. The Key decision maker would activate the incident to be a crisis. | N.A. | The system will react and flash red. | Information will be sent to the public via Facebook, Twitter and SMS. |
| #03 | The Key Decision Maker will be able to declare a category of incidents to be a crisis. | Existing active incidents are in the system. | 1. The Key Decision Maker determines the category of incident to be a crisis. 2. On the map UI of key decision maker, toggle the switch of the category to declare it a crisis. | N.A. | The system will react and flash red. | Information will be sent to the public via Facebook, Twitter and SMS. |
| #04 | The Key Decision Maker will be able to declare the country to be in crisis.. | Existing active incidents are in the system. | 1. The Key Decision Maker toggles the switch on the map UI to declare the country to be in crisis. | N.A. | The system will react and flash red. | Information will be sent to the public via Facebook, Twitter and SMS. |
| #05 | A Facebook post will be generated and sent once a crisis has been declared. | A crisis has been declared | N.A. | N.A. | A Facebook post with detail on the crisis will be generated and posted on Facebook. | N.A. |
| #06 | A Tweet will be generated and sent once a crisis has been declared. | A crisis has been declared | N.A. | N.A. | A Tweet with details on the crisis will be generated and posted on Twitter. | N.A. |
| #07 | An SMS will be generated and sent out once a crisis has been declared. | A crisis has been declared | N.A. | N.A. | An SMS with details on the crisis will be generated and posted on Twitter | N.A. |

1. Function: **Make Request and Dispatch Resource**
   1. Test Objective
      1. Ensure that Call Operators are able to dispatch agency resources successfully
      2. Ensure that once an agency resource has been dispatched by the call operator, an SMS will be sent to the Agency.
   2. Evaluation Criteria
      1. Call Operators must be able to dispatch agency resources before the incident is being submitted.
      2. Once an incident has been submitted with the request to dispatch agency resources, the system must be able to send an SMS to the agency.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Pre-condition** | **Steps** | **Test Data** | **Expected Results** | **Post Condition** |
| #01 | Ensure that Call Operators are able to dispatch agency resources successfully |  | 1. The call operator chooses the relevant agency to dispatch a resource to a new incident 2. The call operator confirms and add the incident into the system. | Details on a new incident. | A new incident would be created.  The details of the source dispatched would be shown on the incident details on the UI of call operator and Key Decision Maker | A new incident would be created and reflected on the map in real time. |
| #02 | Ensure that once an agency resource has been dispatched by the call operator, an SMS will be generated | A new incident has been successfully created.  The call operator chose a relevant agency during the request. | N.A. | N.A. | An SMS with the relevant information for the case would be sent to the agency for them to react accordingly | Agency will be able to request for more resources or update the case once they have attended to the incident. |

1. Function: **Create and Manage Case**
   1. Test Objective
      1. Ensure that Call Operators are able to create new incidents.
      2. Ensure that Call Operators are able to manage existing incidents.
   2. Evaluation Criteria
      1. Once a call operator has made a set of new incidents, all incidents are correctly stored by the system.
      2. Call Operator must be able to successfully retrieve a pre-existing incident which a Call Operator previously have created.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Pre-condition** | **Steps** | **Test Data** | **Expected Results** | **Post Condition** |
| #01 | Ensure that Call Operators are able to create new incidents | The call operator have details about the new incident | 1. Call Operator creates new case 2. Call Operator inputs the relevant details of the incident 3. Call operator submits the incident. | Details of a new incident. | The call operator successfully add a new incident into the system. | An SMS will be sent to the agency if a request is sent.  The marker of the incident will be reflected on the map of the Key Decision Maker in real time. |
| #02 | Ensure that Call Operators are able to manage existing incidents | The call operator has new details to update the incident.  There is an existing incident for the call operator to update on | 1. Call operator finds the incident to update 2. Call operator updates the incident with the new information 3. Call operator confirms the change. | Details to update an existing incident on | The Call operator successfully modifies the details of an incident. | An SMS will be sent to the agency if more request for resources is required.  The marker of the incident will be updated and reflected on the map of the Key Decision Maker in real time. |

1. Function: **Run Security Validation**
   1. Test Objective
      1. Ensure that only users which have the correct authentication is able to access the various UI.
      2. Ensure that only users which have the correct authentication is able to perform certain actions.
   2. Evaluation Criteria
      1. If the user is able to access a part of the system which he/she is not authenticated to, system fails.
      2. If the user is able to perform an action which he/she is not authenticated to, system fails.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Pre-condition** | **Steps** | **Test Data** | **Expected Results** | **Post Condition** |
| #01 | Ensure that only users  which have the correct authentication is able to access the various UI | The user is logged into the system. | The call operator attempts to access the UI of a Key Decision maker. | N.A. | Call operator is unable to find the button to the link.  OR  An error showing that the user does not have the authentication to access the page. | N.A. |
| #02 | Ensure that only users which have the correct authentication is able to perform certain actions | The user is logged into the system. | The call operator attempts to declare the country to be in crisis | N.A. | Call Operator is unable to find the button to declare a crisis.  OR  An error showing that the user does not have the authentication to declare the country to be in crisis. | N.A. |

1. Function: **Provide Useful Information**
   1. Test Objective
      1. To ensure that the system is able to provide the public with useful information periodically
      2. Ensure that the information provided is updated
      3. Ensure that the information provided is accurate
      4. Ensure that the information provided is consistent within the public platforms (Facebook, Twitter, SMS)
   2. Evaluation Criteria
      1. System to be run for a set amount of time. Useful information of any relevant cases should be made available to public users through Facebook, Twitter and SMS.
      2. An analysis of the informations provided to the public should be done to ensure that all information provided is updated.
      3. An analysis of the informations provided to the public should be done to ensure that all information provided is accurate
      4. Contents between the platforms (Facebook, Twitter, SMS) should be done, where information which are provided at the same time should be checked for consistency.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Objective** | **Pre-condition** | **Steps** | **Test Data** | **Expected Results** | **Post Condition** |
| #01 | Ensure that the system is able to provide the public with useful information periodically. | There are existing active incidents in the system. | N.A. | Details of existing active incidents in the system. | The details are successfully retrieved and sent via Facebook, Twitter and SMS | N.A. |
| #02 | Ensure that the information provided is updated | New incidents are added into the system periodically. | A new incident is added into the system every 30 minutes | Details of new incidents | Details of the new incidents will be reflected on Facebook, Twitter and SMS. | N.A. |
| #03 | Ensure that the information provided is accurate | New incidents are added into the system periodically | A new incident is added into the system every 30 minutes | Details of new incidents | True Details which only belongs to the new incidents will be reflected on Facebook, Twitter and SMS. | N.A. |
| #04 | Ensure that the information provided is consistent within the public platforms (Facebook, Twitter, SMS) | Information sent out to the public platforms are successful | N.A. | Details of existing active incidents in the system. | Format and details provided between the different platforms are same and consistent. | N.A. |

# Recommendations

The recommendations that our team would like to suggest to the system that has been designed is to not only consider one of the non-functional requirements but instead excels on other quality attribute. Though it is difficult to focus all the quality attributes, more quality attributes can be satisfied as some of them have direct relationships. Social media platforms such as Instagram can be also used to update the public with research showing that it has become the second most popular social media networks in Singapore. Furthermore, the system should be able to open on more browsers which allows more interoperability, increasing the efficiency of the CMS.

Conclusion

# 

# Appendix A: Glossary

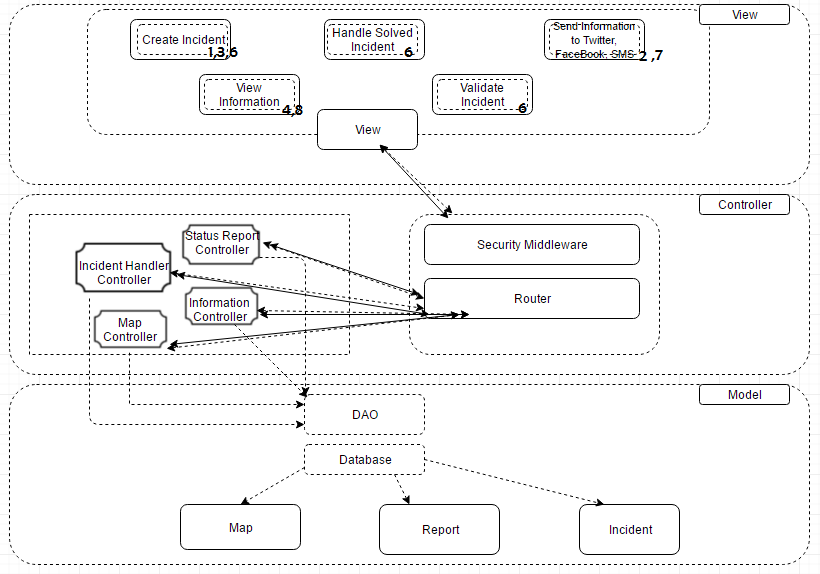
## Data Dictionary

|  |  |
| --- | --- |
| **Terms** | **Description** |
| Assistance type | 4 types of assistance request:   * Emergency Ambulance * Rescue and Evacuation * Fire-Fighting * Gas Leak Control |
| Call operator | In charge of handling calls made by the members of public for assistance and incident reporting, by adding, updating and removing incidents. |
| Category | Overview of the map includes:   * Haze crisis * Dengue crisis |
| CMS | An abbreviation of Crisis Management System. |
| Dangerous conditions | Dangerous conditions which are viruses:   * Ebola * SARS (Severe Acute Respiratory Syndrome) * Zika |
| Government | * Prime Minister * Cabinet ministers * Government agency key decision makers |
| Identification number | Identity for operator |
| Idle | The lack of user input or interaction with the system. |
| Information request | Information needed from members of the public:   * Name * Mobile number * Location (postal code and building unit number, if applicable) * Type of assistance requested |
| PSI | An abbreviation of Pollutant Standards Index which determine whether the air pollution levels in a particular location are good, unhealthy, hazardous or worse. |
| Real-time | Updates are refreshed instantaneously upon data changes. |
| Region | 5 regions on the overview of the map. North, South, East, West and Central. |
| Relevant agencies | * Emergency Ambulance - Singapore Civil Defence Force (SCDF) * Rescue and Evacuation - SCDF * Fire-Fighting - SCDF * Gas Leak Control - Singapore Power |
| Severity | Categorised in terms of ratings from “E” to “A”. |
| Social media | SMS, Facebook and Twitter. |
| Status | Categorised as “Pending”, “Confirmed”, “Crisis” and “Resolved”. |
| Weather conditions | An overall of the atmospheric conditions, that comprise the state of the atmosphere in terms of temperature and wind and clouds and precipitation, is stated on the map with description. |
| Data Consistency | The assurance that the system will be able to identify the most updated data for a specific incident. |

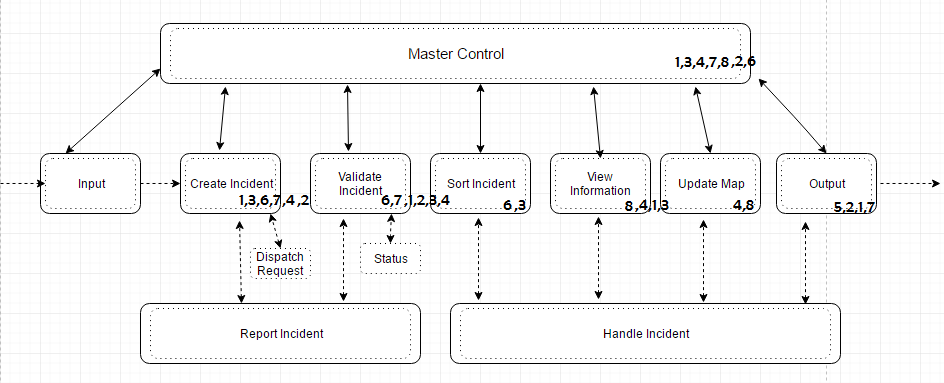
# Appendix B: Analysis Models

## Architecture Design Analysis

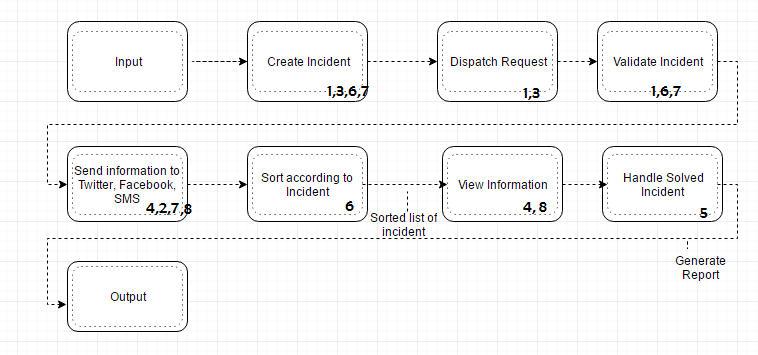
### Layered Architecture



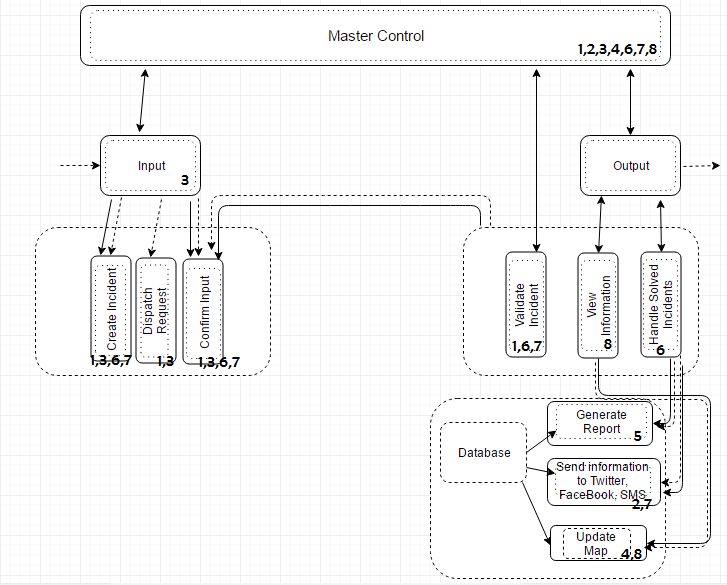
### Main Program/Subroutines with Shared Data



### Pipe and Filter



### Abstract Data Type



## Design Principles

**Modularity**

The architecture design composes of replaceable and self-contained assembly of components, thereby aiding the process of implementation and future maintenance. The Model-View-Controller Layered Architecture divides the large aggregates of components into units, ensuring loose coupling and high internal cohesion between components between layers. Modules within each layer are grouped according to its behaviour and purpose, for example, modules in the controller layer ensures communication is effected through the view and model layers, and demonstrates the concept of abstraction in the design. Each interface is deployed using different frameworks, such as Slim Framework and Angular JS Framework, each handling different interface design details. This allows easy change in the hidden interface design details without affecting its functionality as a system.

**Portability**

The architectural design also allows portability and reuse by employing abstract context interfaces. The application’s interfaces with the platform are separated from the computations and are achieved using a standard model of platform interface. For example, communication APIs such as Facebook, Twitter and SMS are independent of the platform, and can be reused.

**Malleability**

The layered architectural design supports adaptation to changing end-user requirements, and is highly modifiable and flexible. New components can be added in the respective layer without causing interference to the other layers. Also, obsolete components can be removed or updated using this environment.

**Conceptual Integrity**

The Model-View-Controller Layered Architecture is finally selected amongst other design patterns, achieving conceptual integrity with a uniform application of limited number of design forms. This layered architecture is most suited to the system’s requirements and chosen for easier modifiability and assembly of different components. This approach ensures the modules are more independent and parallel and the system does not compromise under circumstance that requires upgrade, removal or insertion of a particular module, thereby offering reliability in the crucial crisis management system.

## Scenario Evaluations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario** | | | **Modification** | |
| **No.** | **Description** | **Type** | **Component** | **Change** |
| 1 | Insertion of new authorised agencies into the system. | Indirect | Create Incident | To change the implementation according to the new agency. |
| Validate Incident |
| 2 | Addition of a new social platform. | Indirect | Send information to Twitter, Facebook, SMS | To add in the respective social media’s SDK and its implementation in the system. |
| 3 | Change the type of outbreak the CMS can handle. | Indirect | Create Incident | To change the implementation according to the type of outbreak added. |
| Input |
| 4 | Change of the current map (due to the updating of the map). | Indirect | Update Map | To change the implementation according to the new map. |
| 5 | Change of the layout of the report generated. | Indirect | Handle Solved Incident | To change the implementation according to the new layout. |
| 6 | Insert new status of the incident (ie, pending, confirmed, crisis, solved) | Indirect | Input | To change the implementation according to the new status. |
| Create Incident |
| Validate Incident |
| Update Map |
| Sort Incident |
| 7 | Include a time period for the operator to inform the general public via social media and SMS | Indirect | Create Incident | To change the implementation according to the new timer. |
| Validate Incident |
| Handle Solved Incident |
| Send information to Twitter, Facebook and SMS |
| 8 | Insert new locations of CD shelters | Indirect | Update Map | To change the implementation according to the new location details. |
| View Information |

## Evaluation of CMS Architectures

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Scenarios** | | **Type of Architecture Pattern** | | | |
| **No.** | **Weight** | **Shared Data** | **Abstract Data** | **Pipe and Filter** | **Layered** |
| **1** | 15 | 5/8 | 4/12 | 3/9 | 1/5 |
| **2** | 5 | 4/8 | 2/12 | 1/9 | 1/5 |
| **3** | 20 | 5/8 | 5/12 | 2/9 | 1/5 |
| **4** | 20 | 5/8 | 2/12 | 2/9 | 1/5 |
| **5** | 5 | 1/8 | 1/12 | 1/9 | 0 |
| **6** | 10 | 4/8 | 5/12 | 3/9 | 3/5 |
| **7** | 15 | 4/8 | 5/12 | 3/9 | 1/5 |
| **8** | 10 | 3/8 | 3/12 | 2/9 | 1/5 |
| **Total (Modification/cost)** | | 53.75 | 30.83 | 25.55 | 23 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Architecture** | **Shared Data** | **Abstract Data** | **Pipe and Filter** | **Layered** |
| Change in data processing algorithm | **-** | **-** | **+** | **+** |
| Change in data representation | **-** | **+** | **+** | **+** |
| Reuse | **-** | **+** | **+** | **+** |

Based on the quantitative approach of evaluating the CMS Architectures, we can see that the Shared Data architecture yield the highest in cost and modification to the program, making it the least desirable architecture pattern to follow due to poor modifiability in handling processing algorithms, data representations and reuse. The most promising architecture is the layered architecture, therefore we have chosen that architecture pattern amongst the rest. Albeit Pipe and Filter features promising results with its use of independent components, the layered architecture will be able to support better enhancement in functionality with the ease of introduction of new components without affecting other layers, making it score better in reliability and maintainability.

The selected candidate architecture follows the pattern of Model-View-Controller(MVC) architecture pattern. The one of the reasons choosing this architecture is MVC design pattern allows the separation of concerns during development phases. MVC design pattern breaks the software components into three main categories, namely model, view and controller. The model is the part of the application that handles the logic for the application data. It formats the data from database into an object by following strict rules of conversion. Besides that, the model formats the incoming data from client side before it is saved into database. It improves the data correctness and reliability. Moreover, the view is presentation of the data that model object contains, whenever the data in the model object is changed, it will generate the view’s presentation of the model based on changes in the model. Besides that, the controller is the part connecting the model and view and control communication between model and view. It controls the data flow into model object and updates the view when there is any change of the data in model object. Thus, it promotes the reusability of business logic and template across the application.

Furthermore, MVC architecture pattern allows parallel development process among team members. MVC architecture pattern suggests the development of business logic, view and model separately and improve the code reusability. It allows all team members to build business logic and design templates independently without affecting other progress and commitment. MVC architecture allows the developers to focus and concentrate on their own part like UI, model or business logic implementations and modifications without bogged down with other developers. And also, the delay caused by user interface design does not affect the progress of business logic design as the processes are carried out independently. Besides that, the minor change in business logic has very less effect to the user interface design, thus it avoids the revision to user interface design as long as the data are formatted according to models.

Moreover, the candidate architecture follows the 3-tier layered architecture pattern. It suggests the separation of processes on different nodes or machines. The candidate architecture splits the application into different layers. The layers suggested are user interface, business logic and data storage and data access which are developed and maintained as independent modules or most often on separate platforms. In order to achieve the candidate architecture in the software development, several frameworks are introduced to the software implementation, namely Slim Framework, Propel ORM, Twig Framework and AngularJS Framework.

First of all, AngularJS Framework takes care of the presentation layer of the software. It allows the presentation layer to work independently in client side. It provides several useful dependencies in data binding, standard REST call and template rendering. Besides that, Slim Framework builds the environment for business logic layer of the software application. It provides several features including middleware control and router which allow the development of software application based on MVC architecture pattern. It allows the software application to work in a separate node or server to provide computation and basic template rendering features. In addition to that, Propel ORM realizes the object-oriented features to PHP application. It builds the data access layer and maintains database according to a designed schema. Besides, it also generates the optimized object-oriented PHP models which speed up the development process of complicated entity-relation design in software application. It handles the modification and changes to the database structure and records the migrations done in organized manner. It helps to maintain the database version and revision.

## Requirements Priority Table

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Feature | Benefit | Penalty | Total Value | **Value %** | Rel. Cost | **Cost %** | Rel. Risk | **Risk %** | **Priority** |
|  | If Present  (1-9) | If Absent  (1-9) | Benefit + Penalty |  | (1-9) |  | (1-9) |  | **(Value %) / (Cost% + Risk%)** |
| Submit Incident  Case | 9 | 9 | 18 | **15.8%** | 3 | **10.0%** | 1 | **5.0%** | **1.053** |
| Update Incident Case | 9 | 9 | 18 | **15.8%** | 3 | **10.0%** | 4 | **20.0%** | **0.527** |
| Generate & Send Status Report | 7 | 7 | 14 | **12.3%** | 3 | **10.0%** | 1 | **5.0%** | **0.820** |
| Dispatchment Request via SMS | 8 | 9 | 17 | **14.9%** | 3 | **10.0%** | 2 | **10.0%** | **0.745** |
| Maintain User Account | 2 | 7 | 9 | **7.9%** | 3 | **10.0%** | 1 | **5.0%** | **0.527** |
| Update Map | 8 | 5 | 13 | **11.4%** | 3 | **10.0%** | 4 | **20.0%** | **0.380** |
| Send Information via Social Media/SMS | 6 | 7 | 13 | **11.4%** | 3 | **10.0%** | 2 | **10.0%** | **0.570** |
| Monitor Crisis Situation | 7 | 5 | 12 | **10.5%** | 9 | **30.0%** | 5 | **25.0%** | **0.191** |
| Total | --- | --- | 114 | **100%** | 30 | **100%** | 20 | **100%** | **---** |

# Appendix C: Prototype (User Interface)