Technical Specification - Report & Respond SMS Tool

Last revision: 14 Jul 2022

Authors:

Tom Feltwell, Ian Johnson
Open Lab, Newcastle University
{tom.feltwell | ian.johnson2}@newcastle.ac.uk

Technical Review:

James Hodge
Open Lab, Newcastle University
j.hodge1@newcastle.ac.uk

Summary

This document describes the technical specification for the *Report & Respond* SMS tool that was designed as part of the Collective Crisis Intelligence (CCI) project, which was run between October 2021 and June 2022. The project was run by Nesta, in collaboration with the Nepal Red Cross and Cameroon Red Cross, and the International Federation of Red Cross and Red Crescent Societies (IFRC) Solferino Academy, and Open Lab, Newcastle University. The Report & Respond tool was designed for use by the Cameroon Red Cross, to facilitate the dissemination of official responses to mis-information and rumours around Covid-19. This document describes the design and a detailed proposed technical implementation of the Report & Respond tool.

Intended Audience

This document describes the technical design and proposed implementation for the Report & Respond SMS tool, designed as part of the Collective Crisis Intelligence project. This document is intended for developers and system architects who will be implementing the tool. This document details initial scoping and design work that has been carried out to implement the tool, which consists of a proposed system architecture, detailed breakdown of all technical components and an estimate of tasks and associated time and financial costs to implement the system.

Acronyms/Glossary

Term	Definition
API	Application Programming Interface
CCI	Collective Crisis Intelligence (Project)
CM	Conversation Manager
CRC	Cameroon Red Cross
IFRC	International Federation of Red Cross and Red Crescent Societies
RC	The Red Cross
R&R	Report & Respond
Rumour	Piece of community feedback could be classified as misinformation/incorrect
SMS	Short Messaging Service, a.k.a text message
Twilio	Customer communication service, allowing digital, SMS and telephone.

System Overview

Report & Respond (R&R hereafter) is a volunteer tool designed for use by volunteers in the field when gathering community feedback. It is designed to allow a volunteer to send a piece of community feedback via SMS to the Red Cross, and in return receive the official response to the community feedback, that the volunteer should disseminate to the community.

As part of this tool, two machine learning models have been developed: a classification model for rumour matching to existing categories, and a clustering model for identifying emerging rumours.

This document focuses on the R&R SMS tool, that allows volunteers to pass rumours via the classification model[1] and receive a response. In use, R&R will appear as a Cameroon registered phone number that anyone can send an SMS message to. Based on the contents of the message and whether this is known by the model, there are two conversational flows that can be entered into.

1. The rumour is known by the model so an official response to the rumour is sent to the volunteer via SMS. They are subsequently asked some contextual questions about whether they think the response is correct, whether they have used it in the field.
2. The rumour is not known by the model. They are asked some contextual questions about the rumour, such as whether they hear it often and what a suggested label might be.

In the conversation flows there are some optional contextual questions which are put to the volunteer, which they can answer if they chose. This information is used to inform the Cameroon Red Cross (CRC) team how the data is being used by the volunteers. This data is stored in a database and sent to the team at regular intervals (e.g. weekly) as a report for their analysis. R&R is designed for use by Red Cross volunteers in Cameroon, and as such SMS was selected due to the high availability of the SMS service across Cameroon (wherever mobile phone signal is available, SMS is available). There are a number of other factors to consider for the final implementation (such as cost of SMS messages) which are discussed in the sections below.

French Language

Cameroon is predominantly French speaking, so the system is intended for use by French speakers and to process French language. It is expected that rumours input by the volunteers will be in French, and the CCI model is already trained to work on French language input. The tool is therefore written entirely in French, so all SMS messages sent will be in French.

[1] Subsequently referred to as the CCI model.

Network connectivity

A key consideration for the success of R&R is the ability to cope with loss of network connectivity in remote locations. SMS provides a resilient way to handle loss of connectivity, as all phones feature a resend SMS feature, which is available when an SMS is unable to be sent due to network connectivity. This allows the volunteer to send the message, which will effectively be queued (depending on the OS of the phone being used) and sent when connectivity is re-established.

Given that the CCI model cannot be hosted locally on volunteers' phones, there is always a reliance on some network connectivity. Using the telephony network provides the widest coverage, but has a higher associated cost for usage compared to mobile data. Arguably the coverage of SMS is better than mobile data, so SMS has been chosen to give the greatest coverage. There is discussion further in this document (link) about the trade offs between using WhatsApp or SMS to carry messages between the system and the volunteers. The technical architecture described in this document is based on use of the SMS network.

The CCI Machine Learning Model

The CCI model has already been developed as an independent part of the project, and the codebase is available for implementation in a public repository (GitHub). This document describes the R&R tool, which will implement and use the CCI model for its core functioning.

The model is trained on a data set comprising 1021 observations, which consists of existing rumours and the labels that have been applied to them. This data was generated prior to this project, and was created by the IFRC and CRC staff, who performed manual pairing of rumours and labels. When a new rumour comes through it is matched to the most semantically similar label.

The R&R tool uses the trained CCI model to evaluate the input from volunteers. When a rumour is submitted via the SMS tool, it is passed via the Conversation Manager and Model API to the model, where it will determine if it matches an existing label. If it does, it will send details of the match with an appropriate response (the label category, the confidence score and the official Red Cross response) back. If it does not match, it will send a message back that it is not recognised Both of these type responses are handled by the Conversation Manager to trigger the next appropriate conversation flow.

In the conversation flows described below, volunteers are asked to provide suggestions of labels that could be applied to new, unidentified rumours. These labels will not be directly input into the model, and our proposed process is that they are collated and submitted to the CRC once per week, where the team can consider whether the labels are useful to be included in the model. Furthermore, rumours that are not recognised by the model are sent to a pool of unmatched rumours that are used by the clustering model, which groups new rumours together based on similarity to identify new clusters that a new label can be generated for. This process is outside the functioning of the R&R tool.

The remainder of this document describes the overall product design, and then a proposed system architecture with full detail about each component.

Product overview

As an overview of the entire product, the below diagram details how a volunteer submits a rumour and how this rumour is processed.

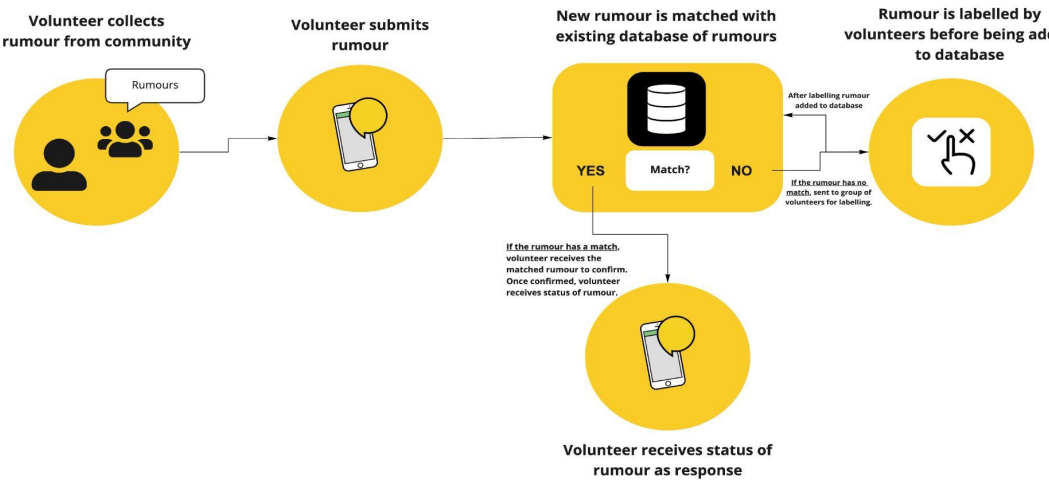

Figure 1. Broad product overview showing how volunteers engage with the model

Expected usage characteristics

Use case	**Expectation**
Volunteer sends in a common rumour to receive official response	A volunteer submits a rumour they have heard, as plain text. They receive an SMS response saying the rumour is known, containing the official response they should tell to the community. They are then asked about how they have used this response.
Volunteer sends in a rumour they have never heard before	Volunteer receives an SMS response saying the rumour is unknown, and is then asked via multiple SMS messages about the rumour.

R&R System Architecture

Based on the above product design, we propose the following system architecture for R&R:

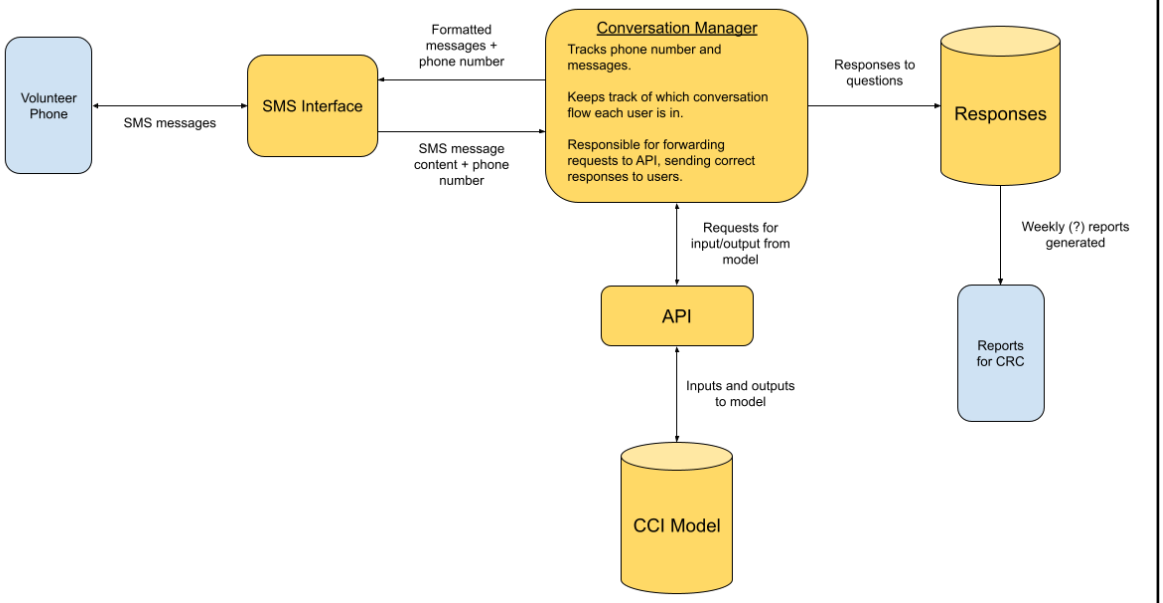

Figure 2. System architecture of the R&R tool showing processes from volunteer SMS input to model queries.

The following table provides an overview of each system component and their functions. A full breakdown and description of each component can be found in System Component Details on the following pages.

Component	**Details**
SMS Interface	Provides routing of messages into the SMS network. It is also a communication service allowing programmatic sending and receiving of SMS messages. This would be used to receive messages on a phone number and direct them to the API of a web server.
Conversation Manager	Required to control conversations with each user, in order to keep track of where in the conversational flow each user is, route their messages to the model where required, and send the correct response SMS to the user.
Model API	API that can directly interact with the model using standardised inputs and outputs. This will allow the conversation manager to query the model and receive appropriate responses.
CCI Model	Machine learning model developed to cluster new rumours, and retrieve labels and responses for identified rumours.
Volunteer Response API	API to store inputs to contextual questions in the conversation flow. Responses from volunteers are then stored in a Response Database for further analysis. We propose the responses are output weekly in the form of an email to the CRC team, so they can monitor how the tool is being used.

Web server / hosting	API will need hosting on a server, which may be the same hosting as the model. Reporting of SMS responses will also need to be done on the web server.
Volunteer Responses Database	To store SMS responses to questions that are not processed by the model (e.g. ‘Have you used this response in the field?’).

Conversational flows

As R&R is a purely SMS user experience for the volunteers and there is a small set of use cases, it can be thought of as a conversational flow which the user moves through. To this end, we have defined these conversational flows below, which detail the expected behaviour and decision points throughout each flow. Note that the contextual questions are pre-defined for each flow, and would therefore be hard coded into the conversation flow.

Name	**Flow description**
Known rumour	A rumour is submitted by a volunteer via SMS. The rumour is processed internally by the model and it determines it is a known rumour. The official RC response is sent to the volunteer. Optional contextual questions follow in the remaining flow. 1. “Is the response correct?” (Yes/No) Yes: “Have you used this in the field?” No: “Tell us why it is not correct” The flow concludes thanking the volunteer for their usage.
Unknown rumour	A rumour is submitted by a volunteer via SMS. The rumour is processed internally by the model and it does not recognise the rumour. A response is sent to the volunteer stating the model does not recognise the rumour. A set of contextual questions follow: 1. “Do you think the RC should investigate this rumour?” (Yes/No) No: [End of flow] Yes: [Continue] 2. “Have you heard this from the community before?” 3. “Can you give a short description of this rumour? It should summarise what the rumour is about. We will use this suggestion to decide an official response. Please do not use this response with the community.” The flow concludes thanking the volunteer for their suggestion.

See flow diagram on the following page.

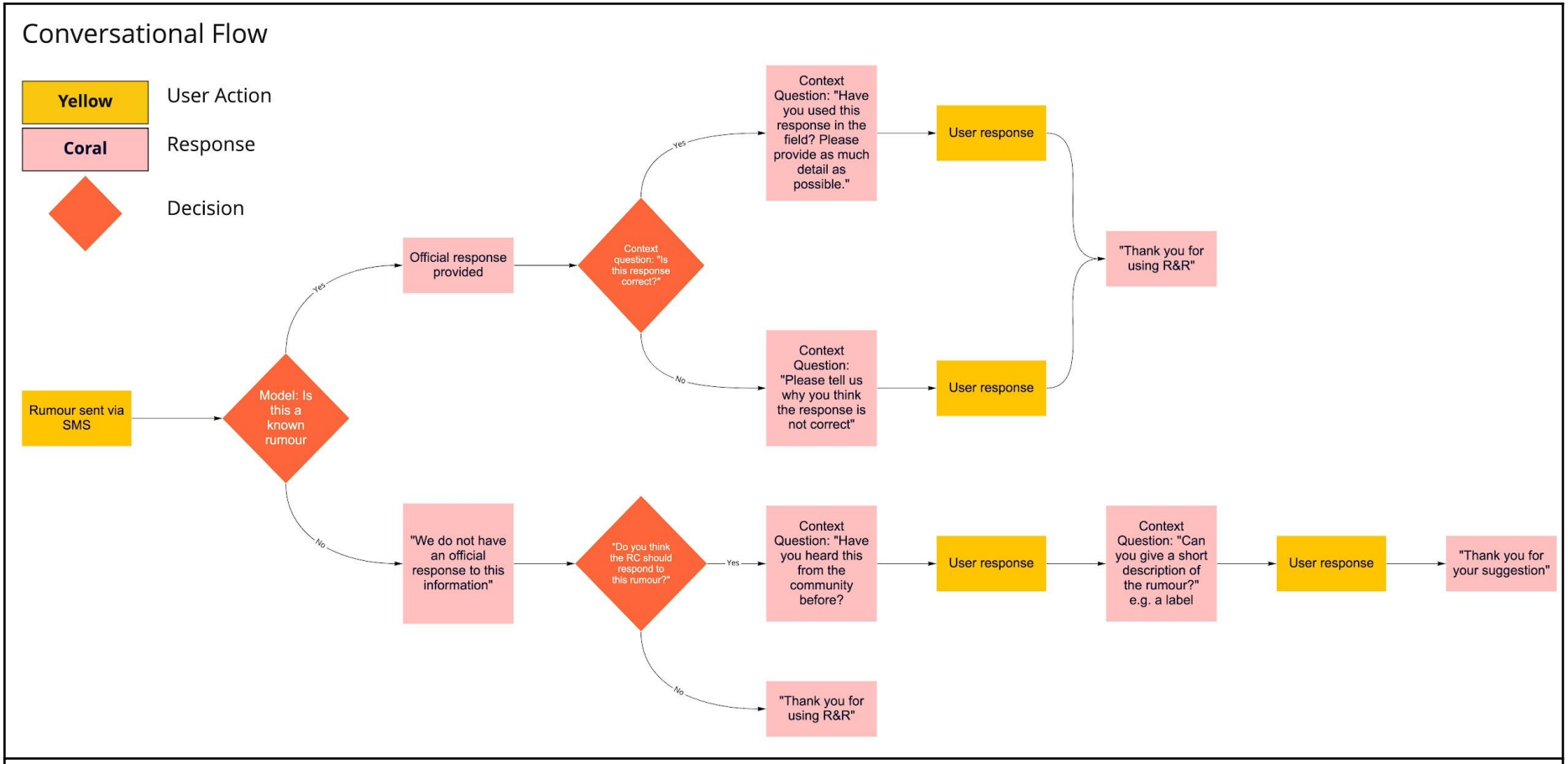

Figure 3. Conversational flow through R&R.

UI Prototypes

A prototype of Report & Respond was built in February 2022 with Adobe XD that shows the whole UI and all key interactions.

Links to prototype files	
XD share link (EN)	https://xd.adobe.com/view/290a3c25-70e0-4e9e-ab01-0c931aaf8289-7c59/
XD share link (FR)	https://adobe.ly/3IWtl8N
XD File EN & FR	GitHub link

Proposed implementation

In this section we detail each of the system components in our proposed R&R system architecture. We also provide concrete suggestions for implementation, such as specific tools and languages to use and how they might be configured to interoperate.

In sum:

SMS is used as the message service which is handled by the service Twilio. Twilio passes messages to a Conversation Manager. The Conversation Manager tracks all conversations and routes messages to and from the model via an API. The Conversation Manager formulates responses (using data from the model as required) and sends this back to the volunteer as an SMS via Twilio.

See the system architecture diagram in the System Overview section. The system layers and data movement can be seen in Figure 4:

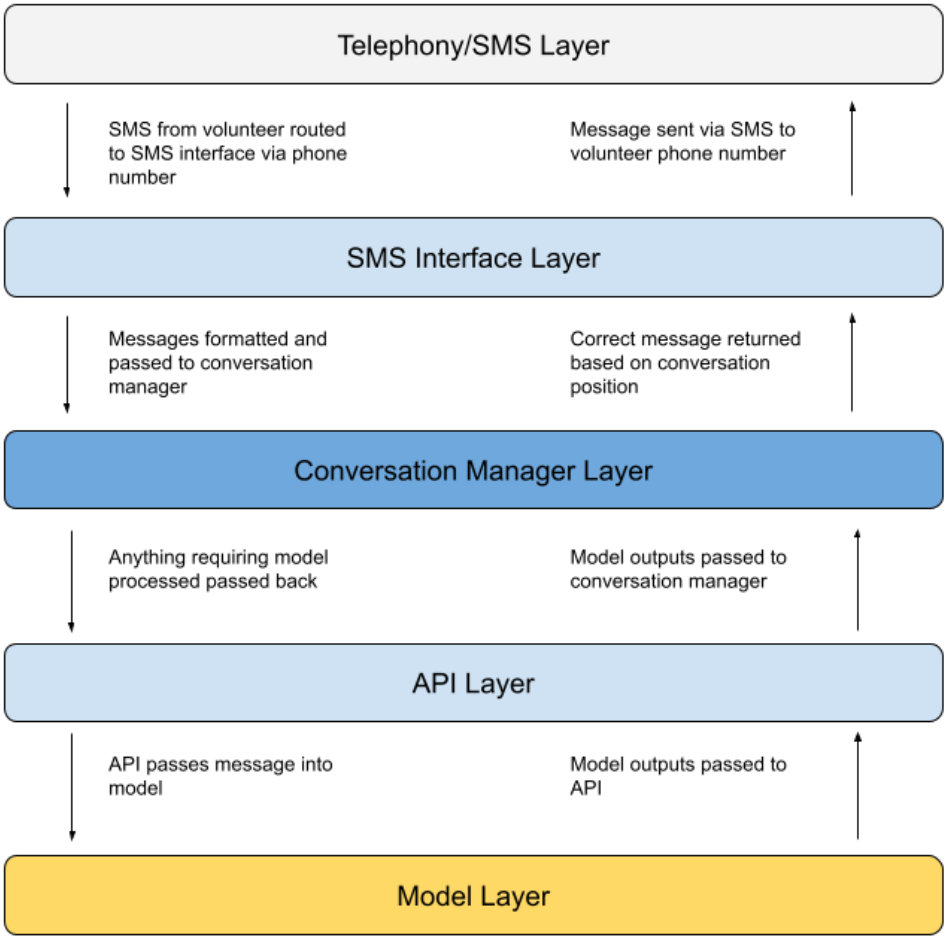

Figure 4. System layers showing data flows between layers.

System Component Details

The following sections provide detailed descriptions of how each component should function, including technology recommendations.

SMS System & Conversation Manager

These two components are tightly coupled. Due to the large number of telephony providers in most countries (including Cameroon), it is sensible to use a service that provides access to the telephony network programmatically and therefore all the telephony providers. Such a service will allow an SMS message to be sent programmatically (e.g. via an API) and then be correctly routed to the provided phone number, on the correct telephone network and in the correct format.

An example of such a provider is *Twilio*, who provide a *Conversations API* which can be used to send and receive SMS messages, (as well as telephone calls). Twilio handles the SMS layer and provides a *Conversation Manager* as part of their service, in order to programmatically manage inputs and outputs to each telephone number.

Conversation Manager

A conversation manager (CM) will be required to keep track of each user (defined by their telephone number), and where each user is in the conversational flow. The CM will handle all incoming and outgoing messages. When a message is received, it will route the message according to the point in the conversational flow, such as sending a request to the API for information from the model. The conversational flows are defined in the Conversational flows section.

As noted in the previous section, Twilio provides a Conversations API which provides the SMS interface as well as a conversation manager. The CM is configured to receive messages, and needs to be programmed to route messages and responses correctly. The Conversations API supports a number of popular languages. Further details:
Twilio Conversation API Pricing

Alternative architecture: SMS is the messaging service proposed for use in the SMS layer, as it is widely available (anywhere there is phone signal). However, there is a consideration of cost here, as SMS messages are often charged, especially in Cameroon. WhatsApp is an alternative that would cost less to the end user (i.e. volunteer), although internet access is required to send messages. Twilio provides a WhatsApp integration within their Conversations API. With this configured, Twilio routes incoming WhatsApp messages to and from the volunteer. In order to use WhatsApp, a licence to use WhatsApp Business API would be required, which incurs a monthly cost.

Model API specification

The API would be invoked by the Conversation Manager based on where a conversation is in the conversational flow. There is a single endpoint, one to input a rumour and receive a response from the model.

Exposure: The API will only be used by the Conversation Manager, so depending on exact implementation could be secured via IP

Endpoint	**Method**	**Input query/params**	**Expected response**	**Onward processing**
/model/get/input	GET	1. Rumour text	1. Label 2. Response text	Passes rumour text to model

Volunteer Response API specification

The API would be invoked by the Conversation Manager in the conversational flow when a user has provided a message in response to a question (such as ‘Have you used this in the field?’). These contextual responses are stored in the Volunteer Responses Database via this API.

Exposure: This API will be used by the Conversation Manager as well as the web server (when generating reports), so depending on exact implementation could be secured via IP

Endpoint	**Method**	**Input query/params**	**Expected response**	**Onward processing**
/resp/post/response	POST	1. Flow ID 2. Question ID 3. Text response	Confirm POST OK	Sends data to database
/resp/get/data/week	GET	None	All responses input to the database since [$TODAY-6 days]	Used to generate weekly reports

Volunteer Responses Database

There are many points in the conversation flow where requests for contextual information are sent to the volunteer, such as ‘Have you used this response in the field?’. The SMS responses to these questions will be stored in a database. The webserver

Field	**Type**	**Description**	**Example**

rowId	Int	Primary Key	*1*
uniqueId	String	Unique identifier of conversation Id	*aaaa-b1cd-bbl9*
flowId	Int	ID of conversation flow	*3*
questionId	Int	ID of which question within conversation flow	*5*
response	String	Text response received by SMS	*Oui j'entend beaucoup*

Web Server

Given the modular nature of the architecture we have described here, there are multiple ways the backend could be implemented (outside of Twilio's Conversations API). We have provided a proposed web server implementation in the diagram below. This uses one server, but for improved separation, the CCI model and associated API could be run on a separate machine.

Hosting APIs

In our proposed implementation we suggest the use of Express.js running on NodeJS to route all incoming API traffic. Based on the proposed endpoint addresses (see sections Model API specification and Response API specification), traffic can be routed to appropriate functions to query either the CCI model or the volunteer responses database.

Model API

API requests that require input and output from the model would be handled by a function that invokes the model and passes in the input from the API request body/parameters, and handles the output from the model. This output will be the label and official response of the rumour, if it is recognised, otherwise a message that it is not recognised.

Volunteer Responses API

API requests that are responses to contextual questions will be stored in the database, using the appropriate DB handler.

Generating Reports

This component will output the responses that have been received as part of the conversational flows. These responses provide contextual information to the CRC team about how responses are being used by the volunteers, as well as suggested labels for rumours that the model does not recognise.

To implement this report generation functionality, we envisage a simple solution could be employed:

A small system script (e.g. Node script, triggered by PM2) could be run at a nominated day each week (e.g. Sunday). This script would query the Volunteer Response Database to fetch all inputs that have been received within the last 7 days. This script could then aggregate the responses by flow and question ID, or similar, and then output these to a designated email address as an HTML formatted email.

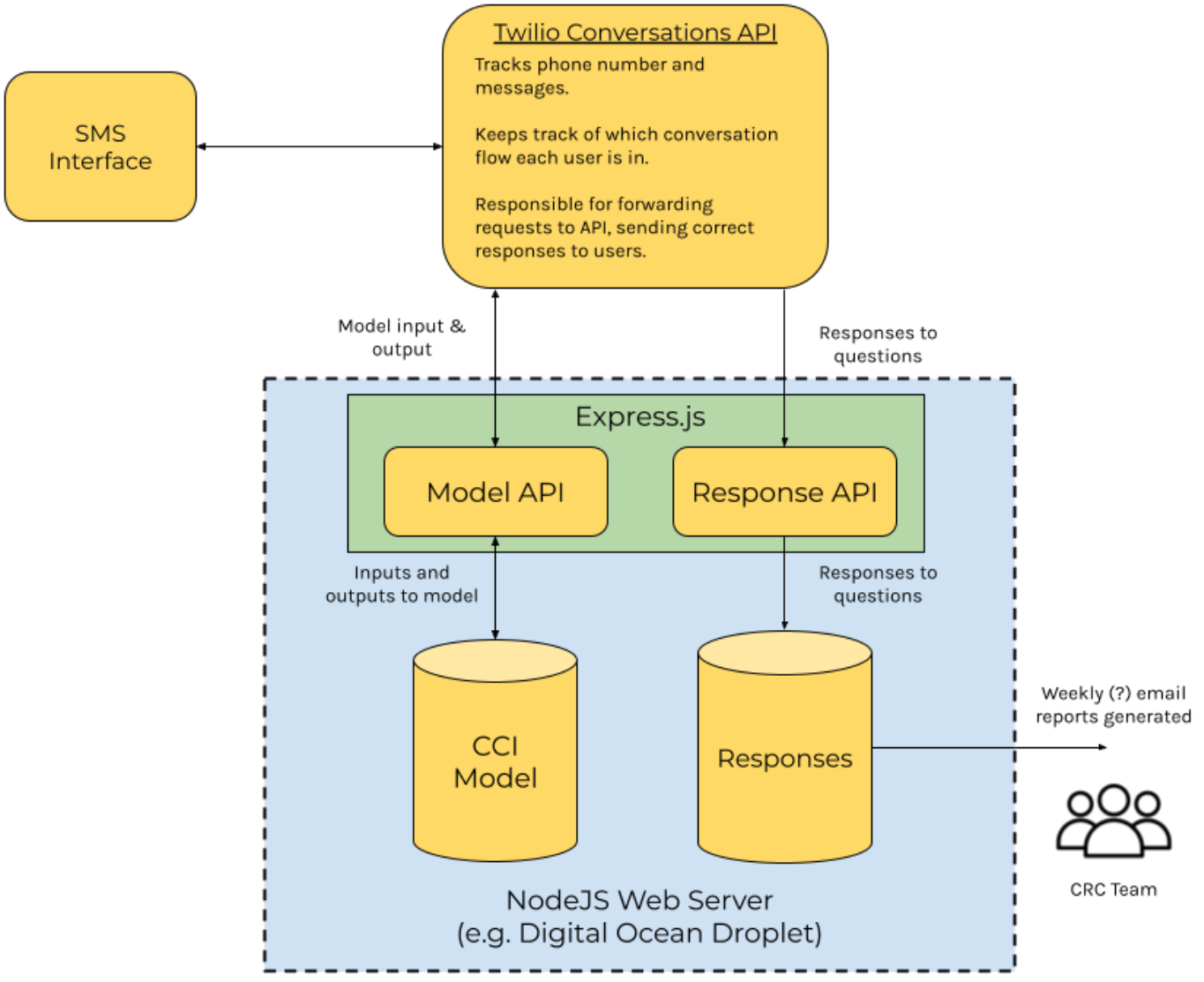

Figure 5. Proposed backend implementation

Security considerations

This tool provides a mechanism for volunteers to receive official RC responses to rumours via SMS. The nature of this data is that it's fit for public consumption (the volunteers will be disseminating this data freely when they receive it from the tool). As such there are minimal considerations for security. Two key issues are the unintentional capture of personal information, and unauthorised usage of the tool. Below, we detail how the service should be implemented to maintain integrity.

Data security

The data being processed in this tool is not 'personal information' (as considered by GDPR), however it is possible that volunteers could unintentionally submit personal information via SMS to the service (e.g. disclosing names of people when answering 'Have you used this response in the field?' contextual question).

As such it will be imperative to implement a flagging/moderation process where the CRC team can identify when personal information is being stored in the volunteer responses database, and it can be removed. It will also be important to educate volunteers on the use of the tool, that they should not submit personally identifiable information.

Unauthorised usage

Telephone numbers are accessible to anyone on the telephone network. Therefore it is important for a security solution to be employed to stop unauthorised usage. In principle there is no problem with members of the public using the service, as the data being used it for public dissemination anyway, however there are a number of factors that would make public use undesirable, such as:

1. Increased usage would add additional load and cost on infrastructure services
2. Input to contextual questions (e.g. "Have you used this response in the field?") would pollute data and significantly reduce the efficacy of this data capture mechanism.

Our recommended security solution would be to restrict usage of the R&R tool to a specific list of telephone numbers (those of the volunteers), and reject all other SMS messages. This can be configured through the SMS interface (Twilio). This stops all unauthorised usage, and only requires that volunteers provide their telephone numbers so they can be allowed access. It should be noted that this does have a couple of drawbacks:

1. If a volunteer needs to use a different phone at the last minute they will be unable to use the service
2. This creates an administrative burden where someone has to get the phone number and input it into the system

An alternative solution is to use a passcode. For example, before the volunteer is able to input a rumour they are required to input a passcode, that could be sent individually, or at the beginning/end of their SMS message containing the rumour. This would open the tool to all phones and reduce the administrative burden of authorising volunteer phone numbers as

proposed above. Our opinion is that this is a clunky mechanism, and allows the possibility of volunteers losing the passcode and being unable to use the service. It also lengthens the conversation flow, and also the cost of using the service (more messages required to be sent). A further security mechanism that could be implemented if the tool is publicly available, is restricting telephone numbers to Cameroon only.

Maintaining integrity

As the tool provides a canonical source of responses for the volunteers to use, it is important that the integrity of the service is maintained. It is important to temper these considerations with the reality that this system is not mission-critical, and therefore extremely strong security measures are not required to protect this service. The architecture we have proposed is inherently secure by design, using secure APIs to communicate between webserver and the Conversation Manager. Using a reputable conversation manager/sms interface is important to maintaining integrity (e.g. their API has an authentication layer, API keys, etc). We also keep as much processing within a single server as possible.

We have identified a number of key areas that should be considered when maintaining integrity of the service:

SMS numbers

Operating a service on the SMS network presents a risk of malicious services being set up using a similar number. For example, an imitation service could be configured with 1 digit altered in the phone number, which could easily be used to trick volunteers into using it. As such it will be important to be vigilant of phishing scams (a likely vector to get volunteers to use an imitation service), as well as remaining aware of new similar services that may be released to ensure volunteers do not accidentally use the wrong service.

Application security

All APIs should be secured in some way (e.g. an authentication layer), and ideally be restricted by IP to only the machines that are using them.

The keys for the SMS interface & Conversation Manager (e.g. Twilio) should be rotated every 3 months, and kept securely.

Projected Time & Costings

Assumptions:
All development time is full-time, based on developers working 9am - 5pm. Time estimates are conservative to factor in unforeseen technical challenges.
Financial cost assumed in GBP

#	Task / Item	Details	Time cost	Financial cost
1	Building and test Twilio Conversation Manager	Configuring the service, getting SMS service working correctly with automation.	6 weeks	Developer time x time cost
2	Building and test Model API	Specifying and handling input and output to the model through an API.	2 weeks	“ “
3	Configuring resources to host model & API	Setting up the server, making all services working interoperably, getting service stable.	2 weeks	“ “
4	Alpha Testing	Testing with small group of users (either external or internal to IFRC)	4 weeks	“ “
5	Iteration based on Alpha testing	Fixing bugs and making changes suggested from the alpha testing.	3 weeks	“ “
6	Beta Testing	Testing in Cameroon in the field (using different cellphone providers)	4 weeks	Developer time SMS message costs Potential phone/sim card purchases for test
7	Iteration based on Alpha testing	Fixing bugs and making changes suggested from the beta testing.	3 weeks	“ “
		Development Total	24 weeks	
	Recurring			

1	Twilio Conversation Service	Initial + maintenance cost for Twilio service	N/a	Requires negotiation w/ Twilio to find this out. Education & non-profit licences available.
2	SMS costings	Mobile phone credit provided to volunteers to use the service	Monthly/weekly providing top ups to volunteers. Estimate 1hr p/m	Dependent on usage, see notes
3	Technical support	Bug fixing, maintaining service availability. Sensible to budget for at least 2 months technical support when the tool is launched to fix issues that arise in the field. Depending on scale of deployment and use of the tool, a service-level agreement should be discussed for expected uptime and maintenance response times.	Variable	2 months 'on call' development time once tool is released Service-level agreement TBC