KINGDOM OF LESOTHO

SYSTEM DESIGN & PROTOTYPING DOCUMENT

Ministry of Social Development

Submitted By

September 2021

Table of Contents

Table of Figures iii
Abbreviations iv
Overview 1
Use Cases 1
Major Use cases 1
Use Case Diagram 2
Design Overview 3
Software architecture 3
Web Application File Structure 4
Static Content 4
REST API 4
Hardware architecture 4
Server Hardware. 5
Infrastructure 5
Data Design 5
User Interface Design 6
User Login 6
Generic Dashboard 7
Case Capture form 7
Reports 10
QA 11
External Interfaces and Integrations. 11
Non-Functional Requirements 12
Performance and structural Requirements 12
Design Constraints 12
Approval Form 13

Table of Figures

Figure 1: Use Case Diagram 3
Figure 2: Infrastructure Map 5
Figure 3: User Login Interface 6
Figure 4: User Generic Dashboard 7
Figure 5: Incoming Call view 7
Figure 6: Follow Up case search form 8
Figure 7: Reporter Details Form 8
Figure 8: Case Form 9
Figure 9:Case Action Form 9
Figure 10: Case Update trail 10
Figure 11:Reports Preview 10
Figure 12:Report Export Formats 11
Figure 13: QA Form 11

Abbreviations

VAC – Violence Against Children
CRM - Customer Relations Management
REST - Representational State Transfer
API – Application Programming Interface
SIP - Session Initiation Protocol
CRUD - Create, Read, Update and Delete
RAM – Random Access Memory
TB - Terabyte
GB - Gigabyte
GHz – Gigahertz
CPU - Central Processing Unit
QA – Quality Analysis
CSS – Cascading Style Sheets

Overview.

The System Design Document here describes how the functional and nonfunctional requirements captured in the requirements document transform into more technical system design specifications for the upgrade of the Lesotho Child helpline. It presents a number of different architectural views to depict the different aspects of the system.

This document informs the system development process and gives the development team guidance on the architecture of the system based on the compiled requirements for the Helpline System.

This document in intended both for the project management & development team and the client. With this document, the client can have the solution development progress and prepare for the system delivery. The consultant and its team is guided by the architecture and designs defined in this document.

Use Cases

Major Use cases

The purpose of the use-case view is to give additional context surrounding the usage of the system and the interactions between its components. For the purposes of this document, each component is considered a use-case actor. The table below lists the actors and gives a brief description of each in the overall use case context of the system.

1.	Client	The client contacts the helpline through the following Medium: Call **Call Termination Outcomes** 1. Client Requests for Support => Agents picks and Responds 2. The Hotline is busy =>A message of wait is sent to a client 3. The call fails due to Network Failure => Please Dial Again Description of Outcomes #1 The client did not get what they wanted #2 The hotline is busy message displayed #3 Network Failures occur during operation Termination/Cancellation of call.
2.	Social Worker	The Agent Receives Calls for request of information or case report from the client. **Termination Outcomes**

		1. Call Rings Via Popup => Social Worker picks and Responds 2. Issue requires Escalation => Social Worker Escalates to respective Supervisor 3. Issue can be Resolved => Social Worker Provides Solution
3.	Supervisors	The Supervisor Receives Escalation Notification from the Social Worker. The supervisor Appraises Social Workers. **Termination Outcomes** 1. Notification Received => Supervisor Solves or Escalates 2. Appraises Social Worker => Submit Appraisal Feedback to Social Worker.
4.	Administrator	1. Manage system users. 2. Manage system configurations.

Use Case Diagram

The most common use-cases are outlined and illustrated using UML use-case diagrams and sequence diagrams to clarify the interactions between components.

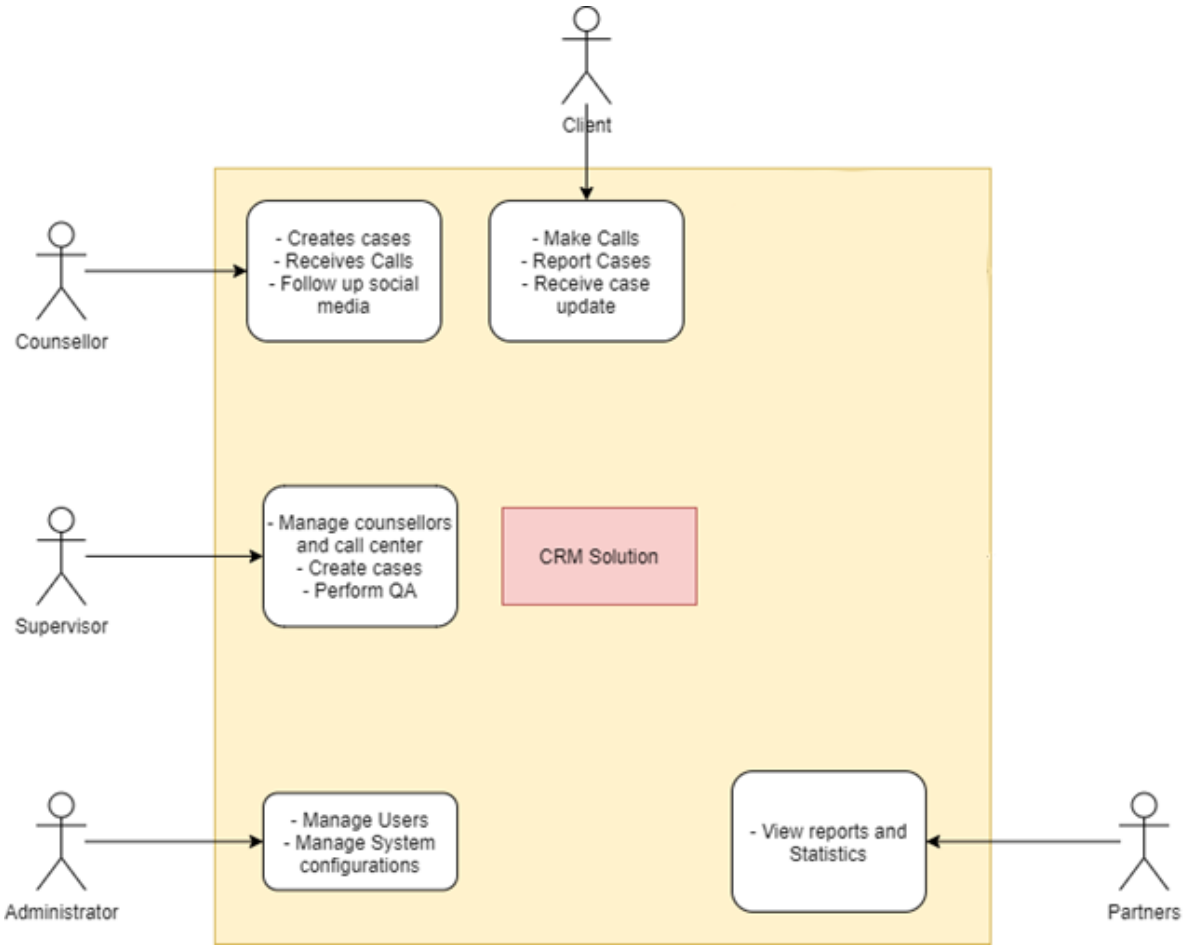


Figure 1: Use Case Diagram

Design Overview

Software architecture

The Solution follows the JAMStack architecture pattern (https://en.wikipedia.org/wiki/Jamstack), which provides the following advantages:

a) Decouples the front-end from the backend - this is achieved by generating the User Interface (UI) on the Client-Side side using JavaScript (instead of generating the UI on the Server-Side)
b) Leverages use of Static Templates to generate the front-end
c) Enables development of a single REST API specification for use by both Web and Android Applications.

The following is an overview of the software stack:

Component	Description
CSS	This contains information on font-color, font-size, padding, margins background-color and borders. This enables the application to have a uniform look and feel though-out all the screens
Static Templates	The User Interface screens are defined as Static templates,

	which are populated at runtime.
Rendering Engine	The Rendering Engine evaluates the REST API response and then generates the User Interface using the relevant Static Template
SIP.js WebRTC Phone	This provides a virtual phone that can be used to make and receive calls
Nginx Web Server	This provides the HTTP/S Gateway to the REST API
REST API	This provides a set of CRUD (Create, Read, Update, Delete) URL endpoints. Each CRUD is mapped to a single database table
Database	This is the persistent store of application data

Web Application File Structure

The Web Application is hosted on the Nginx server and it serves 2 purposes:

1. Delivery of static content (templates, CSS, JS rendering engine and WebRTC Sip client) that comprises the Single Page Web application.
2. REST API

Both the Static Content and REST API are contained in a single folder named **helpline.** The following is file structure of the helpline folder:

Static Content

/helpline/index.php (Single Page Application Entry Point)
/helpline/js/sip-0.20.0.js (WebRTC SIP client)
/helpline/js/*.js (User Interface rendering engine)
/helpline/app/*.js (User Interface static templates)

REST API

/helpline/api/index.php (REST API entry point)
/helpline/api/session.php (Session Management Engine)
/helpline/api/rest.php (REST API engine)
/helpline/api/model.php (CRUD endpoints definitions)

The application is customized mainly by editing files in following 2 locations to implement screens and CRUD logic:

1. /helpline/app/*.js (Front End Screens)
2. /helpline/api/model.php (CRUD Logic)

Hardware architecture

These are the minimum hardware requirement for optimum operation of the solution. It also takes into consideration the different infrastructural components necessary for the functioning of the call center in general.

Server Hardware.

These are the minimum hardware requirements for the server running the call center CRM, database and call modules.

RAM

- Asterisk - 4GB RAM
- Database - 16 GB RAM

Storage (HDD/SSD)

- Operating system partition - 100 GB
- Database 100 GB
- Call Recording 1 TB

CPU

- 4 cores each core 2.0 GHz

Infrastructure

These is a complete hardware infrastructural mapping of the call center. It includes terminations by telecom service providers, power provision & backup and network configuration.

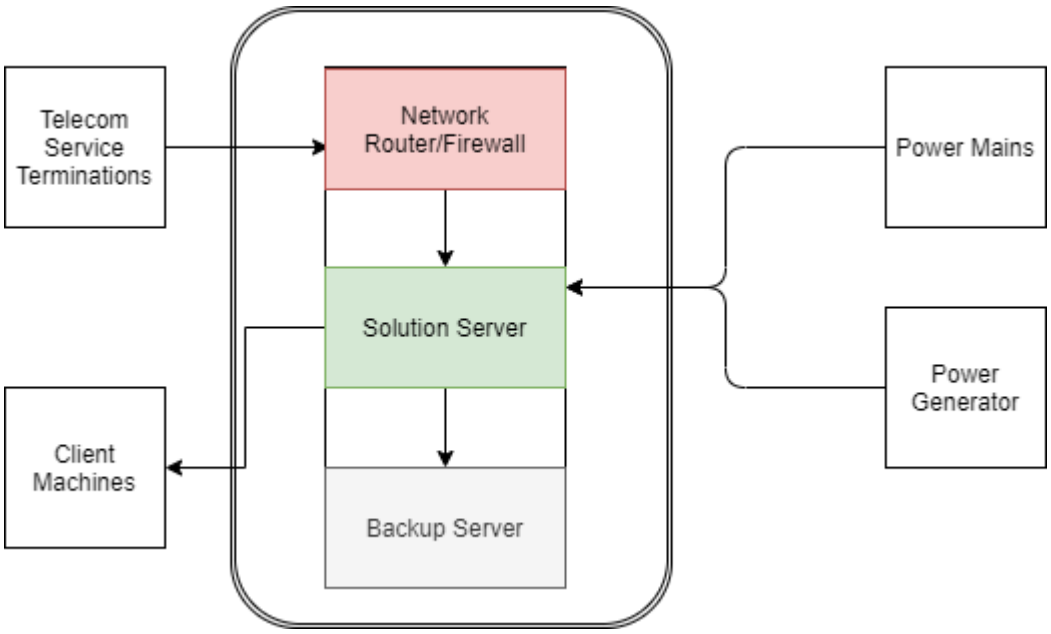


Figure 2: Infrastructure Map

Data Design

TABLE NAME	DESCRIPTION
session	This holds information on the active sessions
Auth	This identifies a user in the system
Contacts	This stores information about a person. The following tables have the contact_id foreign key: auth, reporter, client, perpetrator

Case	This identifies a unique case in the system and its current status
Activity	This keeps an audit trail of case
Reporter	This identifies the person who reported the case
Client	This identifies the victim(s) in the case
Perpetrator	This identifies the offender(s) in a case
Call	This stores call information, such as datetime, phone number, extension, wait-time, talk-time, hangup-status
Category	This holds the definition of various enumerated types used in the case form. For example: age-group, gender, location, case-status, etc
QA	This holds results of a quality-assessment evaluation

User Interface Design

This is a presentation of the user interface of the solution as designed or as shall be designed in line with the user levels, processes and logical data flows.

Some of the major parts of the system interfaces are as listed below:

User Login

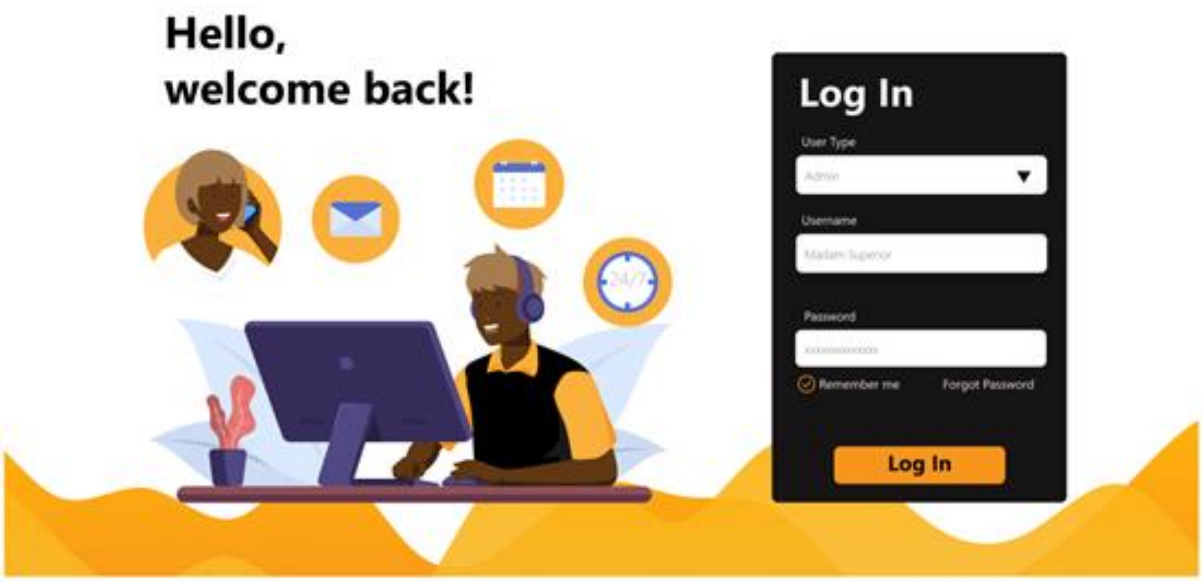


Figure 3: User Login Interface

Generic Dashboard

Figure 4: User Generic Dashboard

Case Capture form

Incoming Call

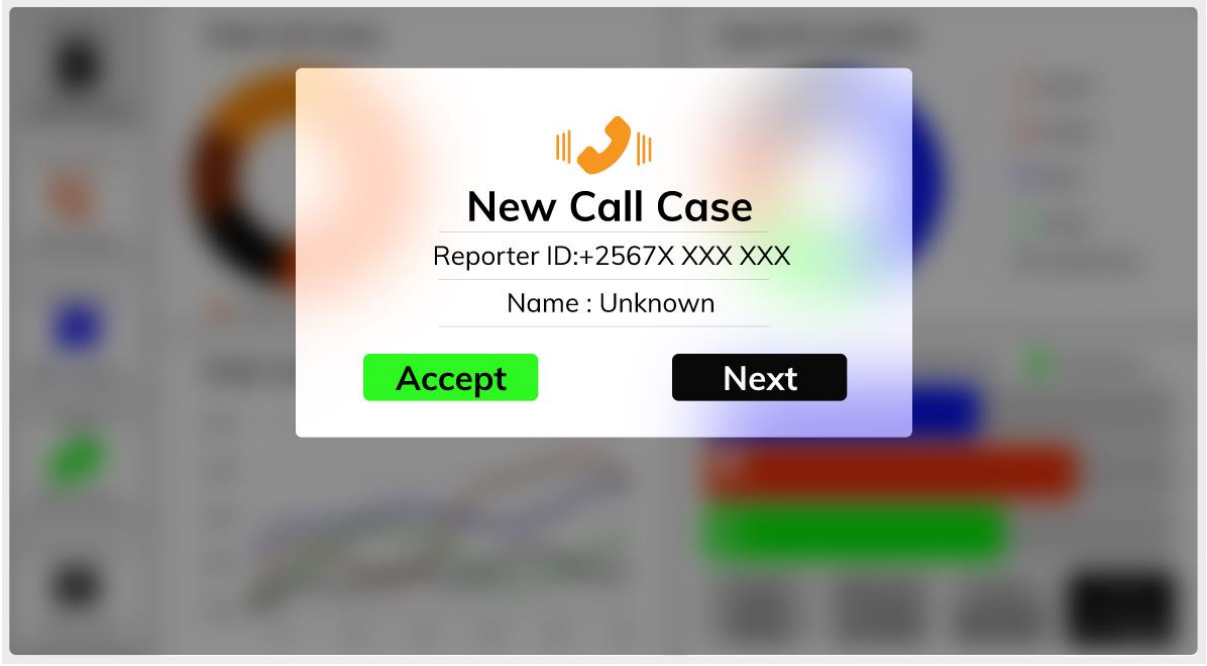


Figure 5: Incoming Call view

Follow Up Case Search

Figure 6: Follow Up case search form

Reporter Details

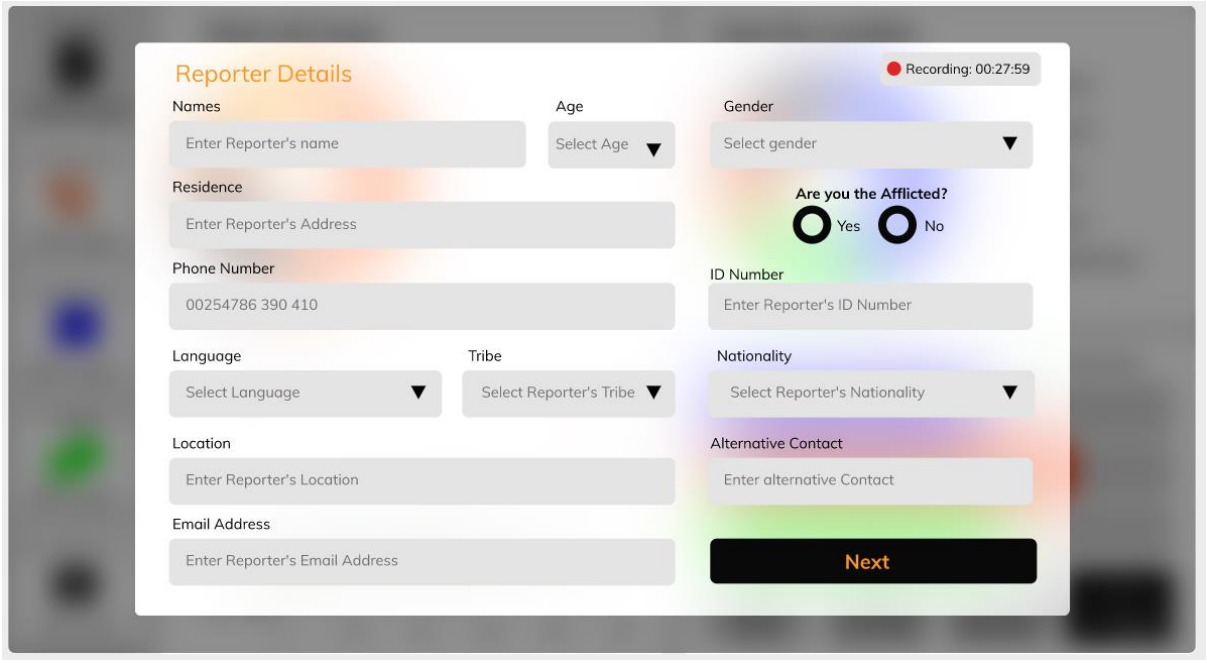


Figure 7: Reporter Details Form

Case Details

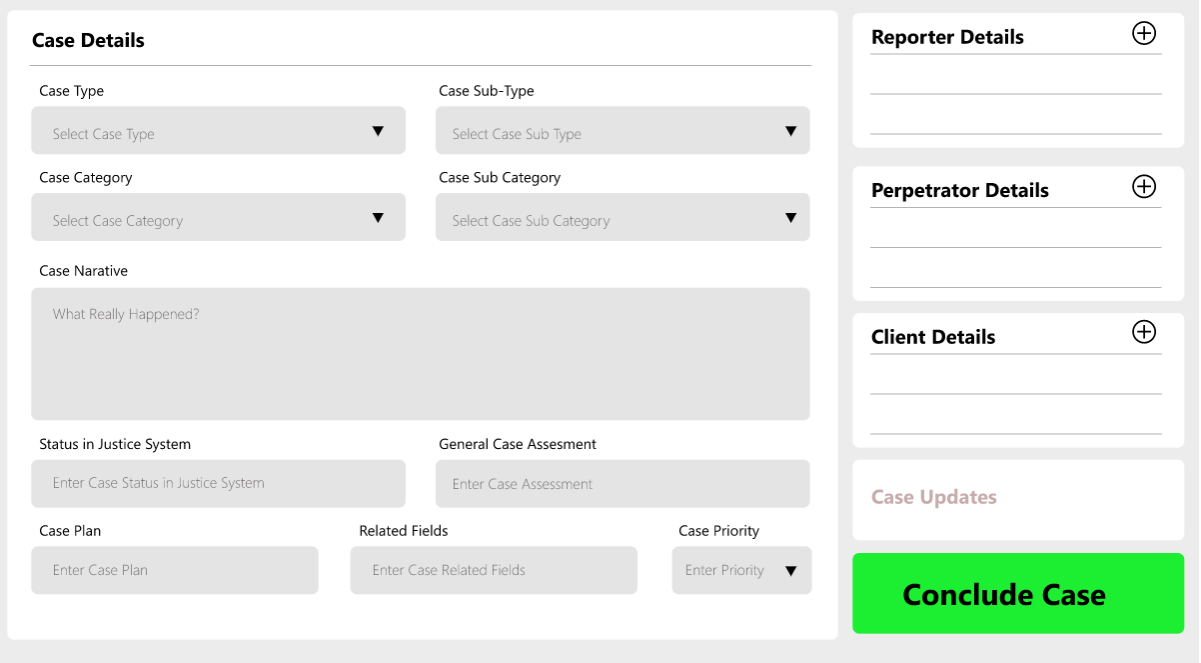


Figure 8: Case Form

Case Submission

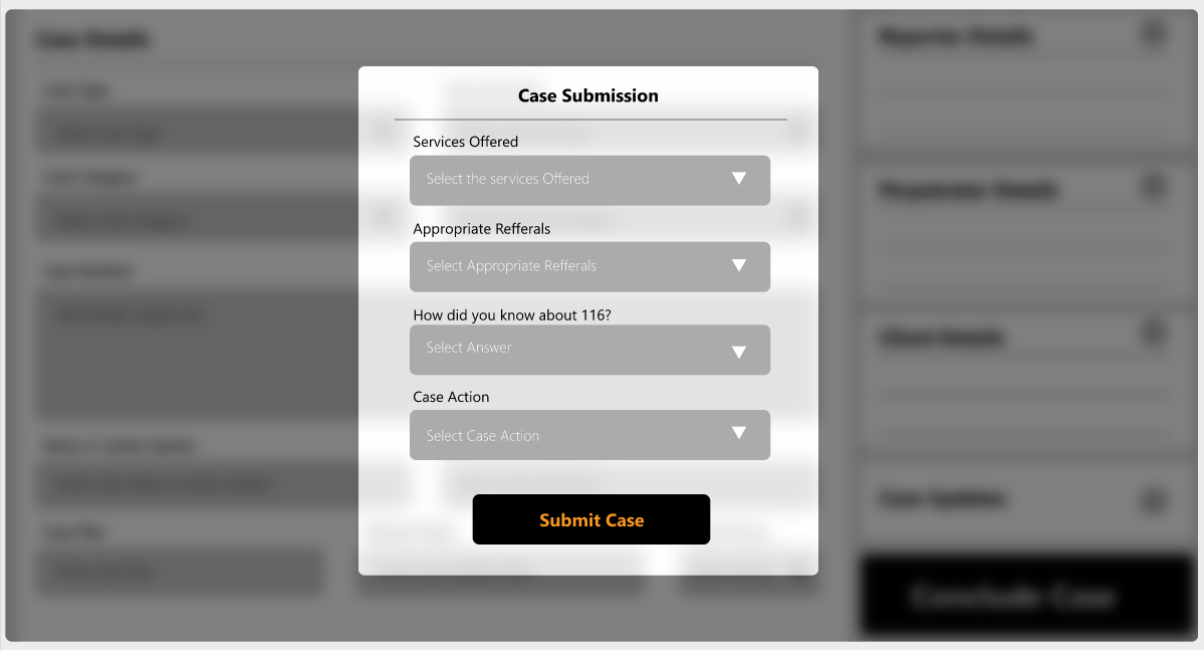


Figure 9:Case Action Form

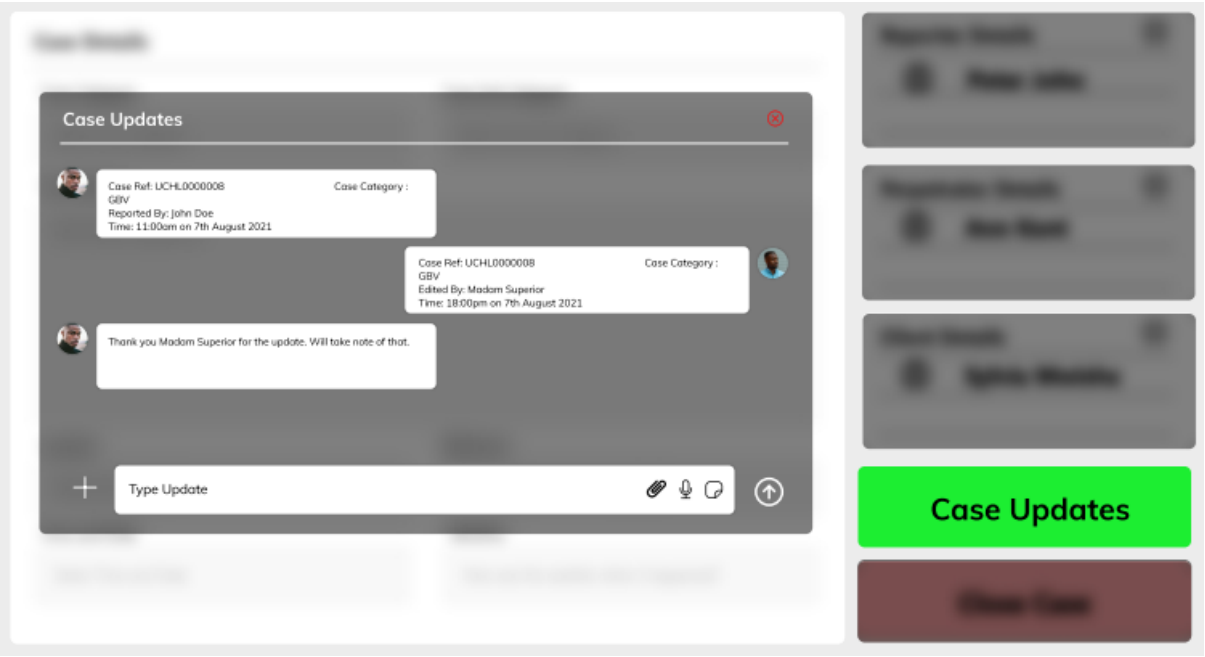
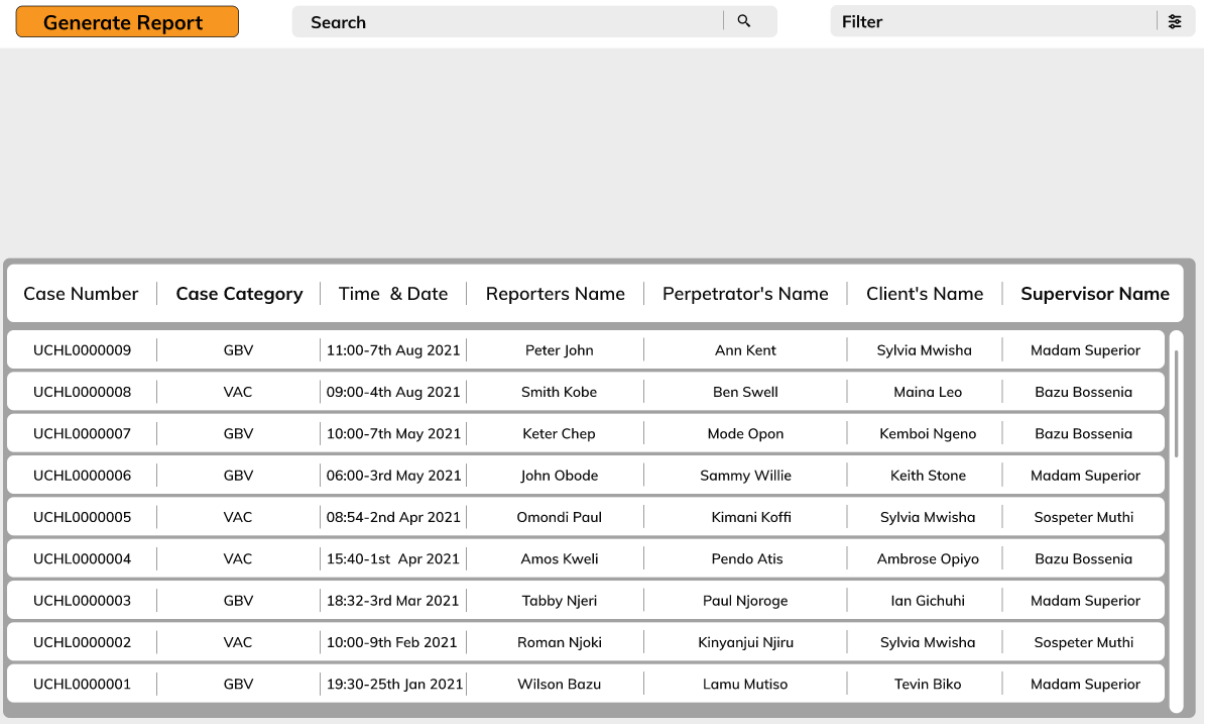


Figure 10: Case Update trail

Reports



Case Number	Case Category	Time & Date	Reporters Name	Perpetrator's Name	Client's Name	Supervisor Name
UCHL0000009	GBV	11:00-7th Aug 2021	Peter John	Ann Kent	Sylvia Mwisha	Madam Superior
UCHL0000008	VAC	09:00-4th Aug 2021	Smith Kobe	Ben Swell	Maina Leo	Bazu Bossenia
UCHL0000007	GBV	10:00-7th May 2021	Keter Chep	Mode Opon	Kemboi Ngeno	Bazu Bossenia
UCHL0000006	GBV	06:00-3rd May 2021	John Obode	Sammy Willie	Keith Stone	Madam Superior
UCHL0000005	VAC	08:54-2nd Apr 2021	Omondi Paul	Kimani Koffi	Sylvia Mwisha	Sospeter Muthi
UCHL0000004	VAC	15:40-1st Apr 2021	Amos Kweli	Pendo Atis	Ambrose Opiyo	Bazu Bossenia
UCHL0000003	GBV	18:32-3rd Mar 2021	Tabby Njeri	Paul Njoroge	Ian Gichuhi	Madam Superior
UCHL0000002	VAC	10:00-9th Feb 2021	Roman Njoki	Kinyanjui Njiru	Sylvia Mwisha	Sospeter Muthi
UCHL0000001	GBV	19:30-25th Jan 2021	Wilson Bazu	Lamu Mutiso	Tevin Biko	Madam Superior

Figure 11:Reports Preview

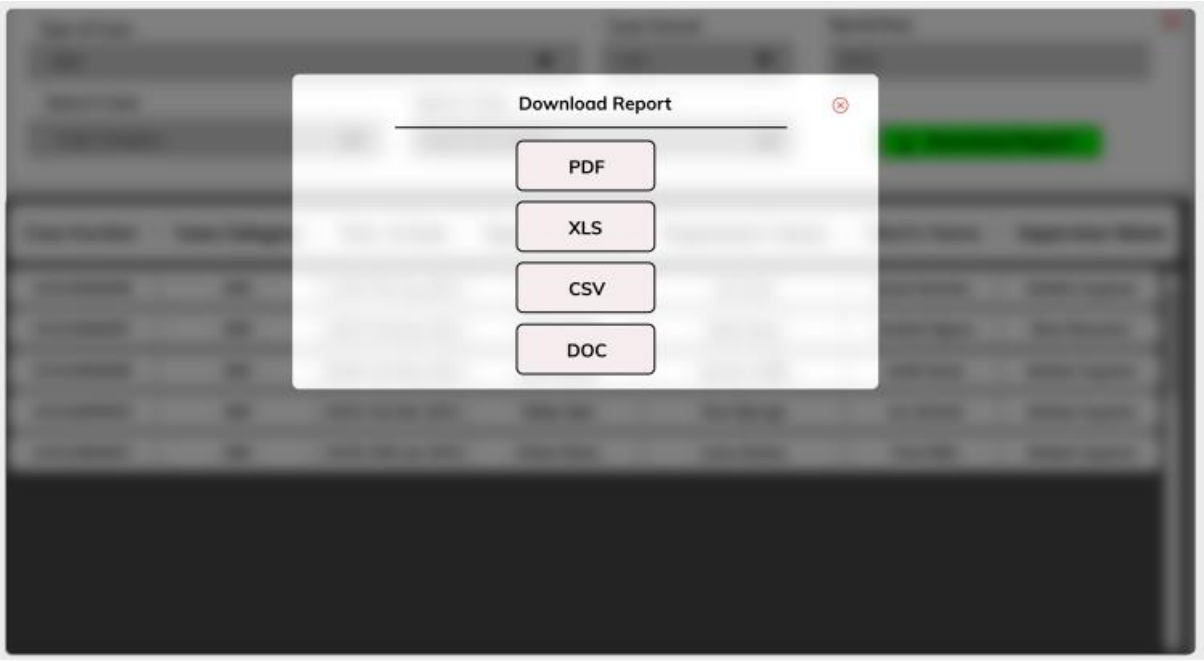


Figure 12:Report Export Formats

QA

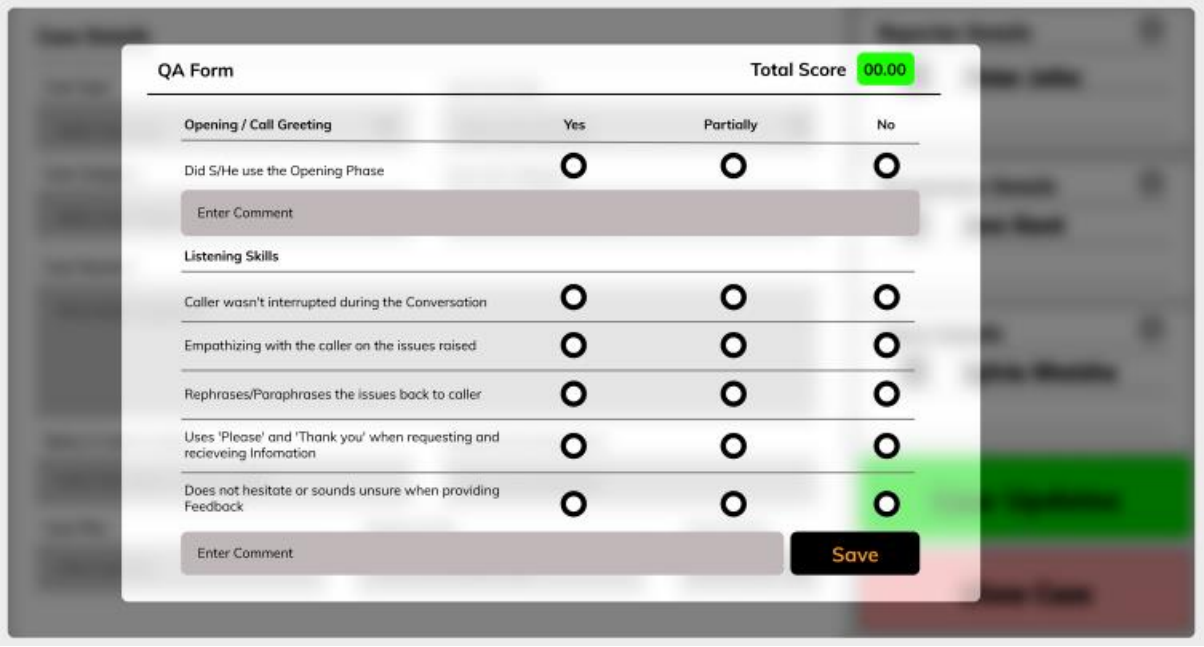


Figure 13: QA Form

External Interfaces and Integrations.

The helpline shall have custom integrations to external systems to share data in different ways. It shall also have a universal API for querying cases with different parameters as defined in the requirement specification.

Non-Functional Requirements

Performance and structural Requirements

The solution shall not limit the number of concurrent user access only to the extent of the system resource capacity which shall be considered during the development and deployment process.

The solution shall design and implement security protocols for call and case data having in mind data confidentiality in remote backup, restore and recovery plans, procedures and systems.

The hosting platform shall have deployed enhancements and Secure Sockets Layer (SSL) certificate for added security.

The system shall not be limited to in-office users but also accessible via VPN for remote working.

Design Constraints

The software solution shall take advantage of and be developed on open source technologies, tools and standards.

Approval Form

By signing this document, I acknowledge that I have received stated deliverables to the agreed quality levels.	
	Signature:
	Date:
	Signature:
	Date:
	Signature:
	Date: