Sahana Biosurveillance Module
Case Management

Software Requirements Specifications

Version 0.5

Authors

Nuwan Waidyanatha

Revision History

Date	Version	Description	Author
19-Dec-2008	0.1	Initial write up	Nuwan Waidyanatha
25-Jan-2009	0.2	Complete first draft to handover to Respere	Nuwan Waidyanatha
03-Mar-2009	0.3	Second draft from Respere	Mifan Careem
04-Jan-2009	0.4	Added Milestones	Mifan Careem
29-June-2009	0.5	Minor modifications to complete the doc	Nuwan Waidyanatha

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

1. Overview

The Sahana Disease Surveillance Module (abbreviated as SHN BSM) intends to capture human health records, analyze the data, and report on the data. The Real-Time Biosurveillance Program (RTBP) will be field testing the software in India and Sri Lanka with the assistance of actual healthcare workers in the rural setting.

The main functional components are – mobile phone j2me application for capturing health records, a database housing the health records, a PHP front end browser application for managing the database information, a browser based C/C++ statistical data mining software for detecting adverse events, and PHP applications for issuing reports to a personal computers and mobile phones via Email, HTTP/WAP posts, and SMS.

This document describes the functional requirements of the GUI components for capturing and reporting the data through desktop web applications. The GUIs will follow a web based structure developed using PHP with control objects to capture person, location, service, facility, and case (disease, symptoms, and signs) information.

The information captured will be stored in a backend MySQL DB with all relevant components named with the prefix “BSM”. In general, the transactional data is gathered via the mobile application; however, the SHN BSM GUIs will provide an alternate computer interfaces for managing the data.

2. Definitions, Acronyms, and Abbreviations

*	Asterisk with an attribute in a table indicates a mandatory element
BSM	Biosurveillance Module
CLDC	Connection Limited Device Configuration
DB	Database
DBA	Database Administrator
GUI	Graphic User Interface
HTTP	Hyper Text Transfer Protocol
IO	Input Output
ISP	Internet Service Provider
J2ME	Java 2 Micro Environment
MIDP	Mobile Information Device Profile
RDBMS	Relational Database Management System
RTBP	Real-Time Biosurveillance Program
SHN_BSM	Sahana Biosurveillance Module
SMS	Short Message Service
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locater
UUID	Universal Unique Identifier
WAP	Wireless Application Protocol
WWW	World Wide Web

3. Network Architecture

The software architecture will strongly consider the network capabilities and capacities when designing the software. The RTBP software is for the developing world and to be accessible via "bottom of the pyramid" affordable ubiquitous terminal devices such as around United States Dollar One Hundred mobile phones. Another aspect to bear in mind is the reliability (coverage and certainty) and affordability of network connectivity. Although GPRS is assumed to solve the internet connectivity problem or EDGE to solve the broadband problem the availability or reliability of these technologies are questionable in the rural settings. However, SMS that functions on the mobile voice service platform is far more reliable but is a lot more expensive than transporting data over GPRS. Developers should investigate the possibility of using alternate technologies such as SMS for exchanging data for redundancy in applications that depend on real time data exchange.

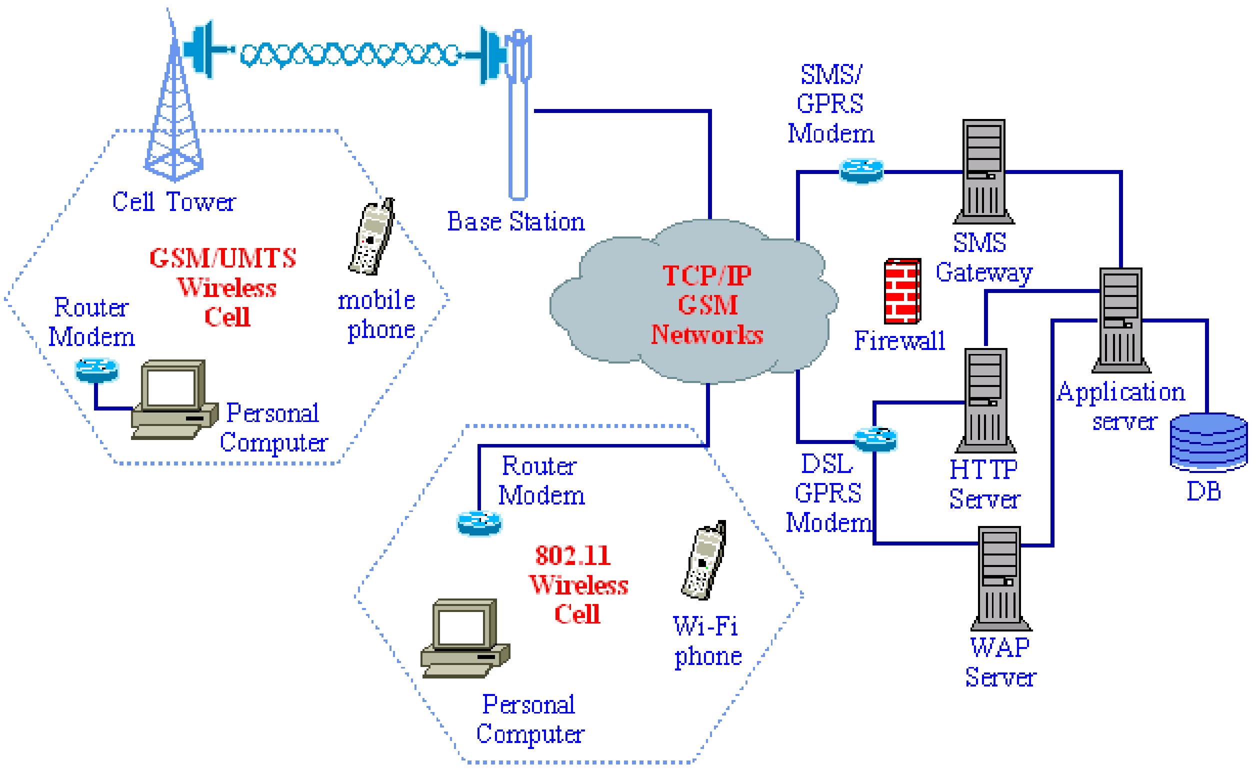

Figure 1 SHN-BSM Communication Architecture

It is important to design the software such that it does not cost the data throughputs or communication overheads. It is easy to get carried away with "all you can eat" DSL networking packages and avoid optimizing the data payloads across the networks. While there may be not much flexibility in optimizing the free text strings, dates, and time it is possible to enumerate fixed list values often used in drop down lists and pass the integer value that takes less capacity than the entire string would. Optimization strategies of this sort will also maximize the throughputs or efficiencies in ever congesting networks.

The network may experience reliability issues; especially with weak wireless signals for communicating data via GSM networks. Therefore, terminals devices must have methods to buffer the data; especially on the client side. This applies not only to mobile phones but also PCs connected via GPRS/SMS modems.

Very thin client and heavy server based WAP accessible applications for mobile terminal devices may be perceived as an effective way to deploy scalable solutions. Establishing poorly available connectivity may cause disappointment with the users with frustrations of connecting and downloading bits form the server. Therefore, the user perceived performance problem can be eliminated by providing hand held device resident applets that provide the functionality for data manipulation and use network connectivity purely to exchange the data between the server and client and not have to burden the network with exchanging GUI specific data as in the case of web pages.

The software components should be reused when scaling for different uses and any new configurations should be backward compatible and not impact any deployed components. For example, mobile phones equipped with WAP 2.0, MIDP 2.0 and CLDC 1.1 should be able to accommodate WAP 1.0, MIDP 1.0 and CLDC 1.0. PHP code must be backward compatible thus run on PHP4.0 inherent browser applications as well as PHP5.0 inherent browser applications.

The DB and UIs should be portable, even though first designed to run on MySQL database and PHP front end programming language the DB will be ported to run on any SQL based DB. Thus the connection component will be independent of the vendor specific connection and shall use ADODB or OLEDB connections generic to most popular DBs.

Ability to add components in the future without having to commit to expensive software architectural or structural changes; i.e. scaling is yet another feature. The data and the components must be independent of the transport protocols or network transport layers. Any parsers or translators that process data received via various transport technologies should be generic and should not be affected by scaling the back end objects to provide additional functionality.

Allocation of abstract functionality within components simplifies extending the functionality via various terminal devices and networks. However, the individual components must be less complex. All actors including users and programmers should be able to understand and verify the concepts and functionality with utmost certainty (i.e. near zero complexity).

Any new additions to the code should not negatively impact the rest of the code (i.e. address evolvability). For example the entity “service” (see section 7.2) is a bare minimal design at the moment which can be extended to fulfill requirements of Inventory services. Extending the service entity to address the Inventory component should not compromise the case, person, or facility services used in the Biosurveillance module. Some customization is required to address the terminal device specific UIs such as mobile phone WAP interface, mobile phone SMS interface, PC based HTTP interfaces.

Visibility can only be sacrificed to the level of protecting confidential data. The system will apply standard security such as firewalls and virus walls to protect from intrusion and malicious destructions to the system software and data.

4. Software Architecture

Figure 2 provides a very high level network based client-server layered architecture of the RTBP software components. The communication between components is through messaging passing by “requests”, “posts”, and “gets”.

Layered architecture is designed such that the layers to the left in Figure 2 does not have to know the mechanics of the layer to the right unless in the case the object specific to that layer and provides similar functions as an object in a layer to the right; e.g. the Module SITREP contains an objects called EDXL/CAP, which is a stand alone object specific to this module but will inherit some of the core objects but if needed will bypass the core layer and associated or inherit with the Database objects.

The levels of abstraction are encapsulated within each of the layers to minimize the complexity of the overall architecture. Figure 2 provides a Domain Specific Software Architecture (DSSA); however, provides the flexibility to reuse most of the objects to address alternate domain requirements; to give an example – Facilities, Person, and Location objects can be applied to generate a Camp Registry Module for managing internally displaced persons after a disaster or the SITREP module can be used to communicate pre and post situation reports of any disaster.

The remote session style architecture allows client side processing to a certain extent as in the case of data capture through the mobile phones using a J2ME applet. Minimizing on the remote data access accept for the occasional case of updating the static datasets in the mobile application. This minimizes the number of server calls and reduces complexity in terms of issues related to unreliable connectivity.

Although WAP as a gateway parser and PDA as a terminal device are included in the diagram (Figure 2) during this phase of the RTBP pilot, these components will not be developed but illustrated to highlight the flexibility of the architecture.

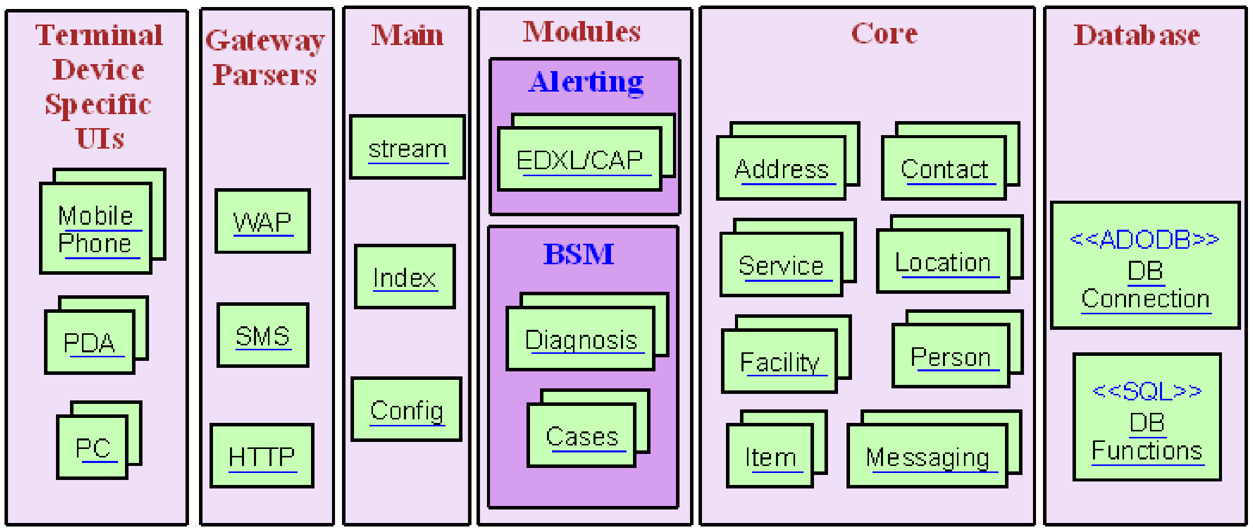

Figure 2 Layers and objects of the complete Biosurveillance product software architecture

Table 1 Layers, objects, and descriptions of elements in the layers in Figure 2

Object	***Brief description***
Layer: ***Database***	
DB Connection	Uses the PHP ADODB to provide the connectivity to the database for accessing data
DB Functions	Standard set of Sahana functions based on: SELECT, UPDATE, DELETE, INSERT, SHOW, CREATE TABLE, etc to provide the Sahana specific logic is provided by this object (handler-db.inc)
Layer: ***Core*** *(reusable abstract classes)*	
Item	Basic element that describes a physical object in relation to “inventory management”; i.e. management of matter in time and space.
Person	Person inherits the core classes: <<contact>> and <<address>> to identify a person’s communication and habitat information.
Location	This may be an abstract class with respect to Sahana but inherits the <<GIS>> class to provide GIS features; especially with Google Maps. A location may be take on any geometric shape: point, line, circle, rectangle, polygon,
Facility	Facilities in relation to Health disease surveillance are mostly hospitals, clinics, etc but can be extended to accommodate any building, property, or infrastructure. Even a camp to house internally displaced people is regarded as a facility. The shape or position of a facility is defined by a location; hence inherits the <<location>> class.
Contact	A contact is a “mode” or “method” to communicate with a person or facility. It is an abstract class.

Address	Address is an abstract class that is related to the postal address of a person or facility.
Service	A service defines a process carried out at a location on a set of items by a processor (e.g. person or machine) on another object such as a person, facility, case, etc. Therefore, a service inherits <<person>>, <<location>>, and <<item>> classes.
Messaging	A message is information sent by a “sender” to one or more “recipients” in the form of voice, text, graphic mediums. This class inherits <<person>>, <<location>> and <<contact>> classes
Layer: ***Modules***	
BSM: Diagnosis	Diagnosis is an abstract class that constructs the relationships between disease, symptoms, sign, and causality factor. These objects will be inherited by the cases class.
BSM: Cases	Module inherits <<person>>, <<location>>, <<facility>>, <<service>>, <<address>>, and <<contact>> core objects along with the module specific health object: <<cases>> to produce the Disease Surveillance (Biosurveillance) functional module.
Alert:EDXL/CAP	Module inherits <<person>>, <<messaging>>, <<location>>, and <<contact>> core objects along with the module specific emergency communication standards based object: <<EDXL/CAP>> to produce the SITuation REPort functional module.
Layer: ***Main***	
Config	“Configuration” (config.inc) file contains the DB (e.g. MySQL), Web service (e.g. Apache), and other installation specific information
Index	Sahana uses a REST like architecture with the index.php as the main API two main parameters: module and action to specify the module containing the object and the action specifying the function to call in the module.
Stream	The method in which data is streamed depends on the terminal device specific application. The desktop web Sahana application uses normal HTTP streaming to apply CSS with banners and theme to provide a uniform look and feel; while J2ME mobile phone application uses pure TEXT streaming to avoid overhead in the data transmitted from server to mobile phone.
Layer: ***Gateway Parser***	
WAP	WAP UIs accessing the Sahana objects, such as mobile phones with GPRS, will use this object which will parse the function to interpret and present in a way understood by the Module, Core, and Main classes.
HTTP	HTTP UIs accessing the Sahana objects, such as PCs through a browser and DSL/UMTS connections, will use this object which will parse the function to interpret and present in a form understood by the Module, Core, and Main classes. If the Module, Core, and Main classes are developed in PHP, parsing will not be required, since they are written in PHP as well. However, a JSP or ASP function will need to be parsed to transform the function call to be

	recognized by PHP code.
SMS	UIs exchanging data via SMS the transport with the Sahana object will use this object which will parse the function to interpret and present in a way understood by the Module, Core, and Main classes.
Layer: ***Terminal device specific UIs***	
Mobile Phone	The miniaturization aspect of the mobile phone and J2ME specific application development restricts the flexibility in adopting this terminal device for data communication. Network efficiencies also constraints transporting data over GPRS/EDGE or SMS transports. WAP gateway allows connecting to a server to exchange information. On board applet can use HTTP, WAP, or SMS gateway for connectivity and communicate information with the DB.
Personal Digital Assistant	PDAs will act in the same manner as desktop web application; where the modules can be accessed via GPRS. Given PDA is J2ME compliant, can execute the same applications developed for the mobile phones.
Personal Computer	PCs provide the best in terms of flexibility and comprehensive functionality; however, doesn't provide the mobility relative to mobile phones or PDAs. Connectivity can be provided through DSL, UMTS, or GPRS/SMS modems with browser based or desktop applications connecting via the HTTP gateway.

4.1. External Interfaces

- Public IP address with ISP to access the web server
- DSL or GPRS/SMS modem to with cables (serial or UTP) to connect to internet
- Intel or Sun server to install the database and web server
- Web server (IIS or Apache) with alias directory system
- Database (MySQL) to install the tables
- Operating system (Windows or Linux) for the RDBMS and Web server to run on
- Browser (Firefox or Explore) to access the application

4.2. Design Constraints

The design constraints are the same for all DB, Core, Main, Module, and Gateway Parser components; therefore, this section provides a general set of design constraints for all.

- Programming language must be a web browser executable code base like PHP, Python, ASP, etc

- The GUI should be independent of the available databases such as MySQL, MS SQL, Oracle, DB2, etc
- Accessing the software is independent of the network connectivity
- Usability of the GUIs will vary upon the terminal device; e.g. PC with Firefox browser will not have the same restrictions as a mobile phone with WAP
- Software development tools should not violate open source standards; i.e. if another programmer wants to change the source code the source code should be independent of the tool used to write the source code
- If using dependent module such as GIS or other the installation and implementation should not be cumbersome and force too many system changes
- The main functions should be written such that they can be used by other GUIs such as a desktop GUI or mobile phone GUI.
- Super user and users must be able to access the information directly from the main menus
- Navigate through the search result set 20 records at a time

5. Software Structure

5.1. Main Scenarios

The main scenarios are identical for the entire set of Core and Module objects, which are the search, edit, add, delete, and associate (lookup) sequences. The generalized version of the scenarios will be discussed in this section and any deviations will be discussed in the relevant sections.

Edit records

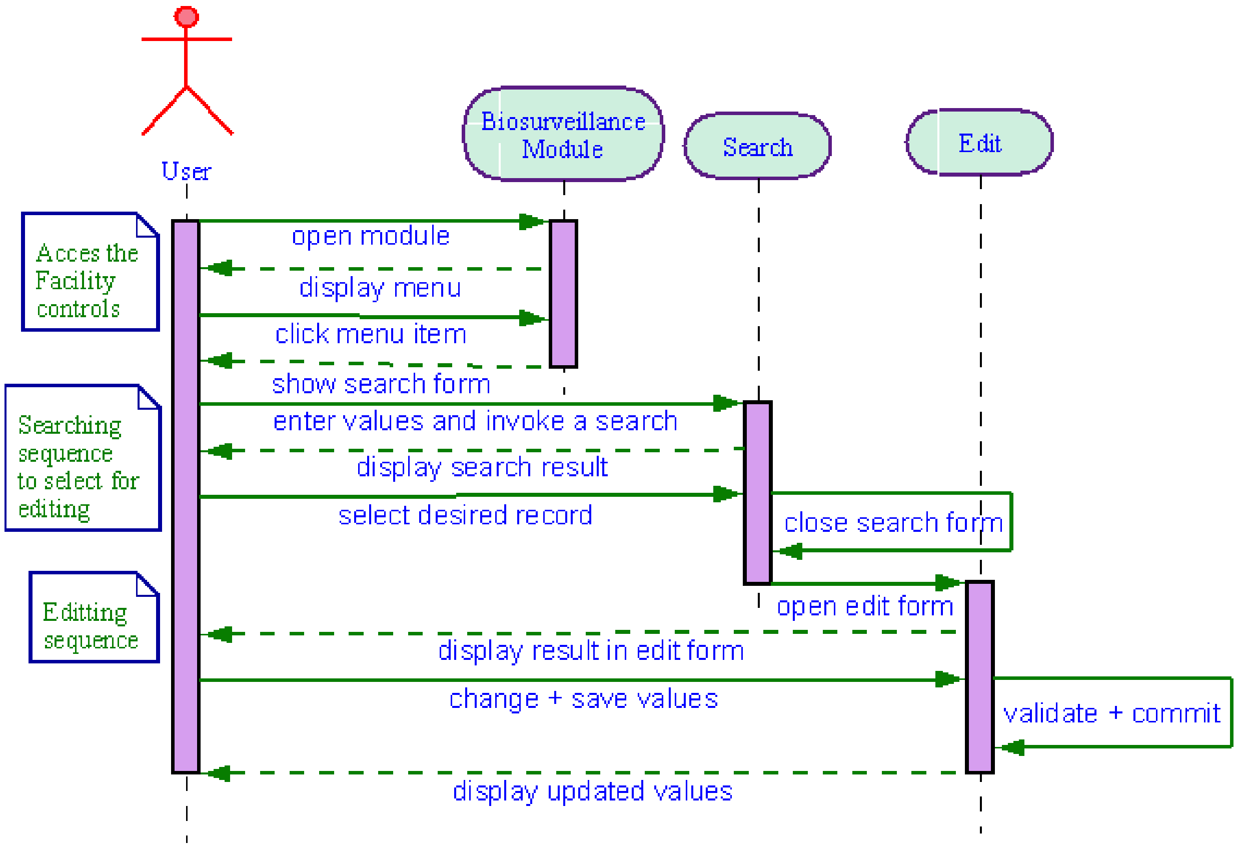

Figure 3 Edit sequence diagram

- Select the desired entity object controls through the main menu by clicking the menu item
- The search form is presented to the user with attributes to filter the search
- User will select values from presented lists and enter partial or full text in free text controls and invoke the process to retrieve records from database
- The returned result set, based on search criteria, is presented in tabulated form giving the user the option to select a desired record for editing or search again with new set of filter values (or criteria)
- Upon selection of record from presented list the record is displayed in the edit control form to change values and save to update record in database.
- Updated record is displayed in same edit form

Add records

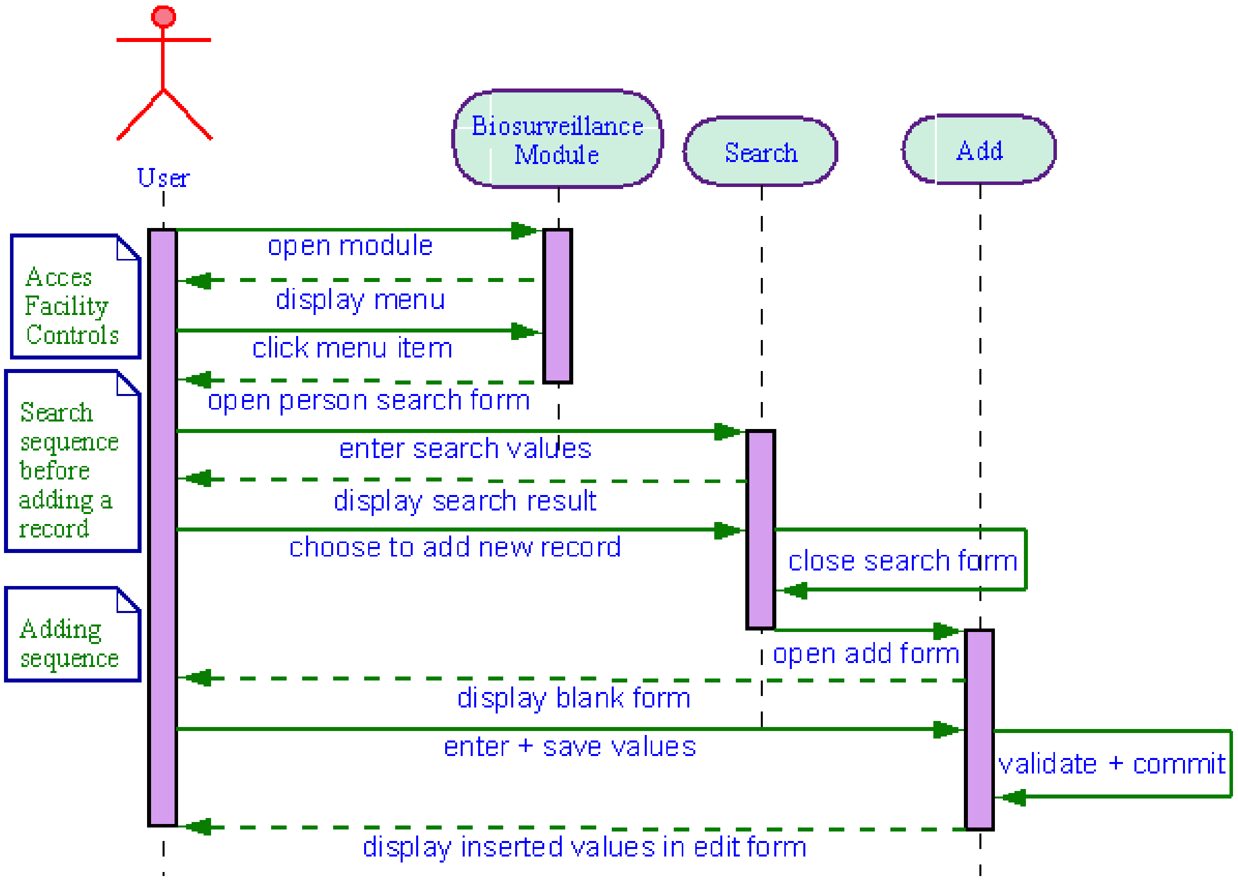

Figure 4 Add sequence diagram

- Select the desired entity object controls through the main menu by clicking the menu item
- The search form is presented to the user with attributes to filter the search
- User will select values from presented lists and enter partial or full text in free text controls and invoke the process to retrieve records from database
- The returned result set, based on search criteria, is presented in tabulated form giving the user the option to select a desired record for editing or search again with new set of filter values (or criteria)
- If search does not return the desired value then the user has the option of adding a new record by clicking the "add" record control.
- The user is presented with a blank form with the pre set list values to select from and cleared text and date controls to set the values.
- Upon completion the form values are save by invoking the save control and committing the record to the DB.
- The committed record is repainted or displayed in the edit control form with options to edit the values and also set the associated information. These related entities (or objects)

are not accessible until the primary details are committed to database and a uuid is available to make the relationships with associated entities.

Delete records

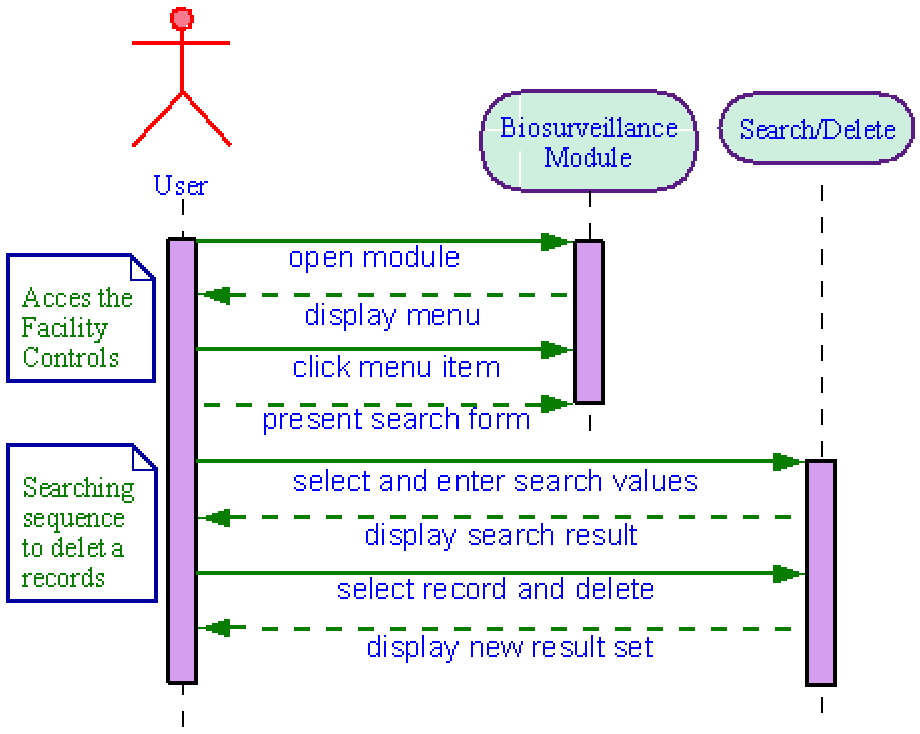

Figure 5 Delete sequence diagram

- Select the desired entity object controls through the main menu by clicking the menu item
- The search form is presented to the user with attributes to filter the search
- User will select values from presented lists and enter partial or full text in free text controls and invoke the process to retrieve records from database
- The returned result set, based on search criteria, is presented in tabulated form giving the user the option to select a desired record for editing or search again with new set of filter values (or criteria)
- The user has the option of selecting one or more records and invoking the delete control button to deactivate the records in the database.
- The records are removed by setting the deactivate date to indicate that the record is invalid from that day on; maintaining the referential integrity without harming any relationships that are in the DB
- Either the deleted records are highlighted in red or simply eliminated from the display

Access external records relating information

A diagram is shown only for "Cases" but the concept of the sequence flow is identical for the other entities. Therefore, this document will only discuss the Cases entity leaving it for the readers to make the analogy for the service, contact, and address entities.

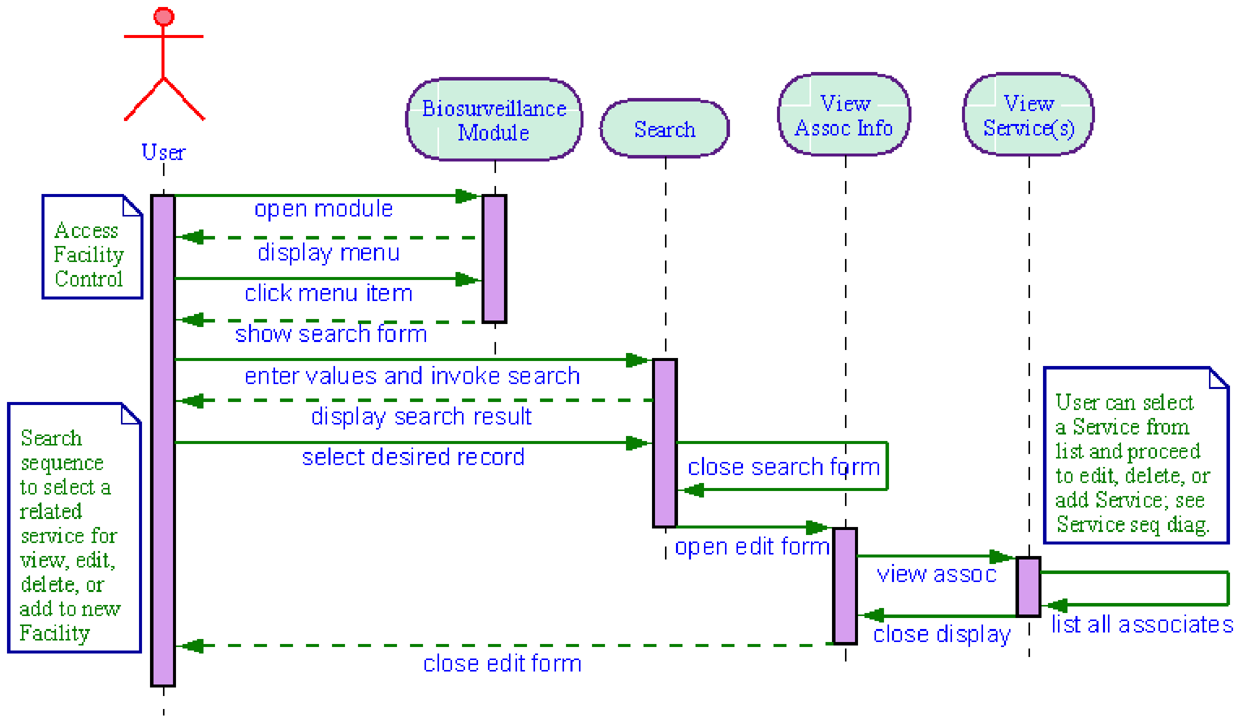

Figure 6 View associated information to add, edit, or delete

- Select the facility controls through the main menu by selecting or clicking the "Facility" menu item
- The search form is presented to the user with attributes to filter the search
- User will select values from presented lists and enter partial or full text in free text controls and invoke the process to retrieve records from database
- The returned result set, based on search criteria, is presented in tabulated form giving the user the option to select a desired record for editing or search again with new set of filter values (or criteria)
- In the Edit control form a section will be designated to display information on the related entities contact, address, service, and cases; in this section necessary and sufficient information or records will be displayed in tabular form or <href> links or button controls will provide access to the full set of information in the respective entity's edit form or access to search form to related a new record or to the add control form to add a new

record and relate to the facility. See the respective entity's use case and sequence diagrams to learn about the functions.

- Upon completion of viewing the record or adding a new record the handle is returned to the Facility edit form control.

5.2. General guidelines to GUI controls design

This section will set out some basic guidelines to designing the GUIs. The elements of the GUIs will be classified as the following objects: forms, reports, text, date, lists, buttons, and links.

- Each core and module object should provide a set of forms to search, edit, add, and delete records per the sequence described in section 5.1. In order to reuse the same form for viewing, editing, and adding records the form will take on the modes: *view*, *edit*, and *add* respectively with embedded functionality.
- A search form is different from the view, edit, and add forms, which does not contain all the attributes of a record but only to the necessary and sufficient attributes to filter the search of desired record(s). A search form will have two modes: *result* and *lookup*. The result mode is simply displaying the filtered search results for the purpose of selecting a record for editing.
- Second mode is a search form acting as a lookup form when relating records. For example, when patient (person) needs to be assigned to a health case the case edit form invokes the person search form to fetch for the results. When an available person record is selected from the list that particular person is related to the health case record through the person uuid.
- Text, List, Date, and Button control modes are – *enabled* or *disabled*; where only enabled controls are active and can be used; while disabled controls cannot. For example a disabled text control can be used to simply display the value but not allow the user to alter the value.
- Reports should provide the user with attributes to set the criteria; i.e. filter. Also provide the option to view all results or sets of results. Another option should be to save the results as a CSV or PDF.

5.3. Guidelines for database

- Create date, create by, and create process – these attributes appear in all transactional tables. This is for audit and mainly for research purposes. They are mandatory and must be set during an INSERT. Create date tells us when the record was created; create by tells us who created the record (i.e. user name), and create process tells us the process or how it was created (i.e. via mobile phone over GPRS or SMS, PC browser over GPRS or DSL, external bulk load interface)

- Modify date, modify by, and modify process – these attributes are similar to the create date, create by, and create process attributes except they are set every time during an UPDATE. This gives us the same set of information of the last update.
- Deactivate date – is seen in all tables. To ensure associations are not broken by removing the record from the table during a DELETE the deactivate date is set to deactivate the record. This maintains the old relationships and does not affect past reports or queries. All select statements should contain the statement "WHERE deactivate_dt IS NULL or deactivate_dt < today()"; the function today() being the present date or it an be replaced with a specific date to exclude all records deactivated before the indicated date.
- Select – when querying the database to retrieve records the statement should only select the necessary and sufficient attributes to minimize the payload.
- Update – when updating an attribute, if the attribute is not nullable and the process is trying to replace a valid value (i.e. not null value) with a null value then that is not allowed.
- Insert – prior to an insert the process should check if the record already exists in the database. If it is a 90% match then the user should be prompted to search for the record again to avoid duplication.
- Delete – the SQL DELETE function is not used instead the record deactivation process is implemented; see above on "deactivate date"

6. Usability of software components

6.1. Software developers

- Software developers enhancing or adding on functionality must be able complete their tasks in minimal time without having to read through code to understand the function of the entire code base
- Good documentation of the functions must in place for evaluators and future developers to understand the processes invoked by the components
- Ideally all inputs and outputs of data should use a standard such as XML to avoid the software components being dependent on a single programming language such as PHP; thereby, future components developed using other programming languages can still communicate with the predecessor components

- The instructions and guidelines will be made available through the web as well as a printed document. The on line web version will have more literature than the printed version.

6.2. System Administrators

- Setting up of public IP to access the web server will be no more than a ½ day task. This is a standard process done in collaboration with the ISP but the application should not complicate the standard process.
- Installation of web server, database, and application, inclusive of downloading the software, will be no more than a 1 – 2 hour task; thus, clear and precise, self training, instruction must be provided
- Self learning training material with instruction set will be provided for installing the application, which should not take any longer than an hour, inclusive of reading the instructions
- Configuring the application is guided through the step by step installation process itself. However, alternate approaches, such as directly writing the values in to the “cong.ini” file will be provided. This process includes setting up the administrative user account.
- Using the administrative user account one can begin assigning other user roles, accounts and grant user permissions. The procedure is self explanatory but additional written instructions will also be provided. Given, the administrator has preplanned the user roles, with the set of users, and permissions, the setup should not take any more than an hour (approximately 20 users); time will vary depending on the number of users.
- A “test” and “live” database will be made available to the implementer for testing implementation concepts and then transferring that to a live version used by all users. Setting up a test version of the application is done through setting the “Incidence” attribute in the conf.ini file and creating a separate database with a different name. This process should not take any longer than a fresh installation of the application.

6.3. Implementers

- Implementer must first write the static values such as the location category, location types, person roles, person types, diseases, symptoms, signs, etc. Since this is a onetime process with very minor updates, it is considered a DBA task. The DBA will use standard SQL language such as INSERT, UPDATE, SELECT, DELETE to write the preplanned values in to the database. Depending on the implementation this process may take from 1 to 2 days, inclusive of a verification process.

- Testing the implementation will be on the “test” version of the installation. Transferring of the implementation to the live version is as easy as repeating the process carried out on the test version or copying the tables, view, and procedures to the live database.
- The instructions and guidelines will be made available through the web as well as a printed document. The on line web version will have more literature than the printed version.

6.4. Users

- A one day training workshop should provide the necessary hands on training to use the application in creating, editing, deleting, searching, and associating records of all the objects.
- Training and reference manuals will be provided for trained users to refer to when in doubt. These manuals will have examples of how to conduct each task made available through a series of controls in the application
- The instructions and guidelines will be made available through the web as well as a printed document. The on line web version will have more literature than the printed version.

7. Core Object Functionality

The core objects will function as abstract classes with APIs to request, get, and post records sets in relation to the individual objects. They will also contain a set of GUIs interface for reading, selecting, and writing data. The following sections describe the specifications for each of the abstract objects.

7.1. Location

Location is a self contained independent object used not only by the BSM but by other Sahana modules such as Victim Management and SITREP. The other objects - person, address, facility, case, and service use the concept of locations.

Location category defines the hierarchy of the ontology and taxonomy of the specific location classification. For the purpose of health and Biosurveillance the category defined will be labeled as “Health”. Examples of alternate location categories are be the Governance structure, Disaster Management structure or Water management structure.

Location Type is the subcategory but has the structure to be defined in a hierarchical order. For example, Sri Lanka health sector hierarchy of locations, in ascending order, are – National, Regional, Provincial, District, MOH divisions, and PHI areas; similarly in India – National, State, District, Block, and Village; which are classified as location types.

7.1.1. Real Need

- Identify the location the case originated such as the village health worker recorded the data or facility the patient reported the illness to
- Define hierarchical areas (or boundaries) based on the health sector governance structure
- Identify clusters of diseases or disease densities by locations
- Target reports and alerts by locations

7.1.2. User Goals

- Search locations by category, type, or name
- Edit location details including assigning the parent location (i.e. wider area location belongs to)
- Add a new location
- Delete one or more locations by deactivating the record
- Assign standard location descriptors such as GIS coordinate or ISO code systems to the records

7.1.3. Actors & Roles

- *Super User (Implementer/DBA)*:
  - Define location categories and types
  - Add, Edit, & Remove location categories and types

- *User (Health Worker)*
  - Add, Edit, and Remove specific locations

7.1.4. Use-Case Diagrams

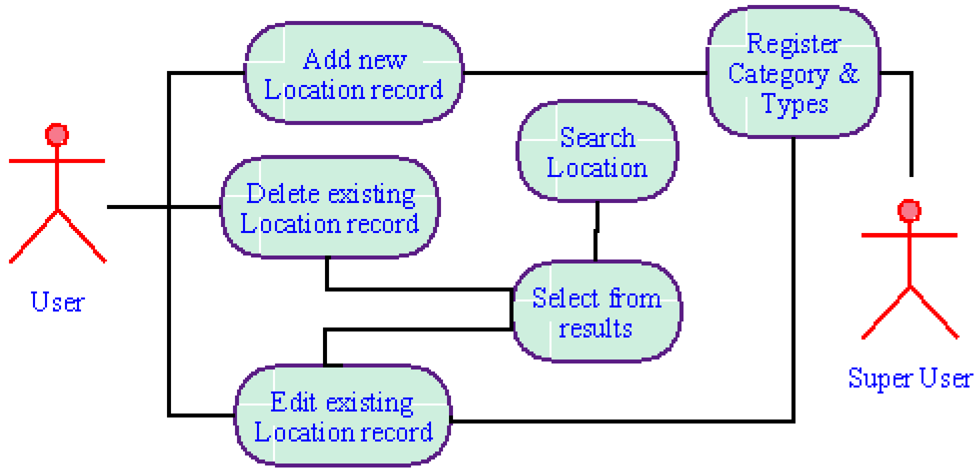
Add new Location record
Register Category & Types
Search Location
Delete existing Location record
User
Select from results
Super User
Edit existing Location record

Figure 7 location use case diagram

Table 2 description of use cases in Figure 1

Search record	Provide necessary and sufficient set of attributes for user to enter full or partial information; then apply search function to retrieve relevant dataset from database to be displayed in tabular for user to select one or more records
Select from result	Tabulated results should be presented with options to select one record for editing and one or several desired records for deleting. Also provide the option to add a record if desired. Tabulation mandatory elements are display the location name, parent location, category, and type; while other values are optional but displayed if they are necessary for user to distinguish between two similarly named locations
Edit record	First search for the record and select the desired record from result set; then make changes to the values and save
Add new record	By default the user should be forced to search for the record and if not in result then allow to add a new location record; the record will select a location category and type from registered list
Delete record	First search for the record and select the desired record from result set; then apply delete function

7.1.5. Main Scenarios

- Create facility categories
- Create facility types and assign a category

- Create facility status
- Create a facility and assign a category, type, and status
- Lookup and associate a location to the facility <<location>>
- Assign an address to a facility
- Edit facility details
- Deactivate facility from database
- Search a facility record

7.1.6. Data Storage (files and databases)

Table 3 Location category information

Attribute	***Data Type***	***Comment***
Location Category	Varchar	Health, Governance, etc the location classification
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Description	Varchar	Detailed explanation of the category to be used in reports, help or other human readable explanations
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 4 Location detail information

Attribute	***Data Type***	***Comment***
Location Type	Varchar	Subcategories with hierarchical structure
Location Category	Varchar	Type should be associated with a category
Description	Varchar	Detailed explanation of the category
Type	Varchar	Health location types: MOH, PHI,
Deactivate date time	Date Time	To deactivate the record instead of deleting

Table 5 Location detail information

Attribute	***Data Type***	***Comment***
Universal Unique ID (uuid)	Integer/Varchar	Unique identifier for the record
Parent uuid	Integer/Varchar	Identifier pointing to the parent location
Location Name	Varchar	Human readable name to id the location
Location Category	Varchar	Health, Governance,
Location Type	Varchar	Health location types: MOH, PHI,
Description	Varchar	Additional human readable info

ISO Code	Varchar	ISO 3116 country codes or 8440 international trade location code
Coordinate System	Varchar	Name of the vector coordinate system such as GIS lat/lon, GIS utm, that gives meaning to the X, Y, Z, coordinates
Shape	Varchar	Geometric shape of the location e.g. circle, polygon, line, point
X vector	Varchar	A polygon will have a set of x coordinates X = (x1, x2, …, xn) e.g. latitudes
Y vector	Varchar	The same polygon will have equal number of y coordinates Y – (y1, y2,…yn); e.g. longitudes
Z vector	Varchar	If 2 dimensional then a set of Z coordinates representing the height else zero Z=(0, 0, …,0)
Create date time	Date Time	To identify when the record was created
Create by	Varchar	User or System creating the record
Create process	Varchar	How the record was created; e.g. web, mobile phone, the application, external
Modify date time	Date Time	Identify when record was last modified
Modify by	Varchar	User or System modifying the record
Modify process	Varchar	The method the record was modified
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

7.1.7. Associations

- Location is inherited by objects such as person, facility, address, person, and service.

Table 6 Other object controls quasi modally accessing the location object

Interface Name	***Remarks***
▪ Person add and edit controls	A health worker is associated with a location; e.g. a Public Health Inspector is assigned a jurisdiction.
▪ Facility add and edit controls	A hospital is associated with a location; e.g. Kuliyapitya base hospital is situated in the Kuliyapitiya MOH division
▪ Service add and edit controls	A service such as investigating a patient can be carried out in a particular Public Health Inspector area.
▪ Case add and edit controls	A case has to be identified with a location to calculate the disease densities.
▪ Address add and edit controls	An address may belong to a particular governance area such as a tribunal or village or a district

7.1.8. Implementation Notes

- Identify and add set of location categories and related types with hierarchy in database
- Set user permissions as to who can add, edit, delete, search, and associate location records

7.2. Service

Service, similar to the location, is a self contained independent object of that is, mainly, used by the health cases object but also can be used by other classes such as inventory, person, or facilities.

Service category defines the specific class of services and the type specifies the sub category of services. For example, the category: Health Worker Services may have several types: investigate a case, follow up case, or notify village.

Service is driven by state transitions which are recorded as the state according to a given sequence. The date and time stamp tells when the state transition occurred.

A service may have inputs outputs such as a set of items; in the context of health we may regard an input as a blank set of document and the output as the filled set of document (reports).

A person (or system or machine), termed as the provider must carry out the service and a person (or system or machine), termed as the recipient, will receive the benefits of the service. For example; the Public Health Inspector will “provide” the service of investigating a case by visiting the patient’s (recipient) home.

7.2.1. Real Need

- Provide a method to define, record, and track services in relation to health cases carried out by health workers such as investigating a case
- Provide a method for the laboratories to carryout their work in relation to a heath case and record the procedures and outcomes (or results)
- Facilities provide services such as quarantine, maternity, cardiac health services. The mechanism should define the services provided by the facilities for the purpose of referrals
- In the event of intervention and prevention procedures the heath officials should have a method to define the services to be carried out specific persons or facilities in given locations.

7.2.2. User Goals

- Create service categories, types, and states
- Associate a service to the entities: Person, Facility, and Case by creating a new service instance
- Track the progress or to edit the content in the service
- Set the status of the service along with a timestamp indicating the state transition of the service
- View service progress reports based on the status and time period

7.2.3. Actors & Roles

- *Super User (Implementer/DBA)*:
  - Define the category and type of services
  - defines the type of person that provides the service and the type of person that may receive the service
  - define the input and output items associated with the service type

- *User (Health Worker)*:
  - Add, Edit, and Remove specific services associated with person, case, or facility
  - Set the status of the service
  - View aggregated service progress reports

7.2.4. Use-Case Diagrams

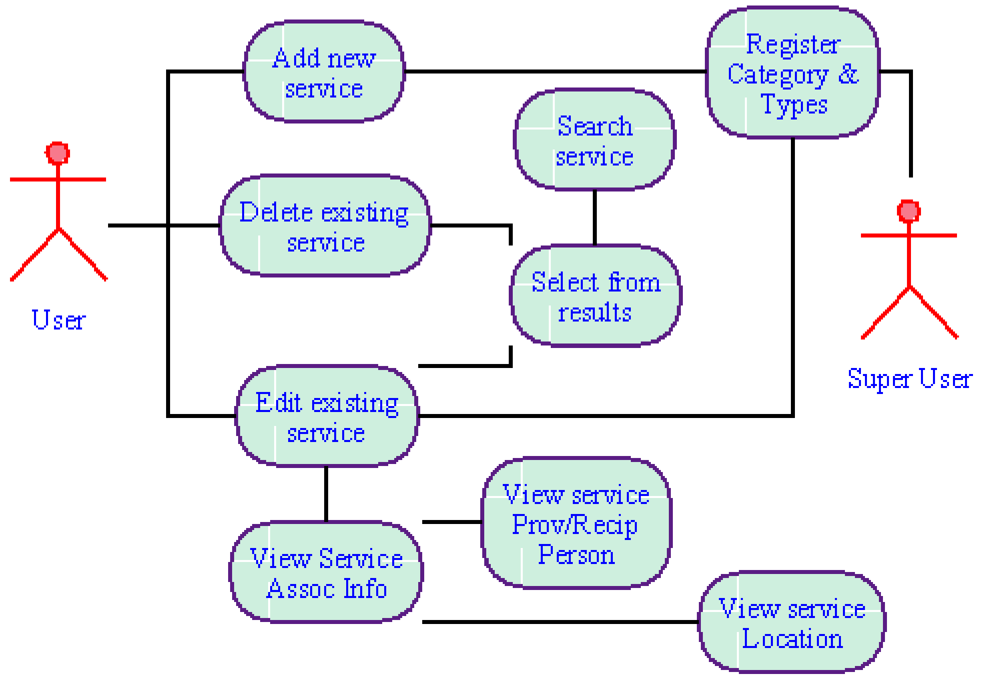

Figure 8 service use case diagram

Table 7 description of use cases in Figure 8

Search record	Provide necessary and sufficient set of attributes for user to enter full or partial information; then apply search function to retrieve relevant dataset from database to be displayed in tabular for user to select one or more records
Select from result	Tabulated results should be presented with options to select one record for editing and one or several desired records for deleting. Also provide the option to add a record if desired. Tabulation mandatory elements are display the service id, category, type, state; while other values are optional but displayed if they are necessary for user to distinguish between two similarly named locations
Add new record	By default the user should be forced to search for the record and if not in result then allow to add a new service record
Edit record	First search for the record and select the desired record from result set; then make changes to the values and save
Delete record	First search for the record and select the desired record from result set; then apply delete function

7.2.5. Data Storage (files and databases)

Table 8 Service category information

Attribute	***Data Type***	***Comment***
Service Category*	Varchar	Define if Health Service or other based on implementation
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Description	Varchar	Detailed explanation of the category to be used in reports, help or other human readable explanations
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 9 Service type information

Attribute	***Data Type***	***Comment***
Service Type*	Varchar	Subcategories with hierarchical structure
Service Category*	Varchar	Type should be associated with a service category in Table 8
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Description	Varchar	Detailed explanation of the category
Process Notes	Varchar	Provide instructions on conducting the type of service
Expected Result	Varchar	Explanation on the expected result of the service outcome
Expected Time	Time	Number of days or hours the service should be processed or completed in; i.e. the start data and end date of the service will be set as a function of this values
Provider Type	Varchar	Person type providing the service; e.g. Public Health Inspector
Recipient Type	Varchar	Person type receiving the service e.g. Patient
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 10 Service status information

Attribute	***Data Type***	***Comment***

Service State*	Varchar	Name to identify the transitional state of the service
Service State Sequence	Integer	A number to indicate the service transitional sequence order for a given type and category
Service Category*	Varchar	State should be associated with a Category in Table 8
Service Type*	Varchar	State should be associated with a Category in Table 9
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Description	Varchar	Detailed explanation of the status to be used in reports, help or other human readable explanations
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 11 Service type item information

Attribute	***Data Type***	***Comment***
Item Name*	Varchar	Name to identify the item
Item Category*	Varchar	Status should be associated with a Category in Table 8
Item Type*	Varchar	Status should be associated with a Category in Table 9
Item State	Varchar	Indicate whether item is an input to or output of the service process; by default will be considered an output
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Description	Varchar	Detailed explanation of the item to be used in reports, help or other human readable explanations
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table Comment: to define the set of inputs and outputs necessary for carrying out each service type

Table 12 Service item detail information

Attribute	***Data Type***	***Comment***
Service Universal Unique ID (uuid)*	Integer/Varchar	Unique identifier for the record same as the service uuid in Table 11 with the option of

		adding additional items specific to the instance
Item Name*	Varchar	Same name as defined in Table 11
Item State	Varchar	Indicate whether item is an input to or output of the service process; by default be considered an output item
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table Comment: table will store all instances of services created through functional processes such as by case, facility, or person

Table 13 Service item detail information

Attribute	***Data Type***	***Comment***
Service Universal Unique ID (uuid)*	Integer/Varchar	Unique identifier for the record same as the service uuid in Table 11 with the option of adding additional items specific to the instance
Service Category	Varchar	Same Service Category in Table 8
Service Type	Varchar	Same Service Type as in Table 9
Service Status	Varchar	Same as in Service Status as in Table 11
Status Date	Date Time	Date and Time stamp the status was set
Provider	Varchar	Actual name of the Provider carrying out the service, obtained from Person entity
Recipient	Varchar	Actual name of the Recipient receiving the service, obtained from Person entity
Start Date	Date Time	The date and time the service is scheduled to be carried out or the date and time the service actually began; choice of the implementers to set the policy
End Date	Date Time	The date and time the service is scheduled to be completed by or the date and time the service actually was completed; choice of the implementers to set the policy
Notes	Varchar	For additional notes, remarks, comments such as to record the outcome or results and any changes in the procedures
Create date time	Date Time	To identify when the record was created
Create by	Varchar	User or System creating the record
Create process	Varchar	How the record was created; e.g. web, mobile phone, the application, external
Modify date time	Date Time	Identify when record was last modified
Modify by	Varchar	User or System modifying the record
Modify process	Varchar	The method the record was modified

Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table Comment: table will store all instances of services created through functional processes such as by case, facility, or person

7.2.6. Main Scenarios

- Create facility categories
- Create facility types and assign a category
- Create facility status
- Create a facility and assign a category, type, and status
- Lookup and associate a location to the facility <<location>>
- Assign an address to a facility
- Edit facility details
- Deactivate facility from database
- Search a facility record

7.2.7. Associations

- Service is inherited by objects such as person, facility, and cases.

Table 14 Other GUIs quasi modally accessing the service controls: forms & reports

Interface Name	***Remarks***
▪ Person add and edit controls	A patient or health worker can be assigned a series of services such as test reports to be completed by a certain date.
▪ Facility add and edit controls	A hospital will be required to setup a quarantine service or a special vaccination campaign
▪ Case add and edit controls	A case will have investigative services, follow up services, reporting services

7.2.8. Implementation Notes

- Identify and add set of service categories and related types with hierarchy in database
- Define the execution times, procedures and expected results in the type entity
- Identify and add set of states for a given type and setup the transitional sequence
- Set user permissions as to who can add, edit, delete, search, and associate location records

7.3. Person

Person is an abstract object used in Biosurveillance. In disease surveillance the main actors: health worker and patient will be defined as a person. The services will identify the service provider and service recipient from the person entity. Person is also used by other Sahana modules.

Persons are classified by their role and each role creates a unique instance; where typical roles can be “health worker”, “patient”, “field worker”, and “governor” with the freedom for implementers to define.

Within a given role there could be many sub roles, which shall be regarded as types. Some examples of person types for the role of health worker are Health Inspector, Deputy Director Health Services, and General Practitioner.

The state of a person can be regarded as the entropy of the person at a given time and can be labeled accordingly. In the existing Sahana system people states are labeled as missing, victim, or dead and in the proposed Biosurveillance module the patients may be be labeled as sick, well, dead, active, inactive, etc per the implementers’ policy.

Although the database may have several instances of a person based on the role different roles can be mapped to a single person through their passport, national identification, or driving license.

Personal details will be the date of birth, vital statistics (height weight), gender, ethnicity (race and religion), marital status, and country of birth (or origin); the list is not exhaustive.

7.3.1. Real Need

- Person is central to the module as preliminary work on Biosurveillance surrounds human diseases
- The intervention and prevention services are carried out by health workers; i.e. a class of persons that play the role of health workers
- The users are another class of people engaging in managing the information system
- The person entity is used by case and service entities

7.3.2. User Goals

- Manage person details
- Associate a person to a case or service

- Associate addresses and contact details of a person
- View case history reports on a person and the status
- View a person’s service history reports and the progress

7.3.3. Actors & Roles

- *Super User (Implementer/DBA)*:
  - Define the roles, types, and states of a person
  - Assign system user roles and rights
  - Setup the person related services

- *User (Health Worker)*:
  - Add, Edit, and Remove specific services associated with person, case, or facility
  - Set the status of the service
  - View aggregated progress reports

7.3.4. Use-Case Diagrams

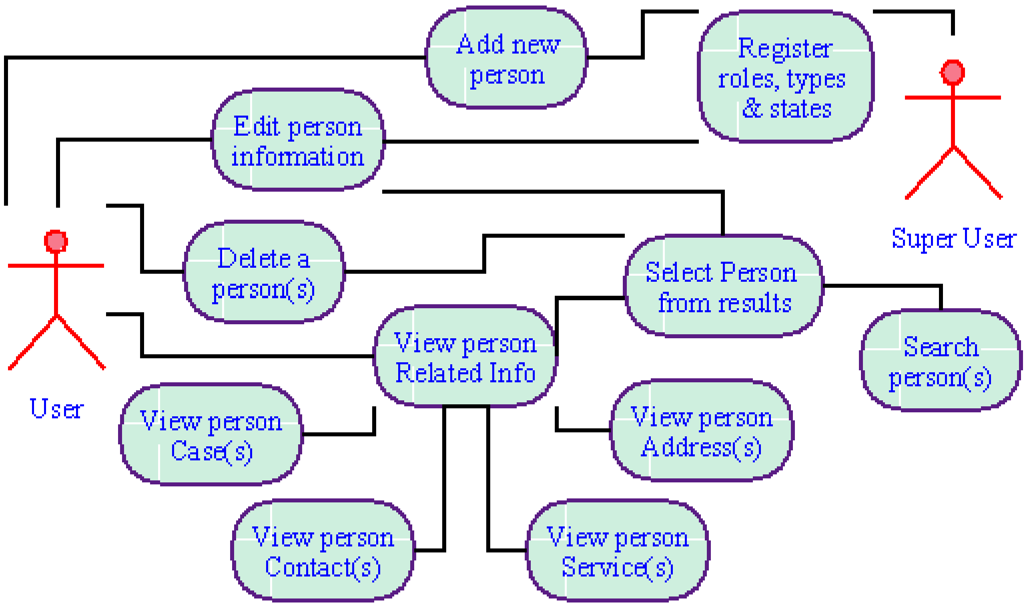

Figure 9 person use case diagram

Table 15 description of use cases in Figure 1

Search record	Provide necessary and sufficient set of attributes for user to enter full or partial information; then apply search function to retrieve relevant dataset from database to be displayed in tabular for user to select one or more records
Select from result	Tabulated results should be presented with options to select one record for editing and one or several desired records for deleting. Also provide the option to add a record if desired. Tabulation mandatory elements are display the person name, identifications, role, and type; while other values are optional but displayed if they are necessary for user to distinguish between two similarly named locations
Edit record	Search for the record and select the desired record from result set; display selected record in edit form, then make changes to the values and save. User is allowed to change the role and type of the person provided another instance of the same person with same role does not exist. Question: if, role or type is changes, do we save the new instance as a new record under an alternate person uuid or do we change the values and save under the same person uuid?
Delete record	First search for the record and select the desired records from result set; then apply delete function; followed by a confirmation and display of new result set. Note record is deactivated and not physically removed from database.
Add new person	By default the user should be forced to search for the person, to avoid duplication, with known attributes such as name and identification information. If anticipated name not in result then allow to add a new person record; only one instance of a person with same name and identification may be allowed for a given role and type; check this criteria before committing record to DB.
View person cases	A control should be provided from the person edit form to the person's related cases with provision to edit, add, and delete cases; functions which will be inherited through the Case object.
View person services	A control should be provided from the person edit form to the person's related services with provision to edit, add, and delete service information; functions which will be inherited through the service object.
View person address	A control should be provided from the person edit form to the person's related addresses with provision to edit, add, and delete address information; functions which will be inherited through the address object.
View person Contacts	A control should be provided from the person edit form to the person's related contacts with provision to edit, add, and delete contact information; functions which will be inherited through the

	Contact object.

When collaborating with other objects such case or service the entry point for associating a location with these objects is through the search use case.

7.3.5. Main Scenarios

- Create person roles
- Create person types for each role
- Create a set of person states for each type and role
- Search a person record
- Edit a person record
- Add a person record
- Delete a person record
- Lookup and associate person addresses <<Address>>
- Look up and associate person contact details <<Contact>>
- Lookup and associate person services <<Service>>

7.3.6. Data Storage (files and databases)

Table 16 Person role information

Attribute	***Data Type***	***Comment***
Person role*	Varchar	Roles can range from health worker, field worker, patient, user, etc providing the freedom for implementers to define as seen fit
Description	Varchar	Detailed explanation of the role to be used in reports, help or other human readable explanations
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 17 Person type information

Attribute	***Data Type***	***Comment***
Person type*	Varchar	Different type of roles within a main role; e.g. role = health worker, types = nurse, doctor, village health nurse, medical officer of health

Person role*	Varchar	Same as in Table 16; one role with many types
Description	Varchar	Detailed explanation of the type to be used in reports, help or other human readable explanations
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 18 Person state information

Attribute	***Data Type***	***Comment***
Person state*	Varchar	States can range from active, inactive, missing, dead, victim
Person role*	Varchar	Same as in Table 16; a person's state can be role specific; e.g. deceased is only applicable to patients and active/inactive is only applicable to health workers
Description	Varchar	Detailed explanation of the status to be used in reports, help or other human readable explanations
Enumeration	Integer	Number to communicate instead of sending string to optimize bytes as in mobile phones
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 19 Person information

Attribute	***Data Type***	***Comment***
Person UUID*	Varchar	Name to identify the item
Person Role*	Varchar	Status should be associated with a Category in Table 16
Person Type*	Varchar	Status should be associated with a Category in Table 16
Person State*	Varchar	Indicate whether item is an input to or output of the service process
Passport	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
National ID	Varchar	Detailed explanation of the item to be used in reports, help or other human readable explanations
Driving License	Date Time	Records will not be deleted from database only

		deactivated to maintain referential integrity
Last Name*	Varchar	Surname or family name
First Name	Varchar	Given name
Middle Name	Varchar	Other given names
Alias	Varchar	Other names or alternate names used
Gender	Varchar	Male, Female, or Unknown
Designation	Varchar	Reverend, Minister, President, Lawyer
Dependent Person UUID	Varchar	Mother, Father, or other dependent person or relative; the dependent person must be registered in the database and related via person uuid
Age	Decimal	Numeric value with 2 decimal places
Age Group	Varchar	Infant (< 1), child (1 – 12), Teen (12 – 19), Youth (20 – 30), Adult (30 – 55), Elder (> 55)
Date of Birth	Varchar	Year / Month / Day
Height	Decimal	Person's height
Height unit	Varchar	Inches, Feet, Meters, Centimeters
Weight	Decimal	Person's weight
Weight unit	Varchar	Kg, lb, gr,
Religion	Varchar	Christian, Hindu, Buddhist, Islam,
Race	Varchar	Han, Naxi, Sinhala, Tamil, Muslim
Marital status	Varchar	Single, Married, Divorced, Separated
Skin color	Varchar	Black, White, Yellow, Brown
Eye color	Varchar	Green, Blue, Brown, Black
Hair color	Varchar	Blue, Blond, Black, White
Body Markings	Varchar	Scar above left eyebrow, birthmark on chest
Blood Type	Varchar	O, O-, A, A+, AB
Notes		For any additional remarks about the person
Create date time*	Date Time	To identify when the record was created
Create by*	Varchar	User or System creating the record
Create process*	Varchar	How the record was created; e.g. web, mobile phone, the application, external
Modify date time	Date Time	Identify when record was last modified
Modify by	Varchar	User or System modifying the record
Modify process	Varchar	The method the record was modified
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

7.3.7. Associations

- Person is inherited by objects such as service and cases.
- Person inherits services, address, and contact

Table 20 Other GUIs quasi modally accessing the Person controls: forms & reports

Interface Name	*Remarks*
▪ Service add and edit controls	A service contains a record of the person providing the service and the person receiving the service
▪ Cases add and edit controls	A health case record will contain the patient and health worker information

7.3.8. Implementation Notes

- Identify and add the set of person roles and related types with hierarchy in database
- Define the set of person states to the implementer's liking

7.4. Facility

A Facility is a building, property or infrastructure. Facility is an abstract object that can be used in many business cases and one being heath facilities in relation to the disease surveillance.

Facility entity is inherited by the health Cases object and the facility inherits the service, location and address objects.

Facilities are categorized and type classified along with a name to create unique instance. A typical category can be a "health care" facility, "medical storage" facility, or "quarantine" facility. For a given category there are many sub categories termed as types; e.g. set of types for category: health facilities are hospital, clinic, maternity home, general practitioner.

The status of a facility describes the functional state of the facility independent of the category or type. The status can easily be confined to the set – designed, building, opened, closed, operational, occupied, unknown, etc.

Another vital piece of information is the location of the facility, which can be defined by a point in space, which may be the most common but also by a geometric shape.

7.4.1. Real Need

- Patients will be reporting their health cases in health facilities and the system should be able to trace back a health record to the facility
- Aggregate health cases by facilities they are reported from
- Different types (or classes) of facilities provide different sets of services and identifying the service by facility may provide benefits in containing diseases by referring patients to the correct facility

7.4.2. User Goals

- Create, edit, delete a facility
- Relate a facility to a case
- Assign a location or address to facility
- Maintain up to date details on the facility
- View case history and aggregate reports by a facility or collection of facilities
- View a facility service history reports and the progress

7.4.3. Actors & Roles

- *Super User (Implementer/DBA)*:
  - Define the categories and types
  - Assign system user roles and rights
  - Setup the facility related services

- *User (Health Worker)*:
  - Add, Edit, and Remove specific facilities
  - Add, Edit, and Remove associated health cases, services, addresses, or location from a facility
  - Set the status of the facility
  - View aggregated reports

7.4.4. Use-Case Diagrams

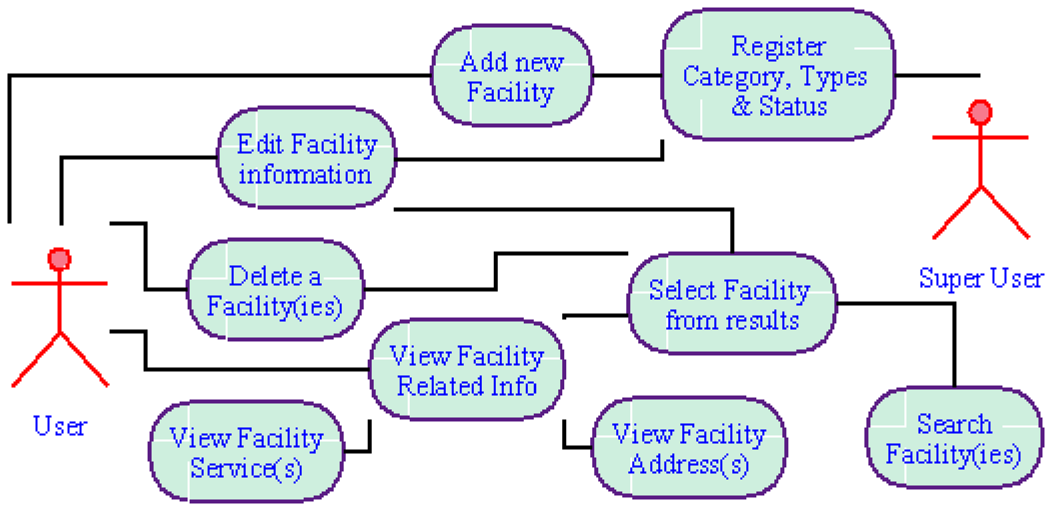

Figure 10 Facility use case diagram

Table 21 description of use cases in Figure 10

Search Facility(ies)	Provide necessary and sufficient set of attributes for user to enter full or partial information; then apply search function to retrieve relevant dataset from database to be displayed in table for user to select one or more records
Select Facility from result	Tabulated results should be presented with options to select one record for editing and one or several desired records for deleting. Also provide the option to add a record if desired. Tabulation mandatory elements are facility name, category, and type; while other values are optional but displayed if they are necessary for user to distinguish between two similarly named facilities
Edit Facility Information	Search for the record and select the desired record from result set; display selected record in edit form, then make changes to the values and save. User is allowed to change the category and type of the facility provided another instance with same name category, and type does not exist. Question: if, category or type is changes, do we save the new instance as a new record under an alternate facility uuid or do we change the values and save under the same facility uuid?
Delete Facility(ies)	First search for the record(s) and select the desired record(s) from result set; then apply delete function through the available control (e.g. button); followed by a confirmation. Upon completion of the delete-function display new result set. Note record is deactivated and

	not physically removed from database.
Add new Facility	By default the user should be forced to search for the facility, to avoid duplication, with known attributes such as the name, category, and type information. If user determines anticipated facility record is not in result set then provide a control to add a new facility record; only one instance of a facility with same name, category, and type may be allowed for a given role and type; check this criteria when validating before committing record to DB.
View Facility Service(s)	A control should be provided from the facility edit form to the facility's related services with provision to edit, add, and delete service information; functions which will be inherited through the service object.
View Facility address	A control should be provided from the facility edit form to the facility's related addresses with provision to edit, add, and delete address information; functions which will be inherited through the address object.

When collaborating with other objects such case or service the entry point for associating a location with these objects is through the search use case.

7.4.5. Main Scenarios

- Create facility categories
- Create facility types and assign a category
- Create facility status
- Create a facility and assign a category, type, and status
- Lookup and associate a location to the facility <<location>>
- Assign an address to a facility
- Edit facility details
- Deactivate facility from database
- Search a facility record

7.4.6. Data Storage (files and databases)

Table 22 Facility category information

Attribute	***Data Type***	***Comment***
Facility category*	Varchar	Categories can range from health, storage, government, etc; user
Description	Varchar	Detailed explanation of the category to be used in reports, help or other human readable

		explanations
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 23 Facility type information

Attribute	***Data Type***	***Comment***
Facility type*	Varchar	Different type of facilities within a category; e.g. health facility type = hospital, clinic,
Facility Category*	Varchar	Same as in Table 22;
Description	Varchar	Detailed explanation of the type to be used in reports, help or other human readable explanations
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 24 Facility Status information

Attribute	***Data Type***	***Comment***
Facility Status*	Varchar	States can range from designed, under construction, operational, closed, etc
Description	Varchar	Detailed explanation of the status to be used in reports, help or other human readable explanations
Enumeration	Integer	Number to communicate instead of sending string to optimize bytes as in mobile phones
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 25 Facility information

Attribute	***Data Type***	***Comment***
Facility UUID*	Integer/Varchar	System assigned universal unique identifier
Facility Category*	Varchar	Same as in Table 22
Facility Type*	Varchar	Same as in Table 23
Facility Status*	Varchar	Same as in Table 24
Description	Varchar	Detailed explanation of the status to be used in reports, help or other human readable

		explanations
Location UUID	Varchar	Location UUID same as in Table 5
Create date time*	Date Time	To identify when the record was created
Create by*	Varchar	User or System creating the record
Create process*	Varchar	How the record was created; e.g. web, mobile phone, the application, external
Modify date time	Date Time	Identify when record was last modified
Modify by	Varchar	User or System modifying the record
Modify process	Varchar	The method the record was modified
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

7.4.7. Associations

- Facility is inherited by objects: cases.
- Facility inherits Address, Contact, and Location objects

Table 26 Other GUIs quasi modally accessing the Facility controls: forms & reports

Interface Name	***Remarks***
▪ Cases add and edit controls	A patient record may be received from a hospital

7.4.8. Implementation Notes

- Identify and add set of facility categories and related types with hierarchy in database
- Define the set of facility status to the implementer's liking

8. BSM Module Functionality

8.1. Entity Relationship Diagram

This section describes the module specific objects that provide functionality with respect to the Biosurveillance domain requirements termed as the BSM. The specifications focus on two main objects called the "diagnosis" and "cases"; where the diagnosis provides the data structure of the disease, symptoms, signs, and causality factors and the cases provides the data structure with functionality to capture health records and related information for carrying out the disease outbreak detection analysis and services for intervention and prevention . The relevant module objects inherit selected classes from the core module.

The entity relation ship diagram in Figure 11 describes the data structure and associations with core module components. The person core object is used to define the roles: "health worker" and "patient" both of whom are actors of the health domain. The health worker role would be further segregated according to the various types such as epidemiologist, regional epidemiologist, Medical Officer of Health, Public Health Worker, and community health worker (Suwacevo) in the Sri Lankan setting and Deputy Director of Health, Public Health Center Doctors, Public Health Center Nurses, Health Inspectors, and Village Health Nurses in the Indian setting. The patients and health workers will contain contact details and address information.

For the purpose of biosurveillance and with respect to the health sector, a "Health" location category will be established with various types embedded with the category such as national, regional, provincial, district, divisional, and areas in the Sri Lankan context and national, state, district, block, and village in the Indian context. The location is associated with the patient records through the cases object. Health workers, health facilities, and patients are further directly associated through the inherited location attribute or the address object.

Service can be associated with any entity but for the purpose of Biosurveillance will be, mainly, associated with the cases entity to define a public health worker to follow up investigating a case that requires a house visitation to inspect the environment or a patient requested to obtain laboratory services of a full blood test.

Health cases are a key entity which is described in section 8.3 below. Cases entity holds source information such as the patient details and location information as well as the medical information such as the disease, signs, symptoms, and causality factors.

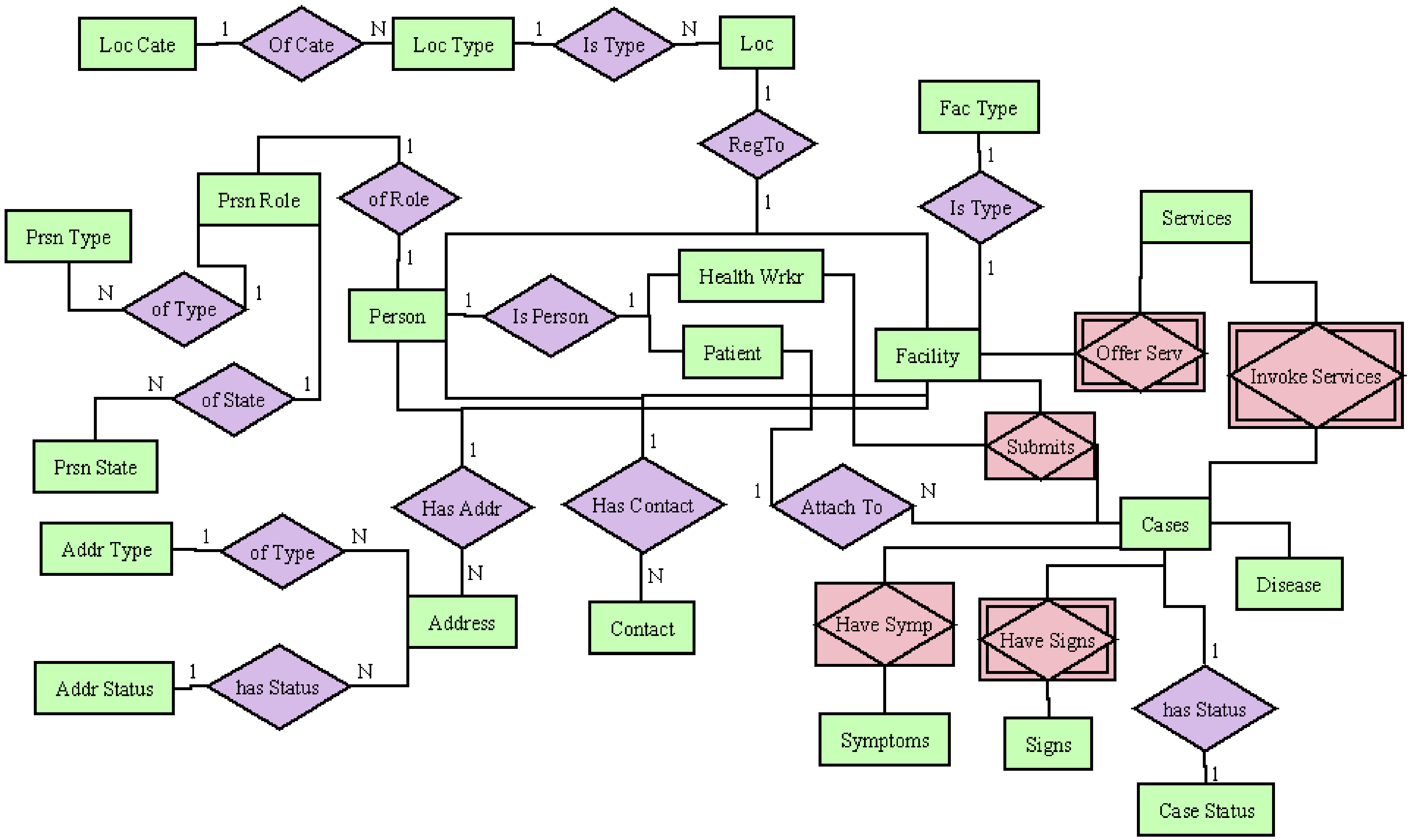

Figure 11 SHN BSM Module Entity Relationship diagram

8.2. Diagnosis

The BSM module comprises a cases and a diagnosis object. This section discusses the diagnosis object used by the cases object, a construction of four entities: disease, symptom, sign, and causality factors. The term diagnosis is used to indicate the functional aspect of the process of identifying the symptoms, signs then implicating a disease. Symptoms are what the patient would report. Signs are what the health worker examining the patient would observe. Causality factor is the element that causes the illness such as consumption of expired food in the case of food poisoning. Disease is the name given to the health condition with the defined symptoms and signs or a variation of them with in the scope. WHO has defined a set of ICD – International Classification of Disease codes which will be assigned to the known diseases whenever they exist (i.e. not all diseases have an ICD code)

8.2.1. Real Need

- Contains elements of the medical knowledge specifics for human disease Biosurveillance
- Used by the health cases object to create records sets of diseases, signs, symptoms, and causality factors
- Provides a grouping structure for producing aggregated health reports for monitoring the health status by certain diseases, symptoms, signs, or causality factors
- Used by the analytics algorithms as the basis information for detecting disease outbreaks

8.2.2. User Goals

- Setup and relate sets of diseases, symptoms, signs, and causality factors
- Use the related diagnosis information in the health cases records with provision to add, edit, and delete the values based on the different scenarios or existence of predefined values
- Design aggregated reports grouped by diseases, symptoms, signs, or causality factors over geo spatial dimensions (i.e. locations).
- Periodically update the diagnosis information and relationships
- Associate periodically updated WHO ICD codes with the datasets

8.2.3. Actors & Roles

- *Super User (Implementer/DBA)*:
  - Define disease types

- *User (Health Worker)*
  - Search, Add, Edit, and Delete disease records
  - Search, Add, Edit, and Delete symptom records
  - Search, Add, Edit, and Delete sign records

- Search, Add, Edit, and Delete causality factor records
- Relate disease with sets of symptoms, signs, and causality factors

8.2.4. Use-Case Diagrams

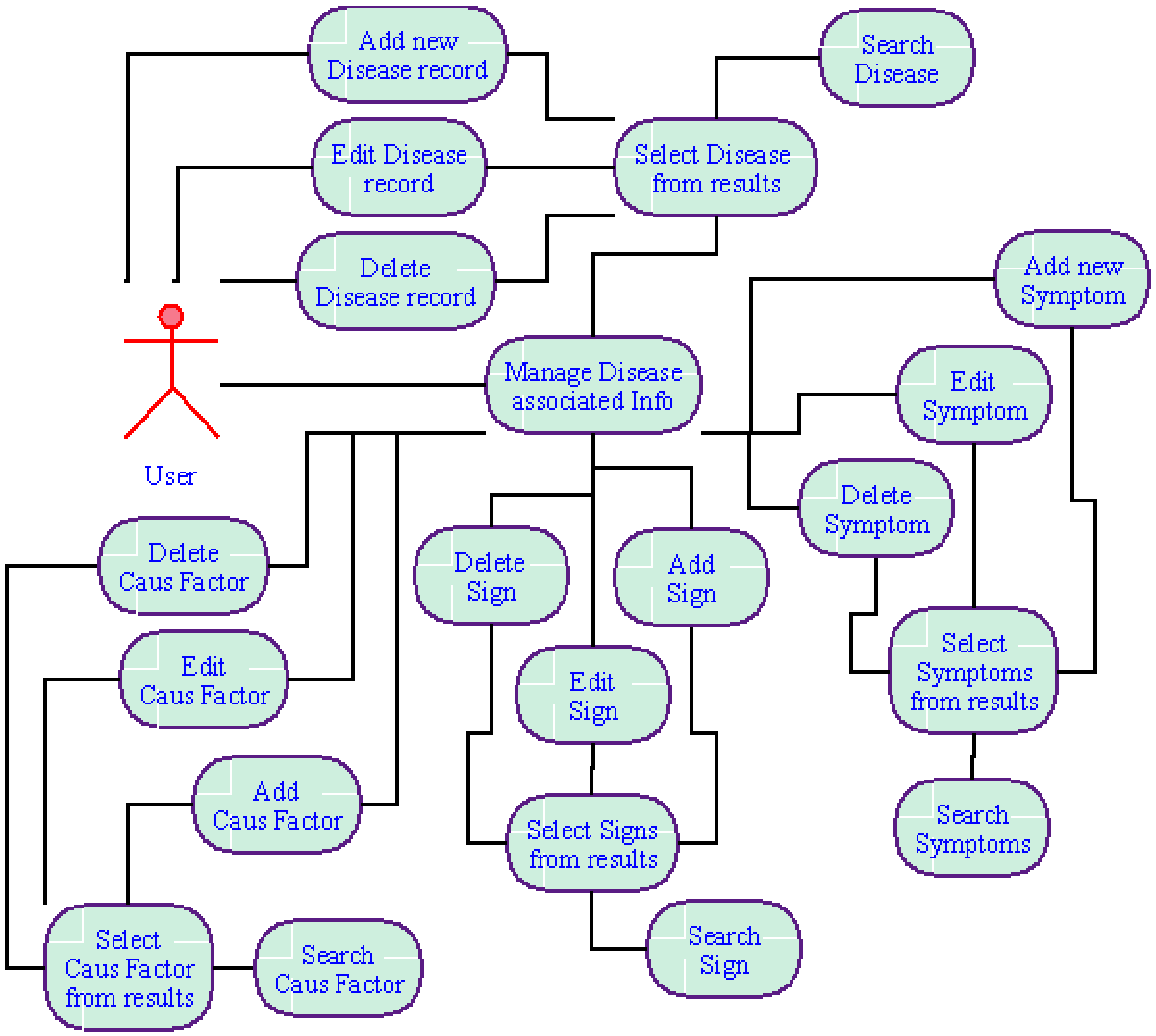

Figure 12 diagnosis (disease, sign, symptoms, and causality factor) use case diagram

Table 27 description of use cases in Figure 12

Search disease record	Provide necessary and sufficient set of attributes for user to enter full or partial information; then apply search function to retrieve relevant dataset from database to be displayed in tabular form for user to select one or more records
Select disease from result	First search for the record and select the desired record from result set to proceed with viewing, editing, or deleting functions
Add new disease	By default the user is forced to search for the record and if not in

record	result then allow to add a new disease record; the record should be assigned to a particular disease type from registered list
Edit disease record	First search for the record and select the desired record from result set; then make changes to the values and save
Delete disease record	Tabulated results from the search will be presented with the option to select one or more records for deletion; delete process only deactivates the record by setting the deactivate data and time
Manage disease associated information	Through this use case the user can access the symptoms, signs, and causality factors objects to associate the respective information with a disease
Search symptom	Provide necessary and sufficient set of attributes for user to enter full or partial information; then apply search function to retrieve relevant dataset from database to be displayed in tabular form for user to select one or more records
Select disease from result	First search for the record and select the desired record from result set to proceed with viewing, editing, or deleting functions
Add new symptom	By default the user is forced to search for the record and if not in result then allow to add a new symptom record to save to database
Edit symptom	First search for the record and select the desired record from result set; then make changes to the values to update record
Delete symptom	Tabulated results from the search will be presented with the option to select one or more records for elimination; delete process only deactivates the record by setting the deactivate data and time
Search sign	Provide necessary and sufficient set of attributes for user to enter full or partial information; then apply search function to retrieve relevant dataset from database to be displayed in tabular form for user to select one or more records
Select disease from result	First search for the record and select the desired record from result set to proceed with viewing, editing, or deleting functions
Add new sign	By default the user is forced to search for the record and if not in result then allow to add a new disease record to save to database
Edit sign	First search for the record and select the desired record from result set; then make changes to the values to update record
Delete sign	Tabulated results from the search will be presented with the option to select one or more records for elimination; delete process only deactivates the record by setting the deactivate data and time
Search cause factor	Provide necessary and sufficient set of attributes for user to enter full or partial information; then apply search function to retrieve relevant dataset from database to be displayed in tabular form for user to select one or more records
Select disease from result	First search for the record and select the desired record from result set to proceed with viewing, editing, or deleting functions
Add new cause factor	By default the user is forced to search for the record and if not in result then allow to add a new causality factor record to save to database
Edit cause factor	First search for the record and select the desired record from result

	set; then make changes to the values to update record
Delete cause factor	Tabulated results from the search will be presented with the option to select one or more records for elimination; delete process only deactivates the record by setting the deactivate data and time

8.2.5. Main Scenarios

- Search disease records
- Create disease records
- Edit disease records
- Delete disease records
- Lookup and associate symptoms <<symptom>>
- Lookup and associate signs <<sign>>
- Lookup and associate causality factors <<causality factor>>
- Search symptoms records
- Create symptom records
- Edit symptom records
- Delete symptom records
- Search sign records
- Create sign records
- Edit sign records
- Delete sign records
- Search causality factor records
- Create causality factor records
- Edit causality factor records
- Delete causality factor records

8.2.6. Data Storage (files and databases)

Table 28 Disease Type information

Attribute	***Data Type***	***Comment***
Disease Type	Varchar	Distinguish between classes of diseases such as heart diseases, skin diseases, ENT diseases
Description	Varchar	Explanation of the type, which may be found in the ICD definitions as well
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 29 Disease information

Attribute	***Data Type***	***Comment***
Disease	Varchar	The name of the disease in short form, which is also the primary key of the record and cannot

		be duplicated
Enumeration	Integer	Value to communicate instead of sending string to optimize bytes as in mobile phones
Disease Type	Varchar	Same as in Table 28
Disease Priority	Varchar	Indicates whether it is a high, medium, or low priority, which defines the level of attention given to the health case
ICD Code	Varchar	International Classification of Diseases codes
ICD Description	Varchar	Human readable description of the ICD code
Notes	Varchar	Other notes relevant to the disease
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 30 Symptom detail information

Attribute	***Data Type***	***Comment***
Symptom	Varchar	The name of the symptom in short form, which is also the primary key of the record and cannot be duplicated
Enumeration	Varchar	Value to communicate instead of sending string to optimize bytes as in mobile phones
Symptom Priority	Varchar	Indicates whether it is a high, medium, or low priority, which defines the level of attention given to the health case
Description	Varchar	Human readable explanation of the symptom that can be used in presentation material such as reports
Symptom Code	Varchar	A standard coding system, similar to the ICD codes, used to classify symptoms
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

Table 31 Sign detail information

Attribute	***Data Type***	***Comment***
Sign	Varchar	The name of the sign in short form, which is also the primary key of the record and cannot be duplicated
Enumeration	Varchar	Indicates whether it is a high, medium, or low priority, which defines the level of attention given to the health case
Sign Priority	Varchar	Indicates whether it is a high, medium, or low priority, which defines the level of attention given to the health case
Description	Varchar	Human readable explanation of the sign that can be used in presentation material such as

		reports
Sign Code	Varchar	A standard coding system, similar to the ICD codes, used to classify signs
Deactivate date time	Date Time	

Table 32 Causality Factor detail information

Attribute	***Data Type***	***Comment***
Causality Factor	Varchar	The name of the causality factor in short form, which is also the primary key of the record and cannot be duplicated
Enumeration	Varchar	Indicates whether it is a high, medium, or low priority, which defines the level of attention given to the health case
Causality Factor Priority	Varchar	Indicates whether it is a high, medium, or low priority, which defines the level of attention given to the health case
Description	Varchar	Human readable explanation of the causality factor that can be used in presentation material such as reports
Causality Factor Code	Varchar	A standard coding system, similar to the ICD codes, used to classify causality factors
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

8.2.7. Associations

Figure 13 shows the association of the disease class with the symptom, sign, and causality factor classes. The disease class uses the lookup function to invoke the symptom, sign, or causality factors. Each of the class objects have their own search, add, edit, and delete functions to manage the data within the individual classes.

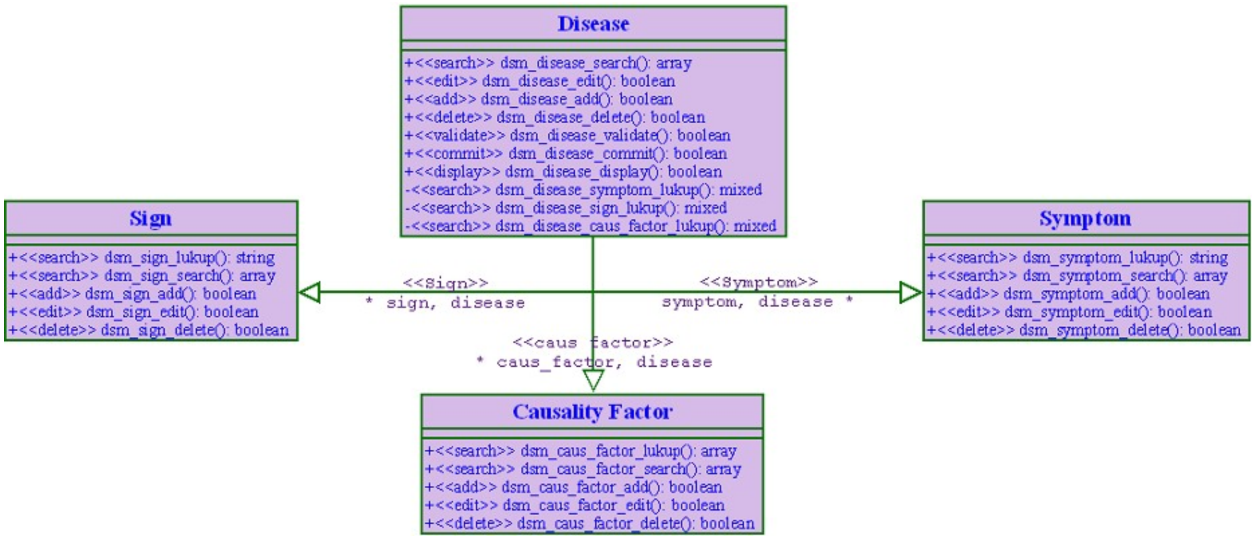
Disease
+<<search>> dsm_disease_search(): array
+<<edit>> dsm_disease_edit(): boolean
+<<add>> dsm_disease_add(): boolean
+<<delete>> dsm_disease_delete(): boolean
+<<validate>> dsm_disease_validate(): boolean
+<<commit>> dsm_disease_commit(): boolean
+<<display>> dsm_disease_display(): boolean
-<<search>> dsm_disease_symptom_lukup(): mixed
-<<search>> dsm_disease_sign_lukup(): mixed
-<<search>> dsm_disease_caus_factor_lukup(): mixed
Sign
+<<search>> dsm_sign_lukup(): string
+<<search>> dsm_sign_search(): array
+<<add>> dsm_sign_add(): boolean
+<<edit>> dsm_sign_edit(): boolean
+<<delete>> dsm_sign_delete(): boolean
Symptom
+<<search>> dsm_symptom_lukup(): string
+<<search>> dsm_symptom_search(): array
+<<add>> dsm_symptom_add(): boolean
+<<edit>> dsm_symptom_edit(): boolean
+<<delete>> dsm_symptom_delete(): boolean
<<Sign>>
* sign, disease
<<Symptom>>
symptom, disease *
<<caus factor>>
* caus_factor, disease
Causality Factor
+<<search>> dsm_caus_factor_lukup(): array
+<<search>> dsm_caus_factor_search(): array
+<<add>> dsm_caus_factor_add(): boolean
+<<edit>> dsm_caus_factor_edit(): boolean
+<<delete>> dsm_caus_factor_delete(): boolean

Figure 13 Class inheritance diagram

Table 33 Other GUIs quasi modally accessing the location object

Interface Name	***Remarks***
▪ Person add and edit form (web interface)	A health worker is associated with a location; e.g. a Public Health Inspector is assigned a jurisdiction.
▪ Facility add and edit form (web interface)	A hospital is associated with a location; e.g. Kuliyapitya base hospital is situated in the Kuliyapitiya MOH division
▪ Service add and edit form (web interface)	A service such as investigating a patient can be carried out in a particular Public Health Inspector area.
▪ Case add and edit form (web interface)	A case has to be identified with a location to calculate the disease densities.
▪ Address add and edit form (web interface)	An address may belong to a particular governance area such as a tribunal or village or a district

8.2.8. Implementation Notes

- Step 1: define the disease types
- Step 2: define the list of symptoms, signs, and causality factors
- Step 3: define the set of diseases and associate the respective type, symptoms, signs, and causality factors

8.3. Cases

Health "cases" is the key object that stores the health domain patient specific information for the BSM module. It inherits the diagnosis, person, service, facility, and location abstract classes.

A health case is a record that contains a patient's identification details, diagnosis information, and location information. The main goal of the Biosurveillance module is to identify adverse disease clusters and disease density propagation through the geographical space. The relevant information for detection is obtained trough the set of cases datasets.

8.3.1. Real Need

- Main piece of health information for Biosurveillance, which contains the source, carrier, disease, symptoms, signs, time, and location
- Required for analysis for detecting disease outbreaks or finding adverse health events in the datasets of patient health data
- Producing aggregated health reports for monitoring the health status
- Provides health case history for a given patient (person)

8.3.2. User Goals

- Define and relate sets of diseases, symptoms, and signs
- Define case related services
- Define locations and facilities to relate to cases
- Assign a parent location (i.e. larger geographic area) to a location; e.g. a province to a district
- Create, edit, add, and delete a health case record with mandatory elements location and symptoms
- Maintain the status history of the case
- Assign and monitor health case services

8.3.3. Actors & Roles

- *Super User (Implementer/DBA)*:
  - Define case service types and locations

- *User (Health Worker)*
  - Search, Add, Edit, and Delete health case records

8.3.4. Use-Case Diagrams

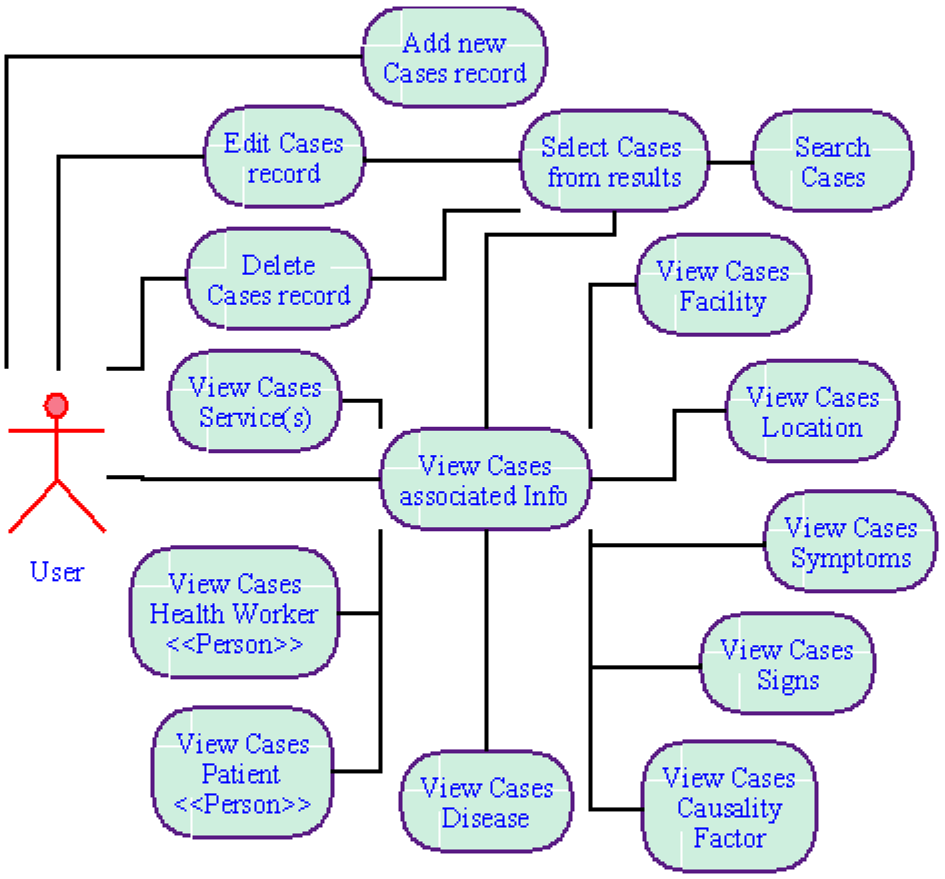

Figure 14 health Cases use case diagram

Table 34 description of use cases in Figure 1

Search Cases	Provide necessary and sufficient set of attributes for user to select or enter full or partial information; then apply search function to retrieve relevant dataset from database to be displayed in a table for user to select one or more records
Select Cases from result	Tabulated results should be presented with options to select one record for editing and one or several desired records for deleting. Also provide the option to add a record if desired. Mandatory elements displayed in table are the case uuid, case reported date, disease, patient name, gender, age group, health worker, facility, and location; while other values are optional but displayed if they are necessary for user to distinguish between two similar cases
Edit Cases record	First search for the record and select the desired record from result set; then make changes to the values and save
Add new record	By default the user is forced to search for the record and if record is not in result then allow to add a new case record; mandatory values for saving record are symptoms and location
Delete Cases record	First search for the record and select the desired record from result set; then apply delete function

Manage Cases Associated Information	This is to relate a case with existing disease, symptom, sign, causality factor, location, facility, service, patient, and health worker information by adding the information, editing already related information, or deleting already added information
Manage Cases Services	If there are no related services, then the process invokes the add service use case; if services already exist then user may select one of the service, which will invoke the edit service use case; or user may chose to delete an existing service from the list of related services
Manage Cases Location	If a location is not specified or already specified, the process invokes the search location use case for user to select a defined location or replace already related location, respectively; if desired location is not registered in DB then user may add the location first then select to relate the location with health case; removing a related location is not allowed as location is mandatory for a case record; see 7.1 for location use cases
Manage Cases Facility	If a facility is not specified or already specified, the process invokes the search facility use case for user to select an already defined facility from DB or replace the related facility with another value from DB, respectively; if desired facility is not registered in DB then user may add the facility first then select to relate the facility with health case; to remove an already related facility, simply chose to delete relationship; see section 7.4 for facility use cases
Manage Cases Disease	If a disease is not specified or already specified, the process invokes the search disease use case for user to select an already defined disease from DB or replace the related disease with another value from DB, respectively; if desired disease is not registered in DB then user may add the disease first then select to relate the disease with health case; to remove an already related disease, simply chose to delete relationship; see section 8.2 for disease use cases
Manage Cases Symptoms	If symptoms are not specified or already specified, the process invokes the search symptoms use case for user to select an already defined symptom from DB or replace the related symptoms with other values from DB, respectively; if desired symptom is not registered in DB then user may add the symptom first then select to relate the symptom with health case; to remove an already related symptom, simply chose to delete relationship; since symptom is a mandatory element of the health case record, at least, one symptom must be related for it to be a valid record, see section 8.2 for symptoms use cases
Manage Cases Signs	If signs are not specified or already specified, the process invokes the search signs use case for user to select an already defined sign from DB or replace the related signs with other values from DB, respectively; if desired sign is not registered in DB then user may add the sign first then select to relate the sign with the health case; to remove an already related sign, simply chose to delete relationship; see section 8.2 for symptoms use cases

Manage Cases Causality Factors	If causality factors are not specified or already specified, the process invokes the search causality factors use case for user to select an already defined causality factor from DB or replace the related causality factors with other values from DB, respectively; if desired symptom is not registered in DB then user may add the causality factors first then select to relate the causality factor with health case; to remove an already related causality factor, simply chose to delete relationship; see section 8.2 for symptoms use cases
Manage Cases Patient <<Person>>	If a patient (category = person) is not specified or already specified, the process invokes the search person use case for user to select an already defined person from DB or replace the related person with other values from DB, respectively; if desired person is not registered in DB then user may add the person first then select to relate the person with the health case; to remove an already related person, simply chose to delete relationship; see section 7.3 for person use cases
Manage Cases Health Worker <<Person>>	If a health worker (category = health worker) is not specified or already specified, the process invokes the search person use case for user to select an already defined person from DB or replace the related person with other values from DB, respectively; if desired person is not registered in DB then user may add the person first then select to relate the person with the health case; to remove an already related person, simply chose to delete relationship; see section 7.3 for person use cases

8.3.5. Main Scenarios

- Create a health case (add)
- Edit a health case
- Delete a health case
- Search a health case
- Edit case status history <<case history>>
- Lookup and associate patient <<person>>
- Look up and associate location <<location>>
- Lookup and associate facility <<facility>>
- Look up and associate services <<service>>
- Lookup and associate health worker <<person>>
- Lookup and associate disease <<disease>>
- Lookup and associate symptoms <<symptom>>
- Lookup and associate signs <<sign>>
- Lookup and associate causality factors <<causality factor>>

8.3.6. Data Storage (files and databases)

Table 35 Cases detail information

Attribute	***Data Type***	***Comment***
Universal Unique ID (uuid)	Integer/Varchar	Unique identifier for the record
Case date time	Date Time	Identifier pointing to the parent location
Patient person UUID	Integer/Varchar	Human readable name to id the location
Patient Full Name	Varchar	Health, Governance,
Health Worker person UUID	Integer/Varchar	Health location types: MOH, PHI,
Health Worker full name	Varchar	Additional human readable info
Facility UUID	Integer/Varchar	ISO 8440 international location code
Facility Name	Varchar	Geometric shape of the location e.g. circle, polygon, line, point
Location UUID	Integer/Varchar	A polygon will have a set of x coordinates X = (x1, x2, …, xn) e.g. latitudes
Location Name	Varchar	The same polygon will have equal number of y coordinates Y – (y1, y2,…yn); e.g. longitudes
Disease	Varchar	If 2 dimensional then a set of Z coordinates representing the height else zero Z=(0, 0, …,0)
Disease diagnose date time	Date Time	To identify when the record was created
Agent	Varchar	
Gender	Varchar	
Age	Decimal	
Age Group	Varchar	
Notes	Varchar	
Create by	Varchar	User or System creating the record
Create process	Varchar	How the record was created; e.g. web, mobile phone, the application, external
Modify date time	Date Time	Identify when record was last modified
Modify by	Varchar	User or System modifying the record
Modify process	Varchar	The method the record was modified
Deactivate date time	Date Time	Records will not be deleted from database only deactivated to maintain referential integrity

8.3.7. Associations

When collaborating with other objects such person, facility, service, address, or case the entry point for associating a location with these objects is through the search use case.

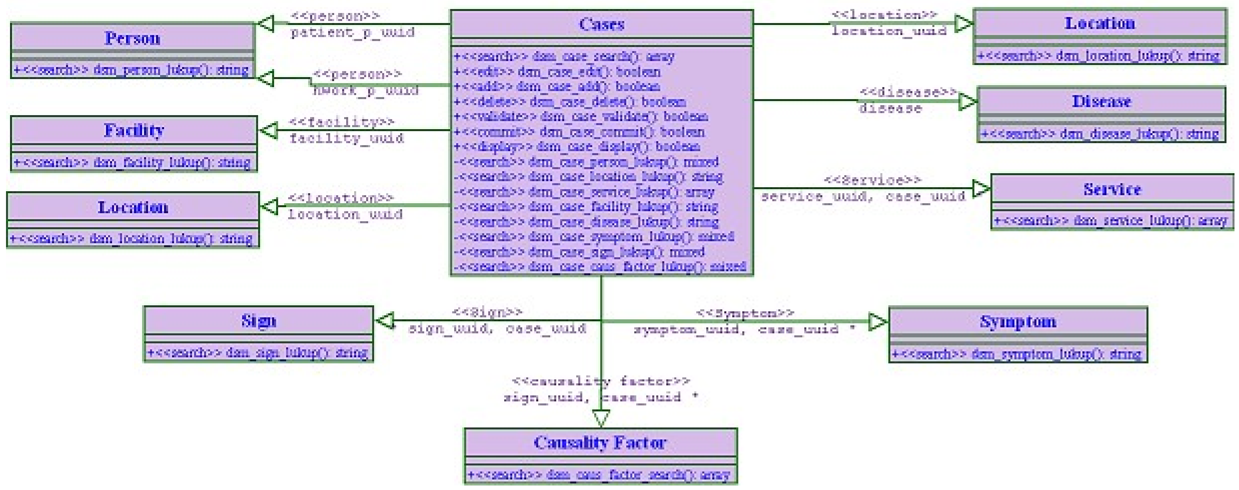
Person
+<<search>> dsm_person_lukup(): string
<<person>>
patient_p_uuid
<<person>>
hwork_p_uuid
Facility
+<<search>> dsm_facility_lukup(): string
<<facility>>
facility_uuid
Location
+<<search>> dsm_location_lukup(): string
<<location>>
location_uuid
Cases
+<<search>> dsm_case_search(): array
+<<edit>> dsm_case_edit(): boolean
+<<add>> dsm_case_add(): boolean
+<<delete>> dsm_case_delete(): boolean
+<<validate>> dsm_case_validate(): boolean
+<<commit>> dsm_case_commit(): boolean
+<<display>> dsm_case_display(): boolean
-<<search>> dsm_case_person_lukup(): mixed
-<<search>> dsm_case_location_lukup(): string
-<<search>> dsm_case_service_lukup(): array
-<<search>> dsm_case_facility_lukup(): string
-<<search>> dsm_case_disease_lukup(): string
-<<search>> dsm_case_symptom_lukup(): mixed
-<<search>> dsm_case_sign_lukup(): mixed
-<<search>> dsm_case_caus_factor_lukup(): mixed
<<location>>
location_uuid
Location
+<<search>> dsm_location_lukup(): string
<<disease>>
disease
Disease
+<<search>> dsm_disease_lukup(): string
<<Service>>
service_uuid, case_uuid
Service
+<<search>> dsm_service_lukup(): array
Sign
+<<search>> dsm_sign_lukup(): string
<<Sign>>
sign_uuid, case_uuid
<<Symptom>>
symptom_uuid, case_uuid *
Symptom
+<<search>> dsm_symptom_lukup(): string
<<causality factor>>
sign_uuid, case_uuid *
Causality Factor
+<<search>> dsm_caus_factor_search(): array

Figure 15 Cases object and the inherited and associated objects

Table 36 Other GUIs quasi modally accessing the location object

Interface Name	***Remarks***
▪ Person add and edit form (web interface)	A health worker is associated with a location; e.g. a Public Health Inspector is assigned a jurisdiction.
▪ Facility add and edit form (web interface)	A hospital is associated with a location; e.g. Kuliyapitya base hospital is situated in the Kuliyapitiya MOH division
▪ Service add and edit form (web interface)	A service such as investigating a patient can be carried out in a particular Public Health Inspector area.
▪ Case add and edit form (web interface)	A case has to be identified with a location to calculate the disease densities.
▪ Address add and edit form (web interface)	An address may belong to a particular governance area such as a tribunal or village or a district

8.3.8. Implementation Notes

- Install the Hardware (Server and Network) Software (operating system and web server)
- Setup the alias directory system to access the application with a web browser over the internet
- Test the accessibility from an external personal computer connected to the internet
- Identify the set of location categories to be used in the implementation and set the category values
- Identify the set of types (sub categories) under each category and their hierarchy then enter those values with corresponding category

- Setup an indexing system for the location universal unique identifier
- Open the search form to ensure the categories and related types are displayed

10. Schedule and Milestones

Date	Milestone
Apr 4, 09	Delivery of MySQL Database Schema
Apr 6, 09	Review of schema and feedback
Apr 10, 09	Alpha Release of BSM
Apr 15, 09	Packaged version for IITM – IITM integration starts
Apr 17, 09	Final Release of BSM
Apr 22, 09	QA testing and handover of BSM
Apr 25, 09	Requirements and Design of Messaging Module
May 5, 09	EDXL and CAP integration
May 10, 09	Alpha Release of MM
May 15, 09	Stable Release of MM
May 16, 09	Project Handover

11. References

[1] Geetha, G., Hewapathirana, R., Waidyanatha, N., and Weerakoon, P. (2008). User requirement Specifications for the Real-Time Biosurveillance Program; web copy - http://lirneasia.net/wp-content/uploads/2009/02/user-requirements-v10.pdf (consulted Dec 2008)

[2] Nuwan Waidyanatha (2009) Software requirement Specifications for RTBP Database.

[3] Sahana Wiki – http://www.sakana.lk/docu/

[4] Fielding, R. T. (2000) Architectural styles and the Design of network-based software architectures, doctoral dissertation, University of California – Irvine, USA. Web copy - http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf