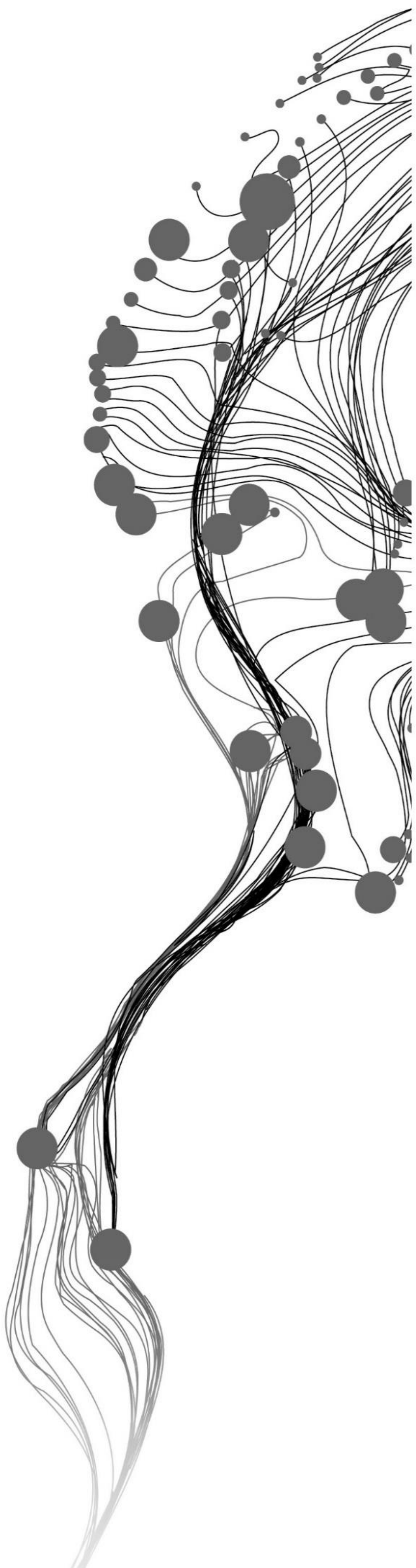
INFORMATION FLOW IMPROVEMENT FOR GEOCOLLABORATIVE SYSTEM BETWEEN IN-FIELD AND IN-OFFICE USER IN AREAS WITHOUT INTERNET ACCESS

DIO DINTA DAFRISTA
March, 2017

SUPERVISORS:
Dr. F.O. Ostermann
Dr. ir. R.A. de By



INFORMATION FLOW IMPROVEMENT FOR GEOCOLLABORATIVE SYSTEM BETWEEN IN-FIELD AND IN-OFFICE USER IN AREAS WITHOUT INTERNET ACCESS

DIO DINTA DAFRISTA
Enschede, The Netherlands, March, 2017

ABSTRACT

Technological advancement provides opportunities for a real-time geocollaborative system where in-field users and in-office users do tasks together over a map interface. The distributed user location in such system has a strong dependency on the network to exchange the data and information. Empirical studies and practices show that internet connection is an important factor in the information flow between users because most of them are implemented based on client-server architecture. However, we also need to accommodate data exchange in areas without internet access. For example, in an emergency situation where the data needs to be sent immediately, but there is no internet connection available. This thesis proposes an improvement for geocollaborative system in areas without internet access.

The improvement is developed based on the gap we found from previous studies and practices on geocollaborative system. First, we select data collection as task of geocollaborative system because it is the task that is most likely used in distributed synchronise geocollaborative system. We review previous works and practices for data collection to find the flows of the information. Then we review characteristic of geocollaborative system to find important component that needs to be considered. Finally, we find the gap from previous research and practices by considering the flows of information and the component of geocollaborative system in areas without internet access.

We propose a new mechanism called SFormBD to fill the gap by creating a new feature of geocollaborative system that allows in-office users to send a new task to in-field user by using SMS. In the end, we evaluate the proposed mechanism by developing a prototype that was created based on a scenario where the new feature is needed.

Keywords: mobile data collection, geocollaboration

ACKNOWLEDGEMENTS

Alhamdulillah,

First, I would like to deliver my gratitude to my supervisors and advisor for all the help, guidance and feedback throughout my research period

Friends, for all precious and beautiful memories. For keeping me sane during the hard times.

My family, for the endless prayer and support.

Finally, I am utmost appreciative of all the support given by LPDP (Endowment Fund of Education) through a scholarship that made this possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

API	-	Application Program Interface
GIS	-	Geographic Information System
GPS	-	Global Positioning System
GSM	-	Global System for Mobile Communication
HTTP	-	Hypertext Transfer Protocol
JSON	-	JavaScript Object Notation
ODK	-	Open Data Kit
RDBMS	-	Relational Database Management System
REST	-	Representational State Transfer
SFormBD	-	Form Base Data for SMS-based Data Collection
SMS	-	Short Message Service
UMB	-	USSD Menu Browser
USSD	-	Unstructured Supplementary Service Data
XML	-	Extensible Markup Language

1. INTRODUCTION

1.1. Background and Motivation

Technological advancements provide opportunities for the implementation of effective real-time collaboration of geospatial tasks. For example, real-time geocollaborative system where in-field users and in-office users do tasks together over a map interface (Cai, 2005; Heard, Thakur, Losego, & Galluppi, 2014; Wang, Qiao, Wu, Chang, & Shi, 2016). In-field users are usually distributed in locations of interest to do tasks using their mobile device, while in-office users allocate and monitor the tasks. Presently, most of the implementation are based on a client-server infrastructure that supports in-field and in-office communication over an internet connection. Wang et al. (2016) stated that one of three important factors for their collaborative system is the network coverage needed to transfer the data to the server.

However, a strong dependency on internet access can cause problems in the implementation because: (i) not all areas are covered by internet connection, especially in remote areas, and (ii) the quality and speed of the broadband are not evenly distributed, especially in rural regions. The Telecommunication Development Sector (ITU-D) stated that, in 2016, area coverage of mobile-broadband network is 84% of the global population, but only 67% of the rural population (Sanou, 2016). Furthermore, the penetration of mobile data technologies is poor in rural area (Dasgupta, Kamble, Ghosh, & Acharya, 2013). Hence, the areas which are the most accessible have coverage first, while the rising costs of access in the remaining areas slow down the extent of the coverage. This issue leads to the initiatives of major companies like Facebook and Google to start projects that connect rural and remote areas with internet, and eventually provide everyone in the world access to the internet (Facebook, 2015; X, 2013). However, these projects are still under development and cannot currently offer a solution for the problem at hand.

As a consequence of these limitations, it is difficult to make use of the full potential of collaborative systems, especially when the tasks need to be completed in a specific time. The following processes of real time geocollaborative tasks would be difficult in no internet areas: (i) access to data, information, and tasks from the server, (ii) transmission of data to the server, (iii) direct communication and interaction between in-office and in-field users, and (iv) visualization of the integrated and processed data to all involved users for further analysis. In these cases, the design of geocollaborative system systems based on client-server infrastructure requires adjustments in the means of communication to allow an effective information exchange between in-office and in-field users.

1.2. Research Identification

The main objective of this research is to improve the information flow between in-field and in-office users so that the collaboration process works even in areas without internet access. Specifically, the design should allow in-office users to create tasks that require being solved within a specific spatial and temporal context by in-field users. The objective is broken down into sub-objectives as listed below, each followed by research questions:

1. To identify geocollaborative workflows that involve tasks from in-field and in-office users, based on existing research and practices

- What are typical flows of information and tasks?
- How can we characterise collaborative settings?
- What are possible problems for each identified typical flow in areas without internet access?

2. To review existing technologies for communication between in-field and in-office users.
    - What communication protocols do mobile devices have and what are their potentials for information exchange in areas without internet access?
    - What are their characteristics and their limitations with regards to the identified collaborative workflows?

3. To improve the information flow between in-field and in-office users in areas without internet connection.
    - What are the options to improve information flows given technological and organisational constraints?
    - Which is an effective and efficient set-up for a reliable information flow in a geocollaborative setting involving in-field and in-office elements?

4. To evaluate the proposed improvement
    - How can we evaluate the proposed improvement?
    - What are the requirements to conduct the evaluation?
    - What is the result of the evaluation?

1.3. Innovation Aimed at

Current research about geocollaborative system is not addressing communication strategies between in-field and in-office users in the areas without internet access. The research proposes a new design that can improve the information exchange by accommodating in-field user in areas without internet access.

1.4. Research Methodology

To carry out the research, several steps are taken. Figure 1 shows the work phase of the research. First step is to do literature review and interviews with practitioners as describe in Chapter 2. The purpose is to evaluate existing technologies and to identify workflows in geocollaborative systems. The workflows need to be identified to assess how the information flows in the systems. In parallel, we review existing technologies that have been used in geocollaborative systems to assess how certain information flows in the workflow have some problem in areas without internet access. Based on the limitation we found, we propose a new design on how to improve the information flow in that scenario(s) in chapter 3.

Later, after the proposed improvement design is completed, a prototype is developed to evaluate the design. The prototype is developed based on simple scenario that need the improvement. Elicit user requirements and data collection is done to support the prototype. Chapter 4 discusses the design, implementation, and the testing result of the prototype.

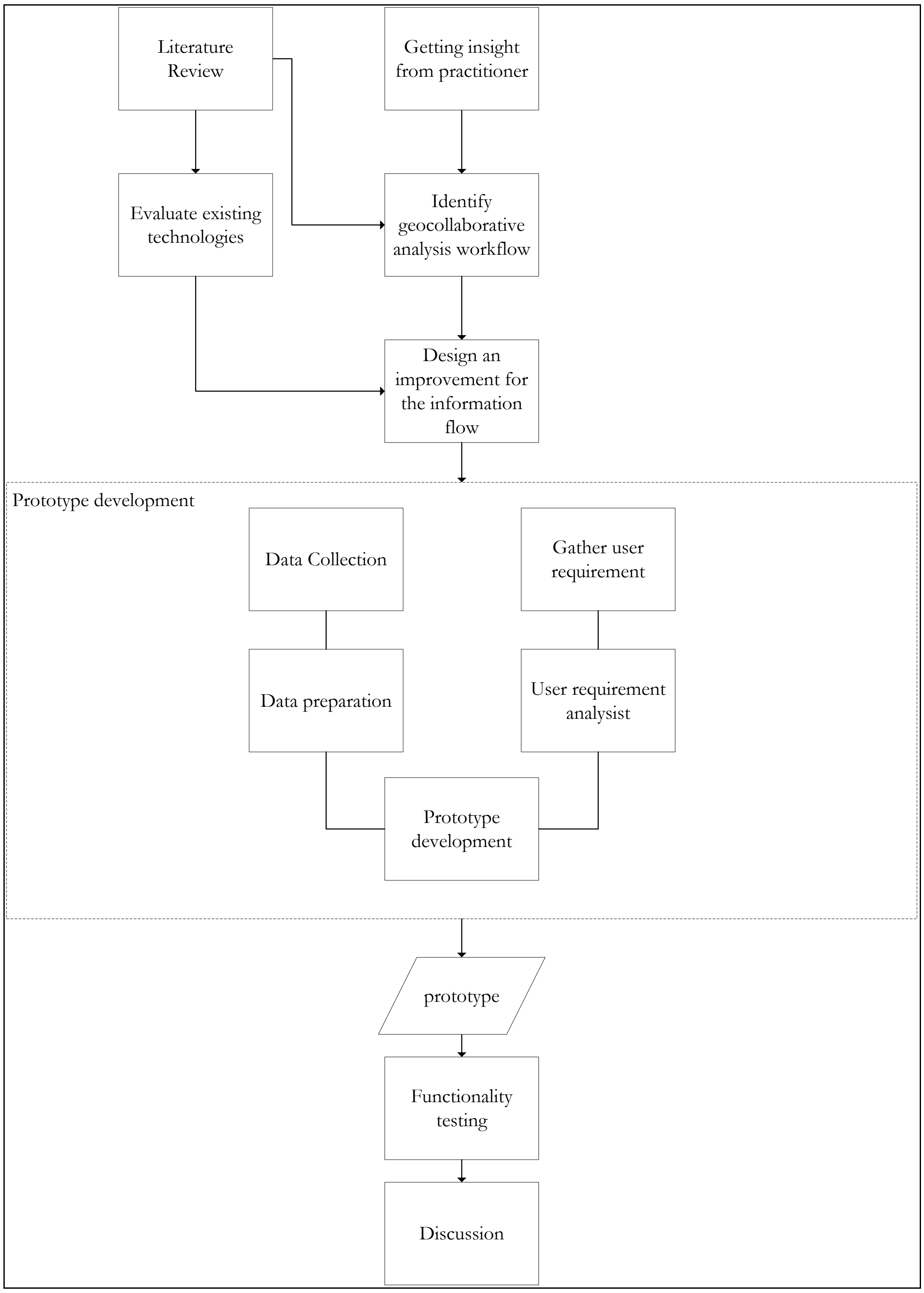

Literature Review
Getting insight from practitioner
Evaluate existing technologies
Identify geocollaborative analysis workflow
Design an improvement for the information flow
Prototype development
Data Collection
Gather user requirement
Data preparation
User requirement analysist
Prototype development
prototype
Functionality testing
Discussion

Figure 1. The flow of the methodology

2. INFORMATION FLOWS IN GEOCOLLABORATIVE SYSTEMS INVOLVING IN-FIELD AND IN-OFFICE USERS

In this section, geocollaborative system is described based on empirical studies. The purpose is to identify geocollaborative tasks workflows that involve in-field and in-office users based on existing research and practices. Maceachren & Brewer (2004) described geocollaboration as an activity that use visually-enabled geospatial technologies to support group work. A group work approach is ideal in cases where the problems are not well-defined, or it is a multi-disciplinary problem or several stakeholders are involved (Cai, 2005). Geocollaborative system is capable of combining the strengths of machine and human analytical thinking. This is because machines have a consistent performance, even in time-consuming processes, while humans possess capabilities of perception and cognition tasks (Thomas & Cook, 2005). Thus, even though most of the geoinformation tasks can be done automatically by the machine, such as in data management, data mining, and data visualisation, geocollaboration between users with different capabilities give more advantages for the use of the system.

Generally, collaborative activities are classified based on two dimensions: (i) space or user location (co-located and distributed) and (ii) time synchronization (same time or synchronous and different time or asynchronous) (Heard et al., 2014; Thomas & Cook, 2005; Wu, Convertino, Ganoe, Carroll, & Zhang, 2013). This research focuses on distributed-synchronous geocollaboration where users in the field work together at the same time with users in the office.

2.1. Workflows in Geocollaboration

To understand how the information flows in such geocollaboration, first, we need to know the workflow in the geocollaborative system. Distributed-synchronous geocollaboration involving in-field and in-office users can be implemented in many ways. The differences can be caused by the diversity of the tasks, the scopes (how big is the data? how many users involve in the system? what is the extent of area for field work?), urgency (what is the acceptable completion time? what is the acceptable waiting time for each part of the works?) and network availability (remote areas, 2G coverage, 3G coverage, and 4G coverage), which then will affect the preference of devices, how the information flows, communication channels between the users, and the system architectures. In the end, the chosen solutions for building the system must allow in-office users to do tasks together with in-field users within a specific spatial and temporal context. That is in-field users allowed to do the tasks in selected extent of areas and send the results within allowed range of time to other users so that the other users allowed to contribute in the works.

An example of the geocollaborative system was implemented by Wang et al., (2016) for monitoring fall webworm in northern China. The main workflow of their application is shown in Figure 2. Crowdsourced data of the occurrence of fall webworm are collected from local reporters (in-field users). At the same time, managers (in-office users) monitor the data mapping collection using a visual interface. If they see that data is not enough to represent the occurrence, they dispatch a task to volunteer rangers to collect more detailed information on the particular site.

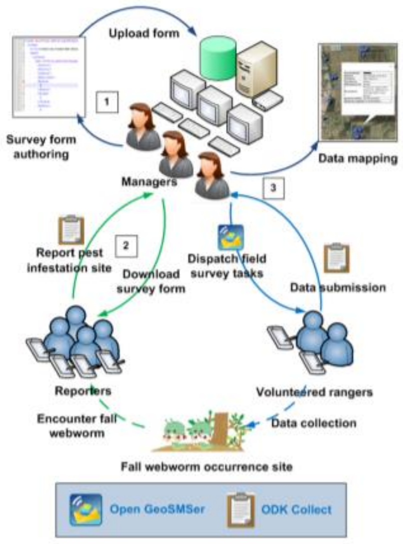


Figure 2. Main workflow of geocollaborative analysis for monitoring of fall webworm (Wang et al., 2016)

Another example of a geocollaborative system is a web-based application for distributed crisis management developed by Cai (2005). In crisis management, an integrated system for sharing information and expertise is important for rapid assessment and to support decision making. The workflow of the implementation is shown in Figure 3. The geocollaboration involves large screen display in Emergency Operation Centres (EOCs) for the decision maker in the office and portable devices for first responders in the field such as police, fire, and medical services.

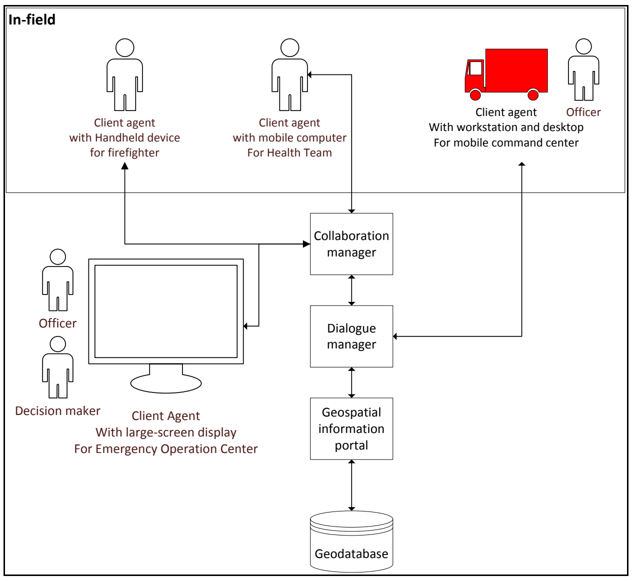


Figure 3. Geocollaborative crisis management workflow adapted from Cai (2005)

2.2. Information Flows in Geocollaborative System

To understand how the information flows between users in the workflows, we need first to understand: (i) what kind of tasks that users do with the system, and (ii) what kind of information to be shared with others. Tasks in geoinformation technologies are related to data collection, data processing and visual presentation (Smirnov & Ponomarev, 2015). While all the tasks in the distributed-synchronous geocollaborative system offer wide scope for research, this research focusses on data collection because it is a common task that is very likely to be done in the field and involves information flow and communication strategies to transfer the data. The geocollaboration examples that are described in Section 2.1 also prove that in-field users collect data from the field (data collection) and send it to the server. Next, depending on the need and interest, the data can be both processed and visualised by the in-office users.

2.2.1. Data Collection Tasks in Geocollaboration

Data collection in the field used to be paper-based form and then, as the technologies evolved, the forms changed into a digital format that could be filled in mobile devices. This improvement not only saves the environment by reducing the use of paper but also save the effort time to post-process the data. For example, (Sa et al., 2016) utilise mobile data collection to improve the efficiency of health initiative and deliver a positive outcome in both data quality and timely delivery. While custom development of mobile data collection needs basic programming and database knowledge, there are also many user-configurable data collection platforms that are extensible such as Open Data Kit (ODK), GeoODK, and Collector for ArcGIS (Esri, 2016; GeoODK, 2014; Kipf et al., 2016; Tayal, 2015).

The data collection platforms provide not only map-based data collection capability but also data management and dissemination that make them potential capabilities to be used as a geocollaborative system in data collection. Figure 4 shows the main workflow in data collection platforms such as ODK, GeoODK, and Collector for ArcGIS. It starts with the creation of a form. Then, the form is shared to the in-field user devices so that the in-field user can do data acquisition by filling up the form. After the data acquisition is completed, the in-field user needs to upload the filled form to the server so that those data can be managed. Data management is designed as storages, including the form and the result of the data collection. From this data, for example, we can extract the data into a useful format or visualise it, so then the other user who interested on that data can work with them.

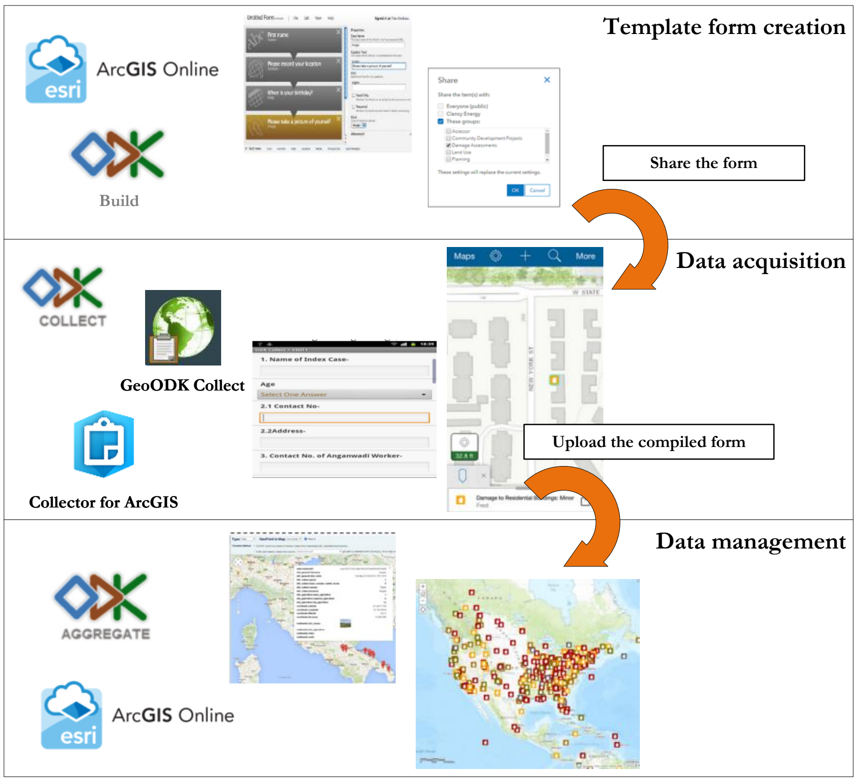
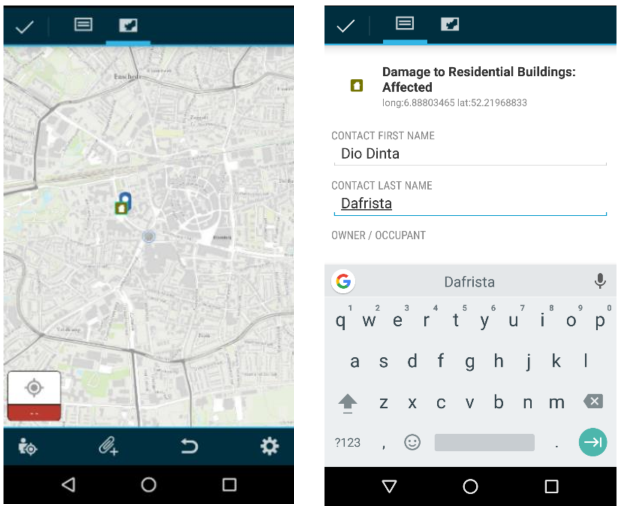


Figure 4. Data collection platform review (Esri, 2017; Lee, 2015; Signore, 2016; Tayal, 2015)

2.2.2. The Information Flows

The workflow of data collection as seen in Figure 4 shows that the process of sharing the form and uploading the compiled form are processes that involve data exchange. Data exchange is also needed after the form is filled and needs to be sent. Data dissemination in mobile data collection platform is responsible for handling the data transfer. It addresses the task of encoding the form and filled form into a suitable format. The form needs to be encoded so that it can be deployed on the mobile devices, decoded into a form that can be filled, and then the filled form can be stored and send back to the server (Kipf et al., 2016). The description of the strategies of disseminating the data will be discussed in Section 2.3 while the example of the form can be seen in Figure 5. The form shows a site-survey questionnaire that is used to do damage assessment survey in the field.


Damage to Residential Buildings: Affected
long:6.88803465 lat:52.21968833

CONTACT FIRST NAME
Dio Dinta

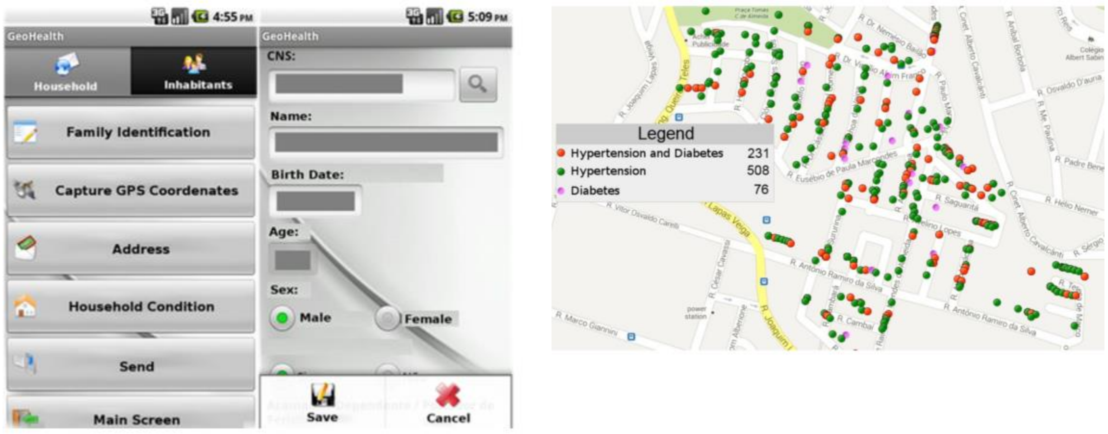
CONTACT LAST NAME
Dafrista

OWNER / OCCUPANT

Figure 5. The example of map-based data collection captured from Collector for ArcGIS

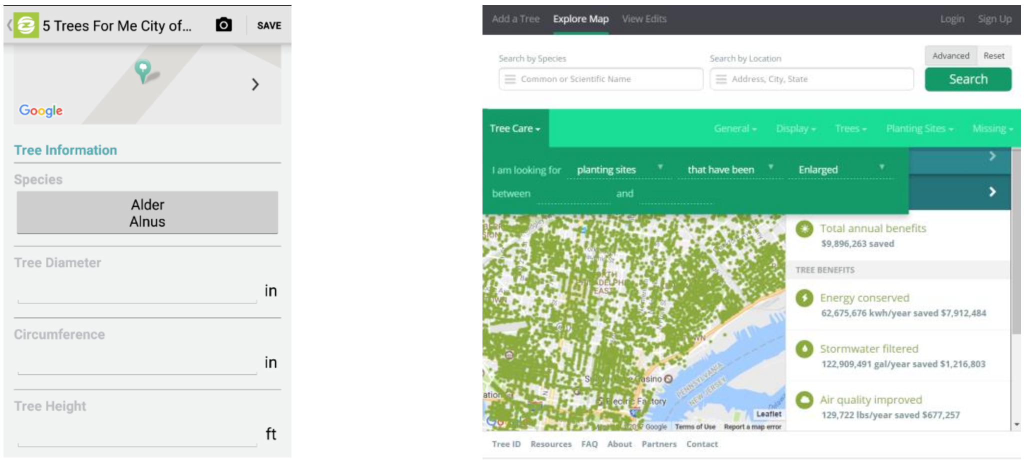
In location-based mobile data collection example, as seen in Figure 5, point location is used to indicate where the data is acquired. ODK, GeoODK, and Collector for ArcGIS have the capability to use built-in GPS (Global Positioning System) to point out their location and send it along with the filled form. However, there is concern about the positioning quality in the built-in GPS of the mobile device. Brovelli, Minghini, & Zamboni (2016), GeoODK collect, and Collector for ArcGIS utilise map with knows accuracy as a reference to help user point out their location in case they do not have satisfactory access to the GPS. The other solution is by using external GPS which then we can input the GPS coordinate directly in the form.

Even though the use of geospatial information in the data collection is limited to the use of location to indicate where the data is acquired, it can be extended depend on the use case. For example, Freire & Painho (2014) implement the data collection by sending the field data into spatial database server, then process them into a Web GIS that has the ability to create thematic maps and statistical reports and to do query analysis, spatial data editing, and visualization. Another example developed by Sa et al. (2016) that plot comorbidity map based the data collected by mobile application as seen in Figure 6. Moreover, OpenTreeMap publishes a new tool to collect tree data that integrate with web-based map database that has capabilities to calculate economic benefits and environmental impacts of trees as seen in Figure 7.



a b

Figure 6. Data acquisition (a) and plotted map as data visualisation (b) (Sa et al., 2016)



a b

Figure 7. Data acquisition (a) and the web-based application of OpenTreeMap (b) (Azavea, 2017)

2.3. Communication Strategies to Transfer Data

Considering that collaborative system needs process coordination and well-managed communication strategies between individuals, it is necessary to have effective and comprehensive communication tools. It enhances performance as it is responsible for transferring, receiving, and integrating the data (Kapucu & Garayev, 2011). While most of them implement client-server architecture to send and receive the data via internet connection, some of them also implement other communication strategies to avoid the unavailability or instability of internet connection. For example, voice call and instant messaging (Heard, Thakur, Losego, & Galluppi, 2014), Short Message Service (SMS), and Unstructured Supplementary Service Data (USSD). This section discusses the previous implementations of data collection using various communication channels and how they handle the information sharing between individuals.

2.3.1. Internet

Internet technologies provide capabilities for real-time interaction between individuals (or components) present in the system. Therefore, this technology is widely used in mobile data collection application as a communication strategy. Although they rely on Internet connection to transmit the data, well-designed internet based data collectors, such as ODK, GeoODK, Collector for ArcGIS, support offline setting which allows them to work even if Internet connection is not available. Offline support helps in-field users to retrieve data from the servers before going to the field and keep it locally in their devices for reference during field work. Offline support in mobile devices that can be implemented in the system are as following:

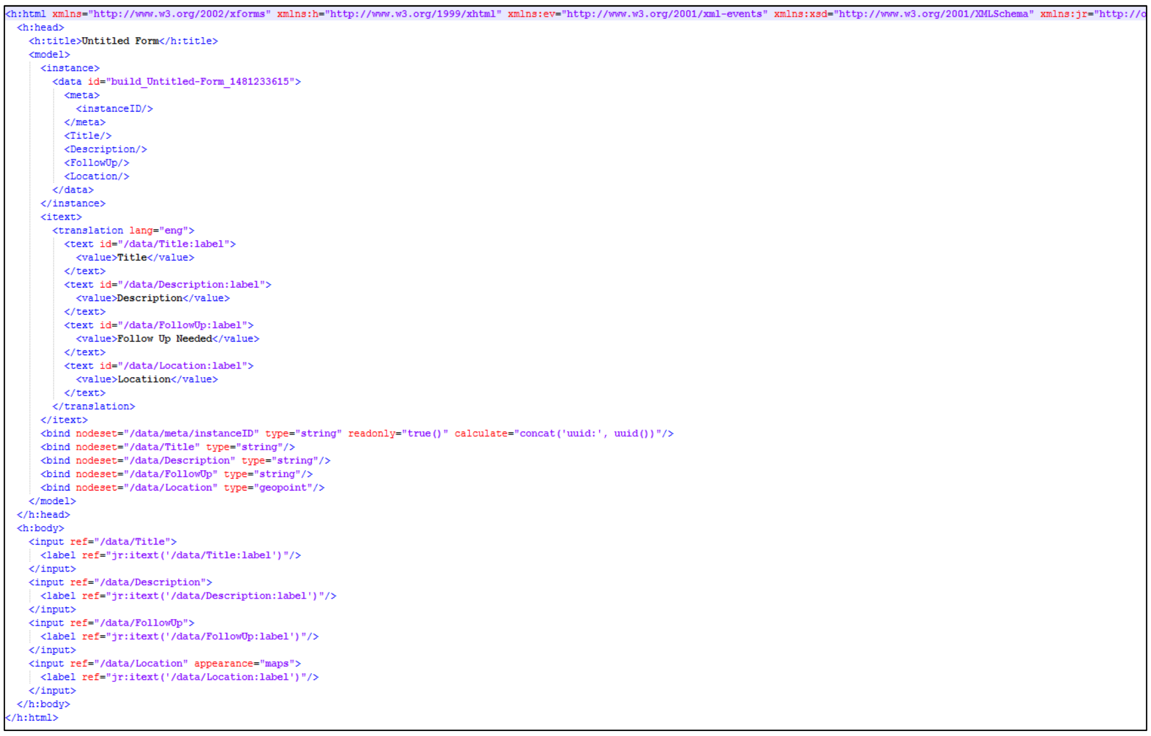
- Local caching

  To support offline mode in mobile application, internal storage service needs to be provided to store data locally. With this strategy, the application can work offline using local data as it would online by using data from the server.
- Local queuing

  To support data synchronisation and data consistency between client and server despite disconnected network, the data changes on the client side (insert, update, and delete) must be in the queue in a persistence way they are executed on server side.
- Data sync

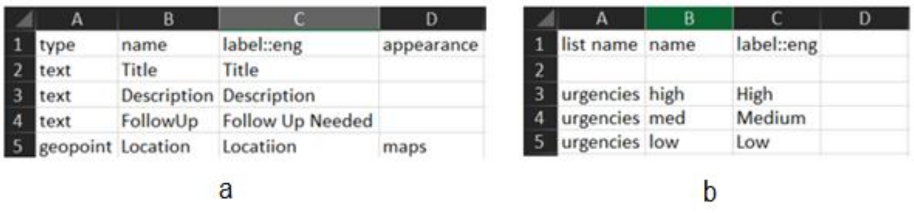
  Offline support should allow the application to keep the new data written by the application when the connection is not available and resend it to the server when the application comes back online.

By using the internet, there are several data formats that have been used to handle the data transfer. The primary purpose is to build a form, send it over internet connection, and save it in the mobile devices. The examples of the data formats are XML (Extensible Markup Language), XLSForm, and JSON formats. XML is a simple and flexible text format from World Wide Web Consortium (W3C) that has important role in exchanging wide variety of data on the web (Quin, 2016). XLSForm is a standard format that has been used to help create forms in Excel. For basic formats, each excel workbook has survey and choices worksheet which are used to specify the form contents; overall structure, constraints, questions, and choices (XLSForm.org, 2017). The example of mobile data collection application that uses XML and XLSForm formats is Open Data Kit (ODK), one of the most common open mobile data collection solution. The ODKs forms are build based on logic and data schema and can be exported into XML or XLSForm format. Figure 9 shows the example of a form from ODK that is exported into XML format, while in Figure 8, it is exported into XLSForm. The exported data then can be downloaded from mobile devices or put directly into mobile devices storage.



```
<h:html xmlns="http://www.w3.org/2002/xforms" xmlns:h="http://www.w3.org/1999/xhtml" xmlns:ev="http://www.w3.org/2001/xml-events" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:jr="http://o
  <h:head>
    <h:title>Untitled Form</h:title>
    <model>
      <instance>
        <data id="build_Untitled-Form_1481233615">
          <meta>
            <instanceID/>
          </meta>
          <Title/>
          <Description/>
          <FollowUp/>
          <Location/>
        </data>
      </instance>
      <itext>
        <translation lang="eng">
          <text id="/data/Title:label">
            <value>Title</value>
          </text>
          <text id="/data/Description:label">
            <value>Description</value>
          </text>
          <text id="/data/FollowUp:label">
            <value>Follow Up Needed</value>
          </text>
          <text id="/data/Location:label">
            <value>Locatiion</value>
          </text>
        </translation>
      </itext>
      <bind nodeset="/data/meta/instanceID" type="string" readonly="true()" calculate="concat('uuid:', uuid())"/>
      <bind nodeset="/data/Title" type="string"/>
      <bind nodeset="/data/Description" type="string"/>
      <bind nodeset="/data/FollowUp" type="string"/>
      <bind nodeset="/data/Location" type="geopoint"/>
    </model>
  </h:head>
  <h:body>
    <input ref="/data/Title">
      <label ref="jr:itext('/data/Title:label')"/>
    </input>
    <input ref="/data/Description">
      <label ref="jr:itext('/data/Description:label')"/>
    </input>
    <input ref="/data/FollowUp">
      <label ref="jr:itext('/data/FollowUp:label')"/>
    </input>
    <input ref="/data/Location" appearance="maps">
      <label ref="jr:itext('/data/Location:label')"/>
    </input>
  </h:body>
</h:html>
```

Figure 8. Form in XML format exported from ODK



a

	A	B	C	D
1	type	name	label::eng	appearance
2	text	Title	Title	
3	text	Description	Description	
4	text	FollowUp	Follow Up Needed	
5	geopoint	Location	Locatiion	maps

b

	A	B	C	D
1	list name	name	label::eng	
2				
3	urgencies	high	High	
4	urgencies	med	Medium	
5	urgencies	low	Low	

Figure 9. The example of survey worksheet (left) and choices worksheet (right) from XLSForm

On the other hand, JSON (JavaScript Object Notation) is a lightweight ECMA (European Computer Manufacturers Association) International standard text format that has the capabilities to facilitate data interchange between many programming languages (Bray, 2014). There are several implementations of structuring form in JSON format. For example, there is a JSON Form JavaScript library which is defining and structuring the data model using JSON Schema (Joshfire, 2014). The other example is JSON Schema that is used in Angular Schema Form, a schema form that is using AngularJS directives (TEXTALK, 2016). JSON Schema is using an Angular JavaScript that can be used to develop an application on any platform, e.g. Web-based and mobile-based application. Table 1 shows the simple example of using JSON format for structuring form.

Table 1. JSON format to handle form

Implementation	JSON Format	Form Result

JSON Form	
	```
	{
	   name: {
	   type: 'string',
	   title: 'Name',
	   required: true
	   },
	   age: {
	   type: 'number',
	   title: 'Age'
	   }
	}
	```

Angular Schema Form	
	```
	{
	  "type": "object",
	  "title": "Comment",
	  "properties": {
	    "name": {
	      "title": "Name",
	      "type": "string"
	    },
	    "age": {
	      "title": "Age",
	      "type": "string"
	    }
	  },
	  "required": [
	    "name"
	  ]
	}
	```

2.3.2. SMS

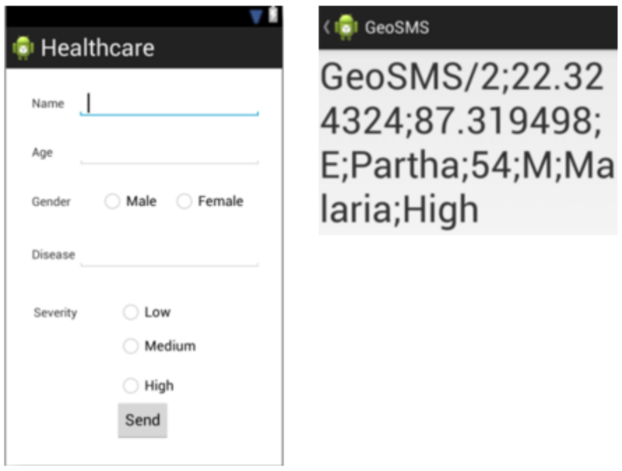
SMS (Short Message Service) is a basic mobile communication service between mobile devices service that is formed in a simple text format. In data collection tasks, it is used to transmit collected data in the field for several reasons; rural regions that do not covered by internet access (Dasgupta et al., 2013), and Emergency responses where the internet service becomes unavailable (Morrow, Mock, Papendieck, & Kocmich, 2011; Puspitasari, 2013). Even though SMS is one of the oldest methods of mobile communication and only accommodates 160 Latin characters in one text message, it requires low-bandwidth to transmit the data and allows messaging with very low latency (Triggs, 2013). Thus, SMS is suitable for communication in limited internet access areas. Additionally, it has SMSC (SMS Centre) that controls the distribution of messages including receiving, storing, and forwarding the messages. To ensure that the information sent successfully to the recipient, SMSC has a store-and-forward method that keeps trying to redeliver the message if the message is failed to be delivered to the recipient.

Most of the implementations rely on format-specific SMS's and a procedure to parse the data. Consequently, the implementations are only used in specific use case by implementing: (i) predefined SMS format for collecting the data, or (ii) predefined form to collect the data and automatically format the SMS based on the filled form so then it can be sent via SMS. Figure 10 shows the example of a predefined SMS format that needs to be typed and sent by in-field user. It is simple, can be used by almost all kind of mobile phone, and can be implemented relatively fast. However, it lacks the flexibility to integrate the data collection into another functionality, and it has big possibility to get error format because it fully relies on in-field user capabilities on typing in the right format.

```
(shelter_code)#(m_old)#(f_old)#(m_ad)#(f_ad)#(m_ten)#(f_ten)#(m_ch)#(f_ch)#
(toddlers)#(infant)#(pregnant)#(breastfeed)#(difable)#(households)
```

Figure 10. Predefined SMS format to simplify identifying and converting SMS into collected data (Puspitasari, 2013)

On the other hand, the use of a user interface as a predefined form as seen in figure 11 can reduce the chances of error in formatting the data, and by using such application, it is easier to integrate with another functionality in the system.



GeoSMS/2;22.324324;87.319498;E;Partha;54;M;Malaria;High

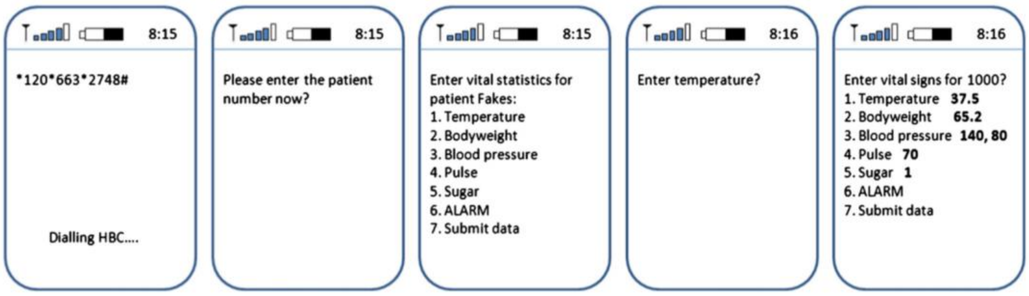
Figure 11. Predefined form (left) and its message format during transmission (right) (Dasgupta et al., 2013)

2.3.3. USSD

USSD best described as an interactive or bidirectional SMS that allows data transmission via Global System for Mobile communication (GSM) network (Wouters, Barjis, Maponya, Martiz, & Mashiri, 2009). It is generally used as a communication channel between mobile provider and mobile user. For example, mobile user can check credit balance and subscribe internet access by using this service. It works on almost every mobile device and easy to use, especially when it is in USSD Menu Browser (UMB) format that allows two ways communication between mobile provider and mobile user. UMB has been used as mobile marketing technologies that can be used to: (i) collect customer data, (ii) set up quizzes, (ii) provide info service, and (ii) conduct surveys.

In a geocollaborative system, Ochoa, Talavera, & Paciello (2015) utilised USSD messages as a real-time identification epidemiological risk areas to avoid the possibility of sending a malformed message by using predefined menu option in USSD module. The data was then be processed, analysed, and visualised in a map viewer. The USSD application uses RESTful web service that communicates with HTTP and XML protocols.

Another example is implemented by Barjis, Kolfschoten, & Maritz (2013) that proposed decision support system for rural healthcare delivery as seen in Figure 12. Care workers as mobile user need to dial a certain number that will establish a session of communication that can provide predefined questions that need to be answered based on the patient visit result.



8:15

*120*663*2748#

Dialling HBC....

8:15

Please enter the patient number now?

8:15

Enter vital statistics for patient Fakes:
1. Temperature
2. Bodyweight
3. Blood pressure
4. Pulse
5. Sugar
6. ALARM
7. Submit data

8:16

Enter temperature?

8:16

Enter vital signs for 1000?
1. Temperature **37.5**
2. Bodyweight **65.2**
3. Blood pressure **140, 80**
4. Pulse **70**
5. Sugar **1**
6. ALARM
7. Submit data

Figure 12. USSD Interface on cell phone as a data collection tools (Barjis et al., 2013)

USSD and SMS are two communication channel that mostly use to communicate in GSM network without internet connection. Compared to SMS, USSD has an interactive interface and can accommodate more characters. Table 2 shows how USSD and SMS are different from each other. A big difference between SMS and USSD is in the approach of communication. SMS has store and forward approach which makes sure that there is no information lost in transferring the data. It is because SMS has control centre (SMSC) that keeps resending the data until it is received by the intended receiver. Even when the message is not delivered successfully, the control centre will have the report that, in the end, the message could not be delivered. On the other hand, USSD has a session oriented approach that establishes a real-time connection during USSD session. It is responsive and able to deliver message relatively faster than SMS. However, the open connection is disabled after 3 minutes (Barjis et al., 2013).

Table 2. Differences between USSD and SMS capabilities (Suddul et al., 2011)

Feature	USSD	SMS
Store and forward approach	X	√
Session-oriented approach	√	X
Maximum message length	182 character	160 character
Guaranteed message delivery	X	√
Interaction between mobile users	X	√

2.4. Summary

In this chapter, we describe previous researches that discuss distributed synchronous geocollaborative system that involves in-field and in-office users. First, we described how they implemented it in various case studies. Based on the description and the workflow of the previous studies, we consider the in-field tasks as in-field data collection because it is the task that involves exchange data between in-field and in-office users and it is the most likely task that in-field user does in the office.

Then, we discussed in detail about mobile data collection. It focussed on how the data is transferred in various communication channel: (i) internet, (ii) SMS and (iii) USSD. From the description, we conclude that SMS and USSD are communication channel that has been used to exchange data in areas without internet access.

3. DESIGNING AN IMPROVEMENT FOR INFORMATION FLOW IN AREAS WITHOUT INTERNET ACCESS

In chapter 2, we described empirical research on the workflow of distributed-synchronous geocollaborative system and how the information flow has been supported in any possible communication channel as part of data and information sharing in geocollaborative systems. In this chapter, we focus on reviewing characteristic and limitation from previous research and practices of in-field data collection in geocollaborative system in regards to its flow of information in the areas without internet access. Then, based on the limitation, we propose an improvement to cover its limitation.

3.1. Reviewing Geocollaborative Workflows in Data Collection Tasks

Based on previous research and practices, there are various ways to implement a geocollaborative system for in-field data collection. However, to focus on the information flow, first, we need to describe the typical workflow of synchronous geocollaboration so then we can understand the links of the data. In parallel, we discuss the components that need to be considered when implementing distributed-synchronous data collection in geocollaborative system. Thus, to improve the information flow, we can assess how certain information links face problems in certain conditions.

3.1.1. Typical Flow

Based on existing mobile data collection platform that we describe in Section 2.2.1, there are three main tasks that needs to be done to complete the process of mobile data collection: (i) creating the form, (ii) data acquisition from the field and send it back to the server, and (iii) data management and visualisation. However, considering the need of geocollaboration between in-field and in-field user, the process need to be modified because (i) there is a possibility for the in-field user to start the work, and (ii) there is a need to have flexible communication for following up the tasks between in-field and in-office user (Wang et al., 2016). Figure 13 shows the typical flow that is developed based on the need of geocollaborative system in mobile data collection.

In diagram, we can see that: (i) Both in-field and in-office users can start the works, (ii) Data and information sharing among individual must be allowed during the works, and (iii) Data storing and data retrieval must be allowed, in this case, even in areas without internet access. Thus, the needed information links that will be discussed in this research become: (i) In-office user define tasks to in-field user, (ii) In-field user send tasks completion to the in-office user, and (iii) In-office user send follow-up tasks to in-field user if needed.

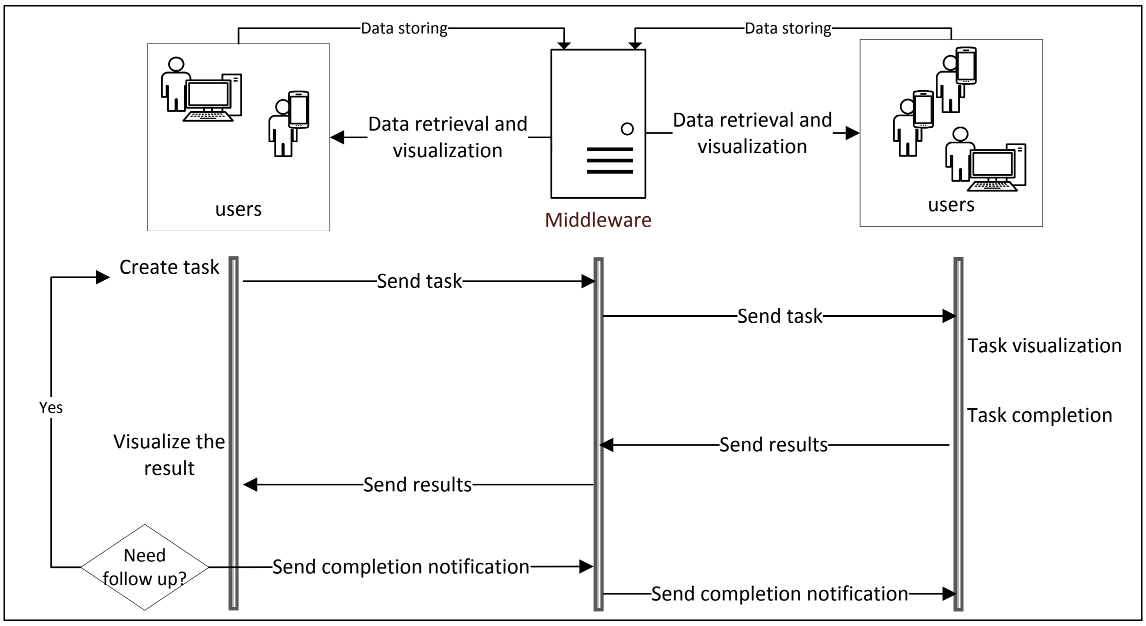


Figure 13. The typical flow of the collaboration between in-field and in-office users.

3.1.2. Previous Research Limitations

In addition to the typical flow, we need to describe the other components of geocollaborative system for mobile data collection that are implemented in the system. Three critical components are the use of map interface, mobile data management, and interactive and user-friendly user interface. As described by its definition, map interface is an important component for geocollaborative system. Mobile data management is a component of mobile data collection that enables access for in-field user to the data that they need in their mobile application. Lastly, interactive and user-friendly user interface makes the application easy to use. Table 3 shows the necessary component and whether the previous research on data collection in areas without internet access make them available in their system.

Table 3. Previous research on data collection in no internet areas and its coverage on geocollaborative system

No	Component	Predefined form	SMS Format	USSD
Information flow component				
1	In-office user define tasks to in-field user	√	√	√
2	In-field user send tasks completion to in-office user	√	√	√
3	In-office user send follow-up tasks to in-field user	X	X	√
Other component				
4	Map interface	X	X	X
5	Mobile data management	√	X	X
6	Interactive and user-friendly user interface	√	X	√

In Table 3, we can see that there are several items that can be improved in regards to the geocollaborative system. As seen in previous research, the advantages of using USSD (Barjis et al., 2013; Ochoa et al., 2015; Wouters et al., 2009) in stable mobile connection without internet is that, by default, it has interactive interface and in-office has the flexibility to send new tasks to in-field users, which means a good communication strategies for distributed synchronous geocollaboration for data collection. Previous researches do not discuss the integration between the use of USSD as data collector inside of a mobile application, which makes it a potential improvement. However, it is not a common service that mobile telecommunication provider gives directly to their mobile users. To explore and take advantage of this communication channel, we need to cooperate with the provider. Because of its closed infrastructures, it will be difficult to implement and evaluate a prototype that runs in the production.

On the other hand, by using a predefined form that is installed in the mobile device application (Dasgupta et al., 2013), the data acquired and filled in the mobile application can be sent to the server using SMS and automatically managed by using specific mechanism. In this way, it is practically possible to integrate the map interface and data management in the system. Compared to a strategy where the user is manually typing the acquired data in specific SMS format (Puspitasari, 2013), this strategy delivers more interactive and user-friendly application for the user to fill the data they acquire in the field. By doing so, it also prevents a possibility for in-field user to send the wrong format. However, by using predefined form, we potentially eliminate the geocollaborative capabilities because the communication is one way, which is limited from in-field user to in-office user.

By using SMS as the communication channel, we can see the gap of information flow in the table by noticing that in previous research and practices, the geocollaborative system lose its flexibility to communicate. In this way, the system needs the flexibility in transmitting data so that the communication is not one way. In this works, we improve distributed synchronous geocollaborative system by creating a new feature that allows in-office users to follow up the tasks by sending a new form to in-field user. By doing so, we make possible all the information flow in data collection for geocollaborative system in areas without internet access. In an urgent situation, it will give a solution to the in-office side that needs to send a new task to in-field users immediately, but the in-field users are not connected to the internet.

3.2. SFormBD: Proposed Mechanism for Compacting a Form

This section discusses a solution to improve the information flow of distributed-synchronous geocollaborative system by allowing the system to transfer forms through SMS as discussed in the Section 3.1.2. The flexibility for transmitting data in areas without internet access can be implemented by structuring the form into a compact format. The structure needs to be: (i) save as much space as possible because the cost of the SMS is per 160 characters, and (ii) systematically possible to be processed automatically. This research proposes a mechanism to limit space needed in the process of sending and receiving the tasks when using SMS called Form Base Data for SMS-based data collection (SFormBD). By adapting the previous formats (JSON, XML, and XLSForm) to construct a form, we learn how to minimise the used components and the characters so it can fit the space and reduce the cost of SMS needed to send the data.

Generally, SFormBD can be implemented by using workflow as seen in figure 14. First, we need to develop the SFormBD base data. The data in SFormBD base data consist of predefined data with unique IDs as will described in section 3.2.1. It is, as a reference to construct the form from the SMS. The idea is, formatted SMS that is sent and received by users are minimising the characters by only defining the ID of its component. In this way, the more complete is the SFormBD base data created in both in-field and in-office local data, the more possible it is to compact the SMS. Additionally, the SFormBD base data stored in the local mobile device need to be in the same state with the server so that the form construction is referring to the same reference. Thus, the use of SFormBD needs to implement database synchronisation method between base data stored in mobile device and base data stored in the server.

Next, after base data is deployed in both in-field and in-office user devices, we can use them in the system. The system needs to define SMS formats that are valid to use in the system, so then we can parse them into significant parts and store them into the database. SFormBD Formatted SMS is used to simplify the information that is sent or received by each individual that will be matched to the SFormBD base data that is needed as a reference to construct the form.

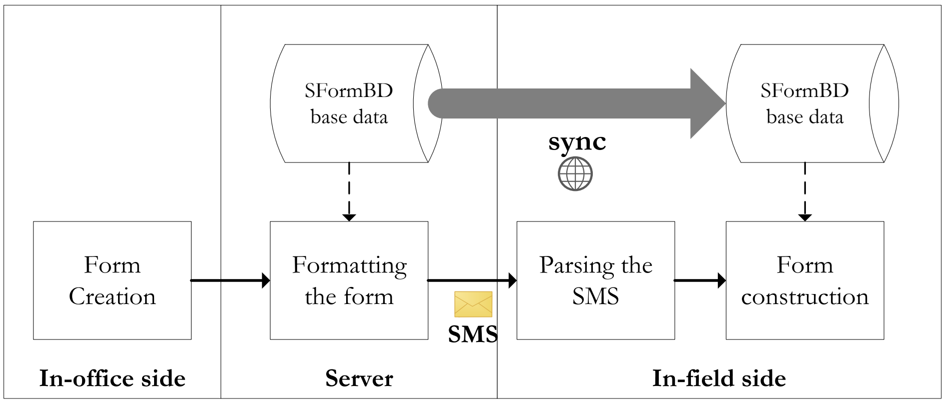


Figure 14. General use of SFormBD

3.2.1. SFormBD Base Data

SFormBD base data are constructed to store base information needed to build a form in the application. First, we inspect the way form is constructed. Compared to form constructed by XML, JSON, or XML, form constructed by SMS need to limit the content, so it contains only highly important information. We propose to leave other information but three sections as seen in Figure 15: question, question text, and question choice. In this way, we can keep what is needed to construct the form. In SFormBD, question section shows the type of question, question text section shows the question asked and, finally, the question choices show the available choice that can be picked for multiple answer questions.

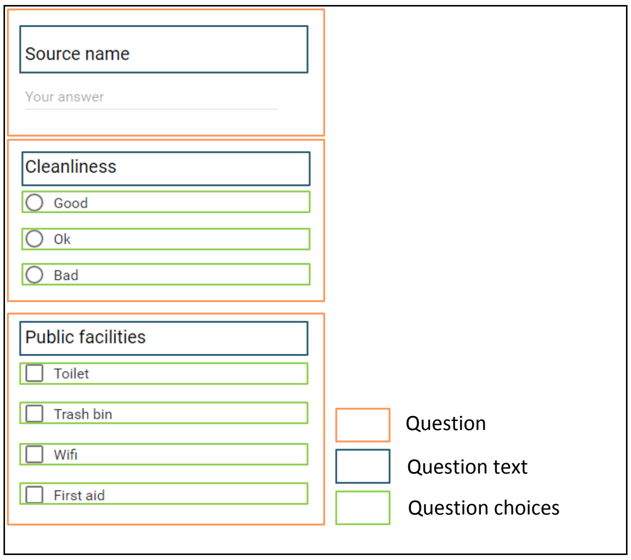


Figure 15. Breaking down the form to create SFormBD

- Question text section

While question section in SFormBD store information about the type of question, question text data in the database will be used as a base question text to limit more space of data transmission. For example, instead of sending a question like "The availability of public facility" every time the tasks need to include the question in the form, the system can just send the ID of the question and then the application will recreate the form based on the SFormBD on their local database. Table 4 shows the example of base question text that can be added to the database.

Table 4. Example of question text in SFormBD base data

ID	Question Text
T1	Source name
T2	Cleanliness
T3	The availability of public facility

- Choices section

The idea of choices section in SFormBD is the same with question text. Instead of sending choices like 'very good', 'good', 'bad', 'very bad' in one question every time it needs, the implementation can just send the id of the choices list. Some choices can be listed to several choices list, especially scaling options like excellent, good, bad, etc. Some choices list can also be matched with several kinds of question. Table 5 shows the example of base choices, Table 6 shows the example of base choices list name, and Table 7 shows the example of base choices list combination that can be added to the database.

Table 5. Example of choices in SFormBD base data

ID	Choices Text
C1	Toilet
C2	Trash bin
C3	Wifi
C4	First aid
C5	Bad
C6	Ok
C7	Good
C8	Very good

Table 6. Example of choices list name in SFormBD base data

ID	Choices List Name
CL1	Quality
CL2	Public facilities

Table 7. Example of choices list combination in SFormBD base data

Choices list name ID	Choices text ID
CL1	C8
CL1	C7
CL1	C6
CL1	C5
CL2	C4
CL2	C3
CL2	C2
CL2	C1

- Question section

Question section provides references to the parameter needed to construct each question in the form. In SFormBD, the most important component that needs to be sent is the type of the question. There are several types of question, for example, a simple question with text answer or multiple choices question with single or multiple answers. Question data in SFormBD store the type of question and its valid component. The example of question type can be seen in Table 8. From the table, we can conclude that the question types are mostly divided into two; the question with choices, and question with no choices. In case of question with no choice, then the question text is the only component that needs to exist in the data. Question text is the question that will be written in the form. On the other hand, question with choices must have listed the option text it has.

Table 8. List of question type in SFormBD base data

ID	Question type		Element Description
QT1	Text	Text	QuestionText
QT2	Text	Paragraph	QuestionText
QT3	Multiple choice	Radio button	QuestionText ,Option1, OptionText1, …, OptionN, OptionTextN
QT4	Multiple choice	List	QuestionText, Option1, OptionText1, …, OptionN, OptionTextN
QT5	Multiple choice	Checkboxes	QuestionText, Option1, OptionText, …, OptionN, OptionTextN
QT6	Linear Scale	Scale	QuestionText, Value Min, Value Max, …, ValueText Min, ValueText Max
QT7	Grid	Grid	QuestionText, Row1, RowText1, …, RowN, RowTextN

Additionally, in question section, SFormBD can also store a full question ready to use in the form. The structure of the question data can be seen in Figure 13. A full question in SFormBD consists of question type, question text, and choices list (if any) as seen in Table 9.

Table 9. Example of full question base data in SFormBD

Question ID	Question Type ID	Question Text ID	Choices List ID
Q1	QT1	T1	Null
Q2	QT3	T2	CL1
Q3	QT4	T3	CL3

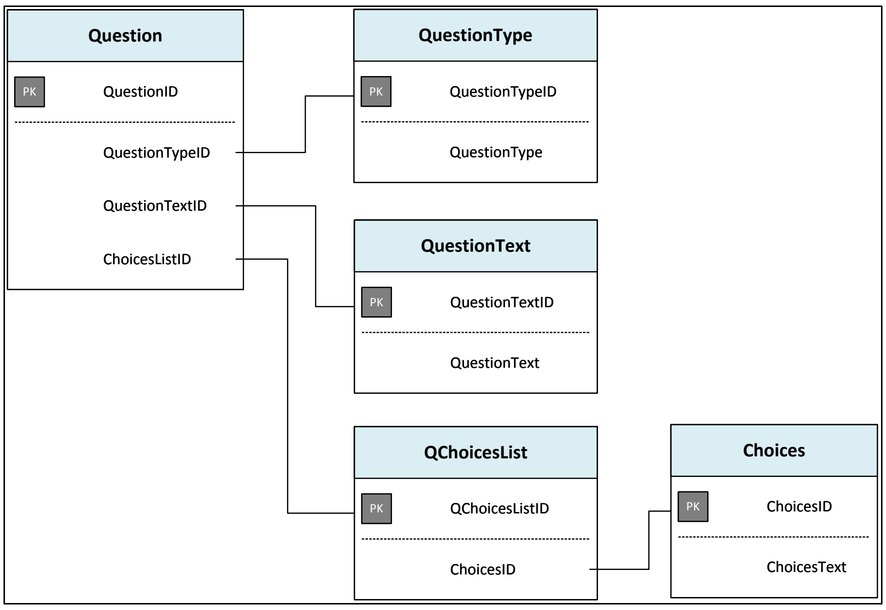


Figure 16. Question data structure in SFormBD

- Form Section

The purpose of form section is to store a complete form component because it is possible to reuse a form in a different location, different in-field user, or different range of time. The form name can be seen in Table 10 and form components can be seen in Table 11.

Table 10. Example of form in SFormBD

ID	Form name
F1	Restaurant Quality
F2	Tourism spot facilities

Table 11. Example of form component in SFormBD

Form ID	Question ID
F1	Q1
F1	Q2
F1	Q3

- **Location Information**

While the form is crucial information to be shared in data collection, sharing the geographical data is also important in geocollaborative works because it provides the location of the impending task. Depending on the format of the location information, direct referencing, indirect referencing, and hybrid referencing can also be implemented based on the reference of the location information. Direct referencing has been used in data collection where user sharing coordinate (X, Y) of the location. In indirect referencing, all location is predefined in the local database with unique ID as a reference to the location. Hybrid referencing are getting popular because they do not need an extensive database as a reference to look up for the data. For example, What3word that named coordinate as three "random" words are using tessellation for tiling the space of a plane into grid square 3 x 3 m in size without gaps or overlaps (Barr, 2008).

Depending on the availability of the data, the size of the data, the use case, and the hardware capability of the mobile device, the use of location information in the tasks can be different. For example, we can use indirect referencing to minimize the characters used because we only need to send the IDs of the location. We recommend using this reference because it uses the same approach with another component that we send via SMS, which is sending the IDs of the location data that already stored in local database. In case of using indirect referencing, we need to save local location of interest layers in the mobile application. Which then we need to assess whether the size of the reference data we need to put in the local database fit the hardware specification of the mobile device we are planning to use.

By using indirect referencing, for example, we have a location that has been saved in the local database with ID 2322 (4 characters). Compared to indirect referencing, direct referencing and hybrid referencing most likely will have more characters. In direct referencing with XY coordinate, depends on the precision the system want to achieve, we have to write, for example, -5.81226, 110.45375 (18 characters) to point at a location. On the other hand, by using what3words, we need to send "guards.touchingly.receives" (26 characters) to point at the same location by using what3words.

3.2.2. SFormBD Formatted SMS

Formatted SMS in the SFormBD structure is designed in such a way that it is possible to send not only a pre-stored form in SFormBD base data but also a brand-new form with entirely new components. In this way, the system can send a new form to in-field user in the right format, even though the in-field user does not have the most updated SFormBD base data at the time the form is being sent. This capability is supported by parsing mechanism that checking if the component in the SMS is already in the local SFormBD base data. This mechanism will be described in detail in Section 3.2.3.

To distinguish different parts in the SMS, formatted SMS in the SFormBD can separate each question using a parser tag. Following tag in JSON format, it is convenient to use curly bracket parser "{}" as a parser to each question in the formatted SMS. In this way, it is clear to see the separation of each section. However, it is not necessary to have both opening and closing tag for SMS to indicate that the question is in the middle of the open and close bracket. The Formatted SMS only need a parser character to indicate that the next character belongs to next section until a parser found again. In this way, we can save at least one character for every section. For example, we can use semicolon character (;) as the parser of the question and caret character (^) as parser between question component as seen in Figure 17. Thus, in case of question with no choices, we can use QuestionTypeID^QuestionText format, while in question with choices, we can use QuestionTypeID^QuestionText^Choices1^Choices2;… format.

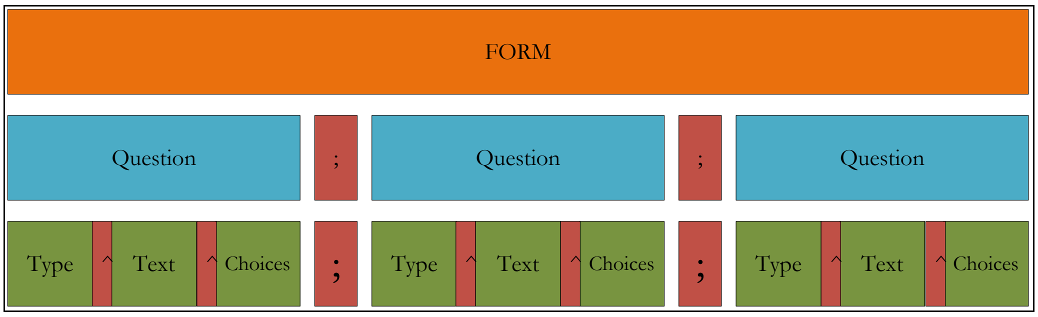


Figure 17. The format of SFormBD Formatted SMS

In Figure 17 we can see how each component of the form is placed in the formatted text. Using the same hierarchy, we can set either the predefined components that already exist in the SFormBD base data or the new one. The example of the SFormBD Formatted SMS can be seen in Table 12.

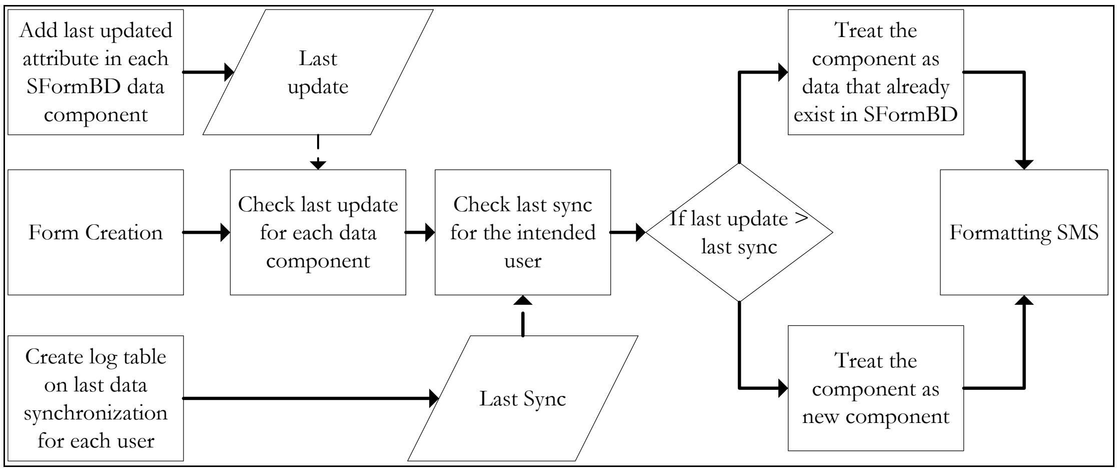
Table 12. The example of SFormBD format based on the availability of SFormBD base data

Case	SFormBD Format
Form exists in SFormBD base data	FormID
Form does not exist, but the question exists in the SFormBD base data	QuestionID;QuestionID;..;QuestionID
One question does not exist in SFormBD base data (with no choices)	QuestionID;QuestionID;QuestionTypeID^QuestionText
One question does not exist in SFormBD base data (with choiceList that already exists in SFormBD base data)	QuestionID;QuestionTypeID^QuestionText^choicesListID
One question does not exist in SFormBD base data (with choiceList that does not exist in SFormBD base data)	QuestionTypeID^QuestionText^choicesText1^…^choicesText N

3.2.3. SFormBD Base Data Synchronisation

In synchronized works, we need to make sure that the user is using the same data. In areas without internet connection, data synchronisation capability ensures that data in the server is always in the same state with data in the local database of the mobile device. This capability usually run when the mobile devices find an internet connection. However, it is also possible that in-office user needs to send new data that does not exist in the local database because the in-field user has been offline for too long. In SFormBD, the step that needs to be taken to fill the gap is shown in Figure 18.

First, we need to know when the last time the intended users synchronised their database. Some cloud-hosted database provides an offline setting that is required for this procedure. For example, Firebase Real-time Database which uses data synchronisation for every data changes instead of using HTTP request ("Firebase," 2017). That is why this data synchronisation capability in SFormBD will not be discussed in detailed in this works. Second, we need to add the last update for each data record in the SFormBD base data database. Then, we need to compare the last synchronisation time of the intended user with the last update of the form component we are using to see if the data are already available in in-field user local database. In this way, we can send the right data in the right format as seen in Table 12.



Add last updated attribute in each SFormBD data component
Last update
Treat the component as data that already exist in SFormBD
Form Creation
Check last update for each data component
Check last sync for the intended user
If last update > last sync
Formatting SMS
Create log table on last data synchronization for each user
Last Sync
Treat the component as new component

Figure 18 Formatting SMS procedure in regards to data synchronization

3.2.4. SFormBD Parsing Mechanism

The steps are as following:

```
Step 1:  Parsing the SMS by the parser tag into arrays question[ ]
Step 2:  Check if the form is already in local database
         if count(question) == 1 {
            check form base data ID. if question[0] is in database {
               Step 7
            }
            else {
                step 3
            }
Step 3:  Parsing each member of array in question[ ] into question component array qComponent[ ]
         based on its parsing character
Step 4:  Check if the question is already in the local database.
         if question is in database {
            step 7
         }
         else {
            Update SFormBD base data
         }
Step 5:  Identify each component of the question based on its position in the array
         n=0;
         qComponent[0] = QuestionType
         qComponent[1] = QuestionText
         if (count(qComponent) > 2) {
            for (i=2, i<count(qComponent);i++){
                     choices[n] = qComponent[i];
                     n = n+1;
`           }
         }
Step 6:  Check if question components are already in database
Step 7:  Store form data to task database
Step 8:  Visualise the form based on task database and the base data
```

3.2.5. SFormBD Example

To understand how SFormBD works, we follow the proposed workflow to create a form as seen in Figure 15. Question 1, which is source name, is a text question type with no choices. Question 2 and question 3 question with choices. After SFormBD is settled, we need to design the SMS format that is valid for the system. Once it arrives at the designed user, the formatted SMS will be parsed into the component needed.

Figure 19 shows the example of formatted SMS to send a brand-new form where the component of the form that does not exist in the SFormBD base data of the client. We can see that the question text and choices for every question are fully typed. SFormBD can make its full potential when the base data prepared in the database stores as many data as possible. So, for example, instead of putting cleanliness (11 characters), we only need to put T2 (2 characters) if we have the data as seen in Table 4. Instead of putting good (4 characters), we can only put C6 (2 characters). Even better, we can just type F1 in the formatted SMS to send a brand-new form in the task to in-field user if the data is available in the SFormBD base data.

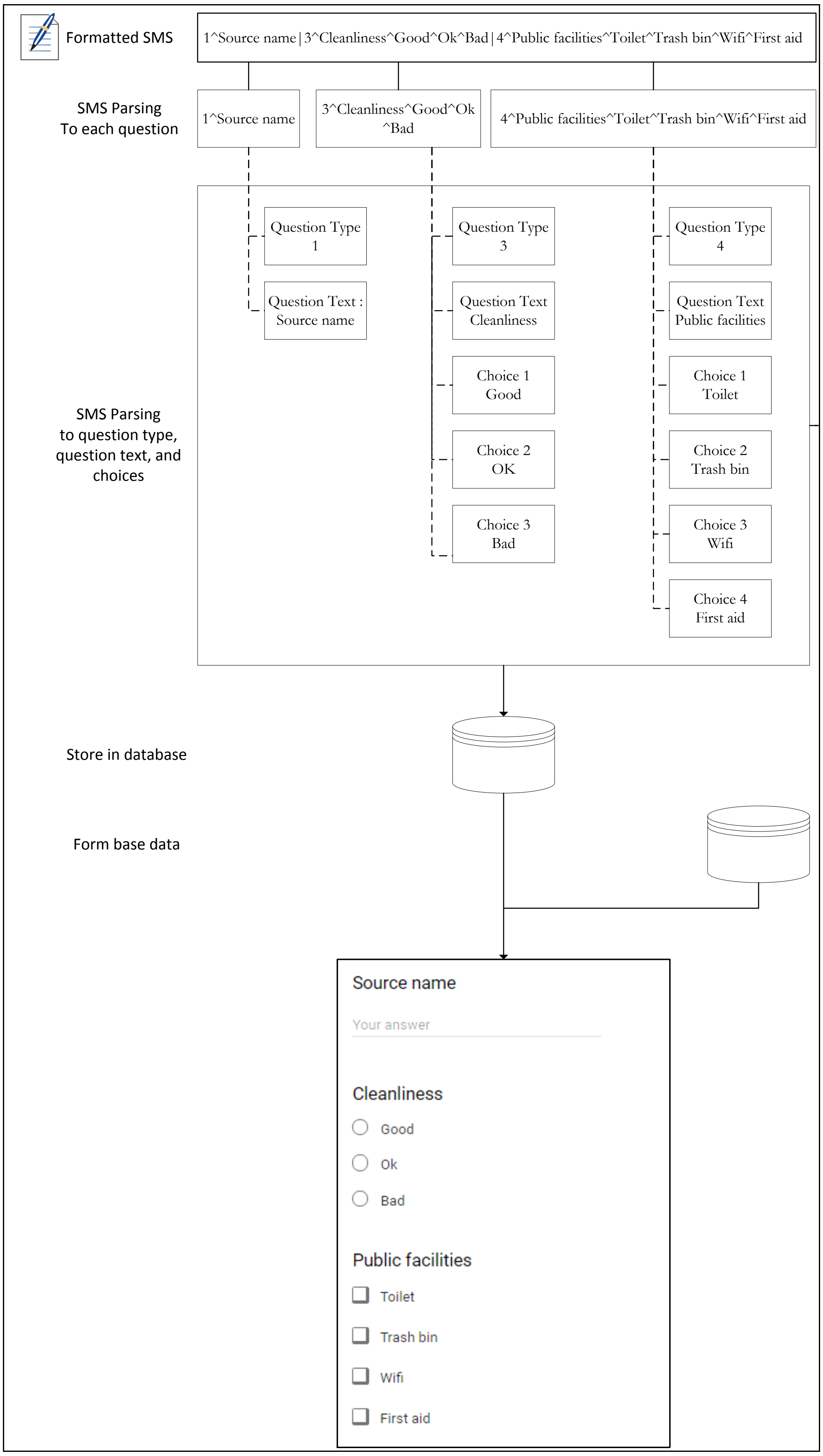


Figure 19. The implementation example of SFormBD workflow

3.3. Summary

This chapter summarises the workflow of the distributed synchronous geocollaborative analysis and creates the link of the information flow. Compared to the previous research, we assess the gaps that have not been covered by the previous research which then we propose a solution. The result is the need for a new feature that utilises SMS as communication channel so that in-office user able to send a new form that needs to be filled as soon as possible. In areas without internet access, the new feature is expected to be a solution for distributed synchronous geocollaborative system in case of urgency.

We propose a new mechanism called SFormBD that make use of SMS to send the data. SFormBD is combining the strength of local database and structuring the form into a compact format so that it cost less to send the SMS. The workflow of using SFormBD starts with the development of SFormBD base data that stores basic component of the form, for example, the question and its component that is likely to be asked in the form. The SFormBD base data is then stored in the local database of in-field mobile device. To keep the data synchronized, we can implement a sync mechanism.

By using SFormBD data as a reference, we just need to send the IDs of the components. However, to accommodate offline setting, where there is no guarantee that in-field users will keep updated, the form is structured in such a way that it can send the form even if the local database has not been recently updated. In this scenario, the result of the compacting the form is not as compact as the one with most updated SFormBD data. However, this mechanism is needed to cover any situation such as the effect of disconnected internet network. In this way, we can conclude that the more complete and more updated is the SFormBD base data in both in-field and in-office user, the more compact is the SMS that needs to be sent. Hence, the cost of sending the SMS is lesser. Finally, we provide the workflow and an example of SFormBD implementation

4. DEVELOPING A PROTOTYPE TO IMPLEMENT SFORMBD

In Chapter 3, we discussed gaps in the previous research and practices of distributed synchronous geocollaborative system in areas without internet access. In the end, we developed a mechanism that allows in-office send a new task to in-field users even though the in-field user does not have access to the internet. To evaluate the mechanism design that we propose in Chapter 3, we develop a prototype to see if the design improves the information flow by allowing in-office user to send a new form to in-field user immediately even when the in-field user is in areas without internet access. For that aim, the geocollaborative system is implemented based on the following scenario:

"A tour guide is guiding his/her tourist in an uninhabited island, where ships come every three days. The trip has been set for six days. However, in the day-5, he finds that the wind surprisingly becomes much stronger, there is a massive rain for half day that surprisingly makes most of the site becomes extremely unpleasant; the land becomes muddy, and the water is not suitable for water activity. He still reports current situation to the office, even though the weather comes back to normal and the tourists decide that they still want to finish the trip until the last day.
The office side receives the report and analyses the situation. Because there will be next trip to the island in the next two days, the office wants to monitor the place, whether it will be a good idea to send their next guess there. The island does not have an internet connection, even though they will find cellular signal in some spots of the island. Even so, the office needs to send the tour guide the monitoring form regarding the current situation of the tourism spots there. The tourist guide needs to send back the results periodically until he/she come back to the base camp."

By using the scenario, we can implement all three information flow described in the geocollaborative workflow. However, the geocollaborative setting requires the prototype to have the other component described in Table 3. Therefore, we integrate all the component into a simple prototype for the tourist guide, and office side to work together on a geocollaborative setting. The application is developed into two based on the users: a mobile application for tourist guide, and web application for office side. The focus is the data exchange between this two application in areas without internet access by using SFormBD.

The discussion starts with the scope of the system, including requirement and architecture design for the software and hardware setting, to explain the required costs to set-up such a geocollaborative environment. Then, we briefly describe the area of interest and the data use in the prototype to complete the prototype. Then, after the prototype ready, we test the prototype in two phase: (i) based on the required functionality, and (ii) by performing the scenario where SFormBD is needed.

4.1. The Scope of the System

The scope of the system defines the boundary of the system and the expected result that will be delivered in the end. Firstly, to determine the boundary of the system, we summarise the workflow that will be covered in the prototype. The workflow can be seen in Figure 20. In the workflow, we can see that the works start simultaneously from both office side and tourist guide. The trigger of the works depends on the situation, so both users have capabilities to create the tasks. Tourist guide can report situation from the field to office side to get their judgement. On the other side, office side can also send tasks to tourist guide when they need data directly from the field. The work is complete when office side does not have anything that need follow up from tourist guide and decide to say so.

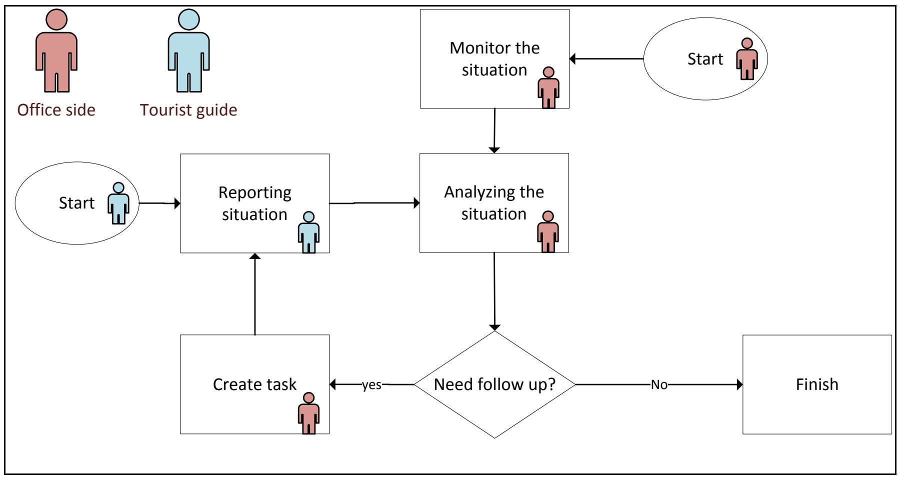


Figure 20. Workflow of the prototype

The scope of the system is defined by describing: (i) the use case diagram to summarise who use the system and what they can do with it and (ii) functional requirement of the system to summarise the goal for each use cases.

A use case diagram summarises the relationship between actors, action, and the system. Figure 21 shows the use case diagram for the prototype. There are five main cases that are developed: (i) create reports for the tourist guide, (ii) send tasks for the office user, (iii) do tasks for tourist guide, (iv) send task result, and (v) monitor the tasks. In the prototype, we can see the task can be created from both sides. Tourist guide can create tasks to office user by creating a report, and office user can create a task to tourist guide by creating the form. A tourist guide will collect data from the field, and send the filled form to the office side. The SFormBD will be implemented in how office user sends a task to tourist guide in case of areas without internet access.

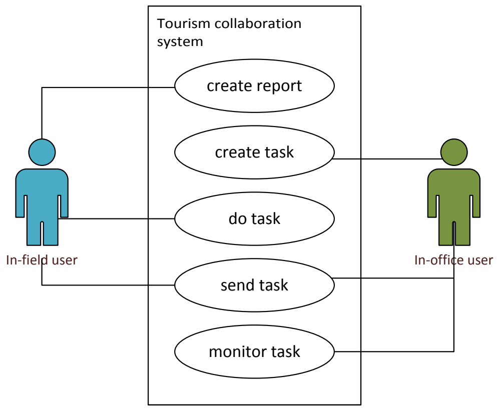


Figure 21. Use case diagram

Functional requirement describes the main functionalities of the application and what the system intended to do that can make the use cases possible. To ensure that improvement in handle information flow in both areas with internet access and areas without internet access, the prototype will focus on the functionality to send task in both areas. Figure 22 shows how the prototype handles information sharing in each network condition. By default, every work done by tourist guide will be saved in the local database. Later, the system needs to send the data via selected communication strategies to office side once it finds the network connection. In weak connections, the prototype will automatically choose the communication strategies based on the urgency setting it has.

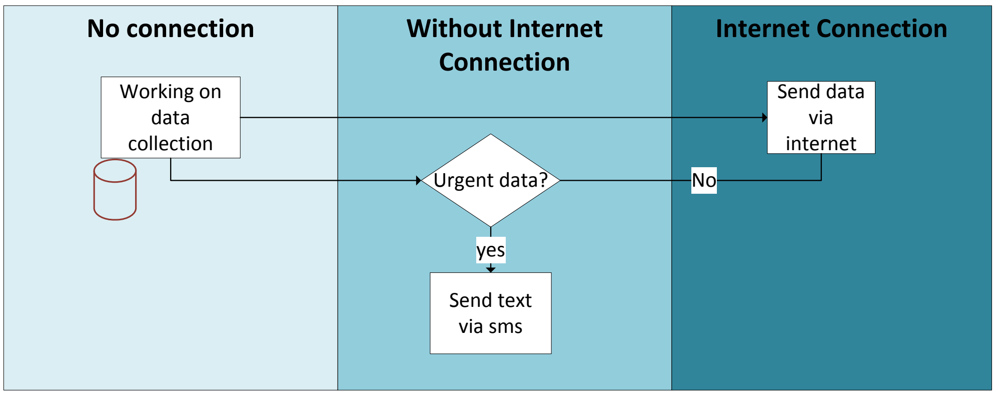


Figure 22. Working with different network coverage

To handle data transfer in various network connection, the prototype needs to have capabilities in sending data via both SMS and internet. The selection of the communication channel to use is based on the connection setting and the urgency of the data. Thus, to summarise the functional requirement to its network connection, the functional requirement in this document are divided into two part based on the application: functional requirement for mobile application and functional requirement for web application. Table 13 shows the functional requirement for mobile application and Table 14 shows the functional requirement for the web application. FR02, FR05, FR08, FR14, and FR17 are functional requirements that utilise the SFormBD, and this functionality is the focus of this prototype development.

Table 13. Functional requirement of mobile application

Requirement ID	Requirement Definition
FR01	Fill report form in the field
FR02	Send report data using SMS
FR03	Send report data via Internet connection
FR04	Save report data to local database
FR05	Receive form via SMS
FR06	Receive form via Internet
FR07	Fill the form
FR08	Send filled form using SMS
FR09	Send filled form using internet connection
FR10	Save filled form
FR11	Visualize location data in a map interface

Table 14. Functional requirement of web application

Requirement ID	Requirement Definition
FR12	Monitor the report
FR13	Follow up the report from tourist guide via Internet
FR14	Follow up the report from tourist guide via SMS
FR15	Send form to tourist guide via the internet
FR16	Send form to tourist guide via SMS
FR17	Visualize location data in a map interface

4.2. System Architecture

System architecture in this section describes the selected solution that meets all technical and functional requirements of the system. Based on the functional requirement of the prototype, what we need are: (i) hosting server to store the web application, (ii) database server to store data, (iii) web programming to create web application, (iv) mobile programming to create mobile application, and (v) SMS gateway to send and receive SMS for the web application. Figure 23 shows the system architecture propose in the research that defines software elements that are used in the system and the relationship among them. The architecture is divided into two parts: server side and client side.

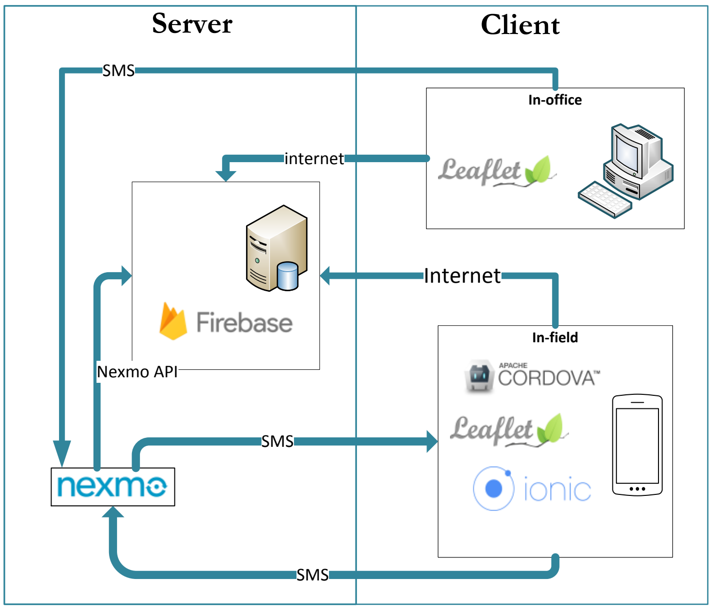


Figure 23. System architecture of the system

On the server side, the prototype uses Firebase database and Firebase Hosting as the environment to build the backend system of both mobile and web application. Firebase is a platform that provides tools and infrastructures to build mobile and web apps. It has capabilities to support online and offline setting that will be useful in the implementation of geocollaborative system. On the other hand, to support sending and receiving data via SMS on the server, Nexmo (Nexmo, 2017) is used as SMS Gateway because it provides SMS API that works with web developments.

On the client side, both web application and mobile application are developed using basic web programming (HTML, JavaScript, and CSS) as a basic visualisation and communication tools to the server and Leaflet JavaScript library (Agafonkin, 2015) to handle spatial visualisation. Ionic framework (Ionic, 2016) and Apache Cordova (The Apache Software Foundation, 2015) are used as a framework to build the mobile application.

4.2.1. Area of Interest

The area of interest that is used in the prototype is Karimun Jawa National Marine Park, north-west Central Java, Indonesia. The Archipelago consists of 27 islands, but only seven islands are inhabited. It is one famous destination with both land and water tourism activities that are well-distributed over the islands. For example, snorkelling, diving, tracking the forest, and exploring the cave. Based on network connection coverage given by Indonesia's Central Statistical Agency, network coverage in Karimun Jawa areas are divided into 3: No connection, weak connection, and strong connection. Assuming that strong connection is a coverage where there is no problem with internet connection, weak connection is where the internet is not available in the network, and no connection is the areas that is not connected to the network, Figure 24 shows the network coverage distribution map of the area of interest. The distribution of network coverage in Karimun Jawa covers not only areas with internet connection, but also areas without internet connection, that makes implementation of SFormBD makes sense in this area.

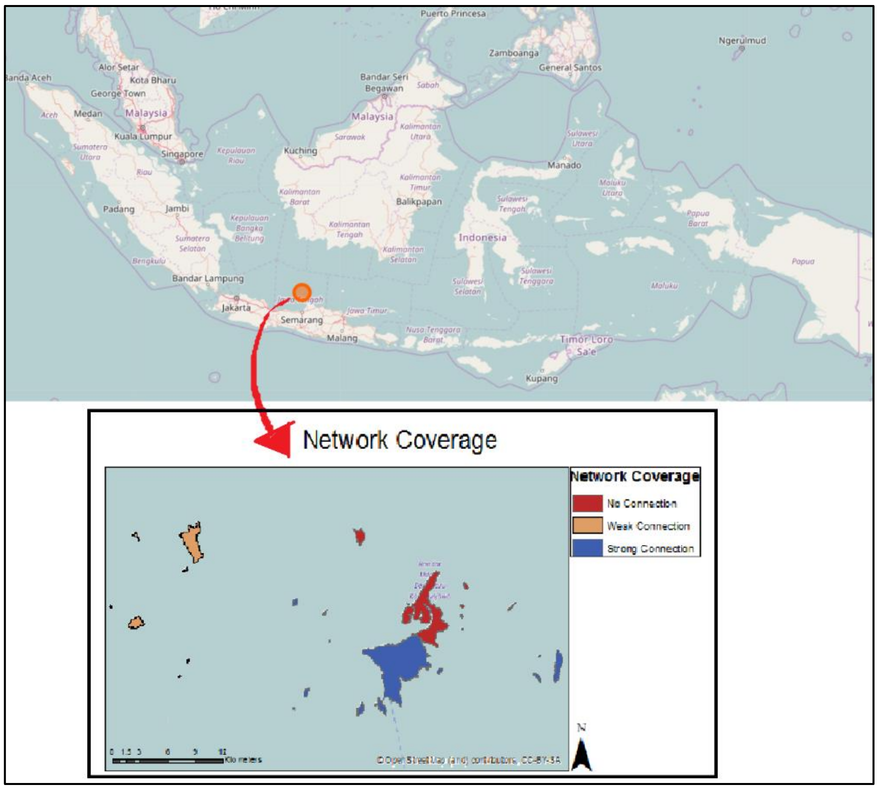


Figure 24. Karimun Jawa island as area of interest and its network coverage (2014)

The prototype provides a simple interactive map of Karimun Jawa as part of a geocollaborative system that uses boundary data from Indonesia's Central Statistic Agency (BPS).

4.3. Prototype Implementation

This section discusses technical steps that have been done to build the prototype. First, we describe the technical setting on how we develop the components of the prototype. Then, we display the results for the selected component. There are three main components that will be discussed. First, we discuss the environment for building the prototype, which includes SFormBD implementation design and the database implementation. Second, we discuss the implementation of the mobile application especially, in regards to the scenario: (i) how to send a report to the office, (ii). How to receive a form using SMS, and (iii) how to send back the filled form to the office. Finally, we discuss the implementation of the web application especially, in regards to the scenario: how monitor report from the tourist guide, and how to receive the data via SMS.

4.3.1. Working with the Environment

SFormBD Implementation Design

SFormBD implementation is done by doing following steps: (i) prepare the SFormBD base data, (ii) defining SMS format for the form, and (iii) defining whole SMS format for information sharing. First, to define SMS format for the form, we limit the implementation for only several data. For example, we limit the type of question that can be used in the prototype as shown in Table 15. We pick two question type as representatives of two type of question that we mentioned before: one of the question types represent question with no choices (text), and the other one is question type that represents multiple choices question (radio button). We use vertical bar (|) parser to identify the next question, colon (:) parser to identify the next component of the question, and semicolon (;) to identify the next component of the choices. We decide on the parser character based on the character that is unlikely needed to be written when a user fills the form and programmatically error that happens during the development when using other characters (number sign '#' do not work during the web development).

Table 15. Question type table in SFormBD

QuestionTypeID	Question Type	SFormBD Format
1	Text	QuestionTypeID:QuestionText
2	Radio button	QuestionTypeID:QuestionText:Choices1;Choices2;…

Second, to define the SMS format, we need to make a list of information that is being sent or received. The general SMS format can be seen in Figure 25. Every SMS must be identified by TextTypeID so that the system can recognise the content it belongs. ID represents the ID of the information that is being sent. Location information represents the latitude and longitude or LocationID (if using direct referencing). TextTypeID is determined by listing all information that is being transferred in the system.

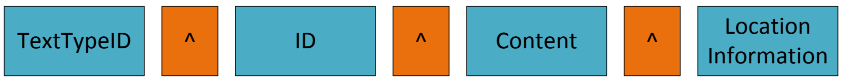


Figure 25. General SMS format used in the prototype

This list of information will help to identify what kind of information it is. In this prototype, there is three information, as discussed earlier, that is being sent or received. This information can be as seen in Table 16. Content to be sent are data that need to be written in the main message, and attached content are data that can be automatically added to the database. For example, the username can be added by acknowledging the username of the application. Status of the report will be automatically set as open once it is stored, while the timestamp will be added automatically using server timestamp whenever data is stored in the database. After the information types in the flows are recognized, then we can decide on the format of SMS for each information as seen in Table 17.

Table 16. Information type list for the prototype

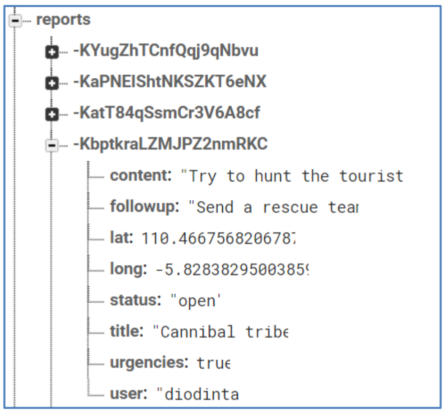
TextTypeID	Information type	Flow	Content to be sent	Attached content
1	Task	Web application to mobile application	TaskID, ReportID, title, lat, long, form	-
2	Report	Mobile application to server (web application)	ReportID, Title, Description, Follow up needed, lat, long, urgencies	User name, timestamp, status
3	Task result	Mobile application to server (web application)	TaskID, Collected data, lat, long	Username, timestamp

Table 17. SMS format for each information type

Information type	SMS format	Array Count
Send Task	1^taskID^reportID^FormComponent^location	5
Send Report	2#reportID#title#description#followup needed#location	4
Send Task result	3#taskID#answerQuestion1|answerQuestion2|…#location	4

Database Implementation

To build the SFormBD in geocollaborative setting, we need to design the data management. Thus, we need to implement the database. The prototype uses Firebase database as a database, a JSON-based database that has different schema and relationship with regular RDMS (Relational Database Management System) that is based on SQL. The example of how Firebase database structure the data can be seen in Figure 26. Reports is a 'table' that stores report data from the tourist guide. Each report is stored as a child that has unique ID.

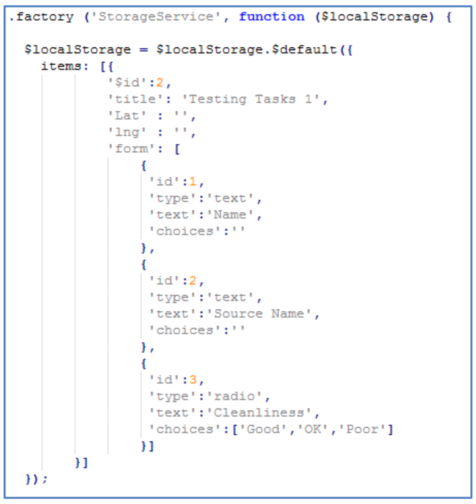


```
reports
    -KYugZhTCnfQqj9qNbvu
    -KaPNEIShtNKSZKT6eNX
    -KatT84qSsmCr3V6A8cf
    -KbptkraLZMJPZ2nmRKC
        content: "Try to hunt the tourist
        followup: "Send a rescue tea
        lat: 110.466756820678
        long: -5.8283829500385
        status: "open'
        title: "Cannibal tribe
        urgencies: true
        user: "diodinta
```

Figure 26. The Firebase database format.

In Firebase database, we need to define it in the program code if we need regular RDBMS capabilities in the system. For example, for filtering data, defining a foreign key, defining attribute type, or join tables. The advantage of using Firebase in a geocollaborative setting are the offline capabilities that write the data to the disk and sync them once the client is connected to the database and its capabilities to know whether the client connected to the database or not.

On the other hand, on the mobile application, the tasks are stored in JSON-based local database by using ngStorage. ngStorage is a free to use AngularJS module that is available on GitHub (https://github.com/gsklee/ngStorage). The tasks are saved in JSON format in the same structure as seen for dummy task data in Figure 27. The data with the same structure also exists in the Firebase database that can be compared and sync to the mobile application if needed. The tasks data are stored in unique ID with attributes as following: title to save the title of the task, latitude and longitude to save location if it needs a special location to be reported, form (question ID, question type, question type, choices).



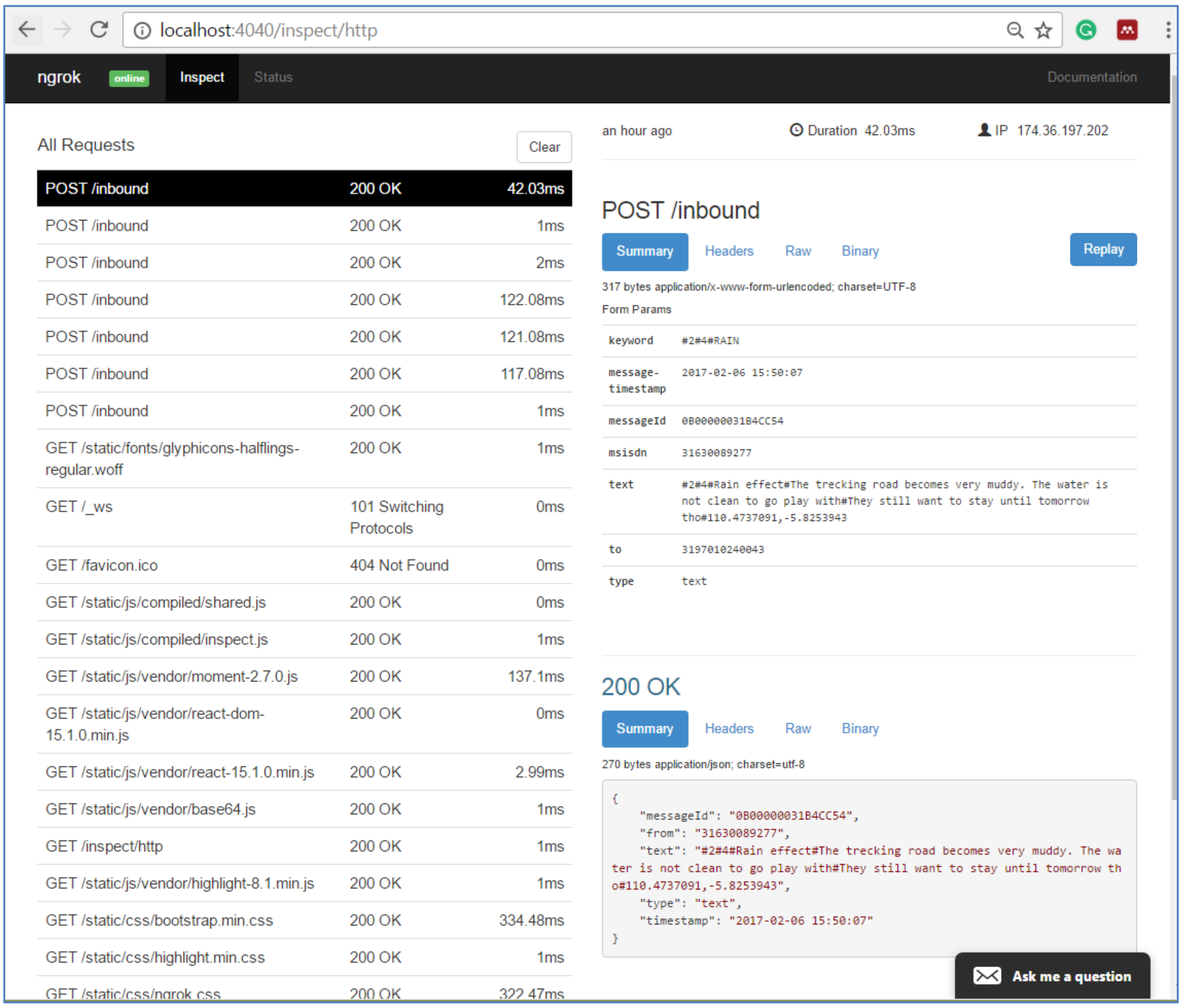
```
.factory ('StorageService', function ($localStorage) {

  $localStorage = $localStorage.$default({
    items: [{
            '$id':2,
            'title': 'Testing Tasks 1',
            'Lat' : '',
            'lng' : '',
            'form': [
                {
                 'id':1,
                 'type':'text',
                 'text':'Name',
                 'choices':''
                },
                {
                 'id':2,
                 'type':'text',
                 'text':'Source Name',
                 'choices':''
                },
                {
                 'id':3,
                 'type':'radio',
                 'text':'Cleanliness',
                 'choices':['Good','OK','Poor']
                }]
        }]
  });
```

Figure 27. Tasks in tree array format

SMS Receiver Environment

Compared to mobile application that has its own SMS receiver, the web application for monitoring need third party tools that allow office user to monitor the incoming SMS. Even though Nexmo SMS API in its cloud dashboard allow us to search for the incoming SMS, directing the incoming SMS into internal system provides opportunities for automating the process after SMS received in the system. Figure 28 shows the inspection page to monitor the incoming SMS.



All Requests

POST /inbound	200 OK	42.03ms
POST /inbound	200 OK	1ms
POST /inbound	200 OK	2ms
POST /inbound	200 OK	122.08ms
POST /inbound	200 OK	121.08ms
POST /inbound	200 OK	117.08ms
POST /inbound	200 OK	1ms
GET /static/fonts/glyphicons-halflings-regular.woff	200 OK	1ms
GET /_ws	101 Switching Protocols	0ms
GET /favicon.ico	404 Not Found	0ms
GET /static/js/compiled/shared.js	200 OK	0ms
GET /static/js/compiled/inspect.js	200 OK	1ms
GET /static/js/vendor/moment-2.7.0.js	200 OK	137.1ms
GET /static/js/vendor/react-dom-15.1.0.min.js	200 OK	0ms
GET /static/js/vendor/react-15.1.0.min.js	200 OK	2.99ms
GET /static/js/vendor/base64.js	200 OK	1ms
GET /inspect/http	200 OK	1ms
GET /static/js/vendor/highlight-8.1.min.js	200 OK	1ms
GET /static/css/bootstrap.min.css	200 OK	334.48ms
GET /static/css/highlight.min.css	200 OK	1ms
GET /static/css/ngrok.css	200 OK	322.47ms

an hour ago   Duration 42.03ms   IP 174.36.197.202

POST /inbound

Summary   Headers   Raw   Binary

317 bytes application/x-www-form-urlencoded; charset=UTF-8

Form Params

keyword	#2#4#RAIN
message-timestamp	2017-02-06 15:50:07
messageId	0B00000031B4CC54
msisdn	31630089277
text	#2#4#Rain effect#The trecking road becomes very muddy. The water is not clean to go play with#They still want to stay until tomorrow tho#110.4737091,-5.8253943
to	3197010240043
type	text

200 OK

Summary   Headers   Raw   Binary

270 bytes application/json; charset=utf-8

```
{
    "messageId": "0B00000031B4CC54",
    "from": "31630089277",
    "text": "#2#4#Rain effect#The trecking road becomes very muddy. The wa
ter is not clean to go play with#They still want to stay until tomorrow th
o#110.4737091,-5.8253943",
    "type": "text",
    "timestamp": "2017-02-06 15:50:07"
}
```

Figure 28. Page for monitor the incoming SMS

4.3.2. Map Visualization

The prototype application provides a simple interactive map with minimum feature like zoom in, zoom out, and panning by using leafletjs javascript library.

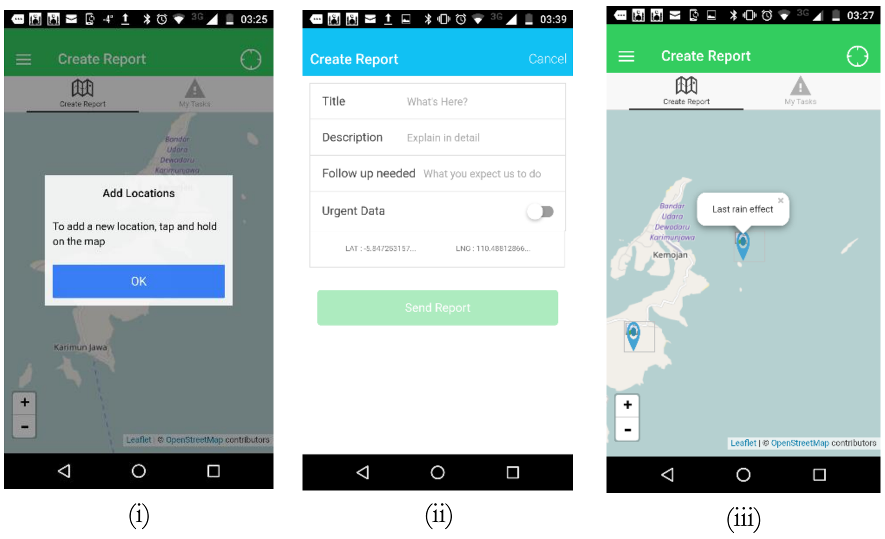
Mobile application

The main functionality of the map visualization in the mobile application is as a reference to input the location data in the form. It allows the system to record latitude and longitude location and send it along with the form submission. We follow the example of Leafletjs in ionic framework application that is available via GitHub in https://github.com/calendee/ionic-leafletjs-map-demo. Figure 29 shows the screen capture of the map visualisation for the mobile application. Figure 29 (i) shows the splash screen to introduce the section to the user and to give a hint to the user how to use the map. Figure 29 (iii) is the map visualisation without splash screen. Figure 29 (ii) is the visualisation of the form that appears after a long push of the map. Notice the latitude and longitude that is recorded using leafletDirectiveMap library as seen below.

```
       $scope.$on('leafletDirectiveMap.contextmenu', function(event, locationEvent){

         $scope.newLocation = new Location();
         $scope.newLocation.lat = locationEvent.leafletEvent.latlng.lat;
```

```
        $scope.newLocation.lng = locationEvent.leafletEvent.latlng.lng;
        $scope.modal.show();
      });
```



Add Locations

To add a new location, tap and hold on the map

OK

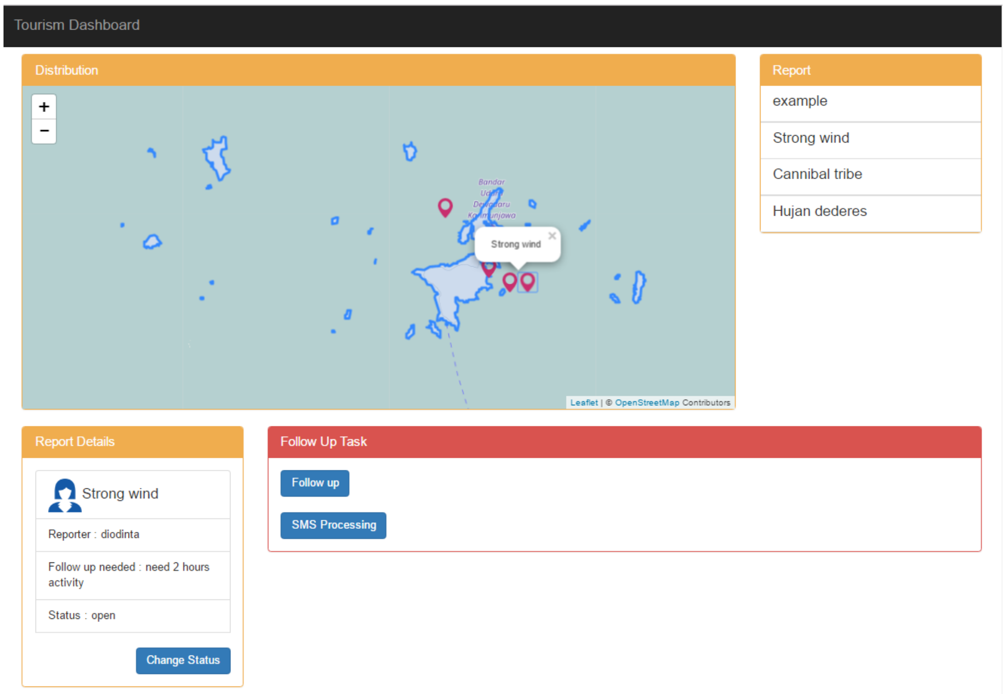
Title	What's Here?
Description	Explain in detail
Follow up needed	What you expect us to do
Urgent Data	
LAT : -5.84/253157...	LNG : 110.48812866...

Last rain effect

Figure 29. The map visualisation in the web application: (i) splash screen, (ii) the form that appears after long push on the map, and (iii) map visualisation without splash screen

Web application

The main functionality of map in the web application is to plot the distribution of the tourist guide report as seen in Figure 30. The monitoring dashboard displays the report with status open, and the point can be clicked to display the report detail.



Report
example
Strong wind
Cannibal tribe
Hujan dederes

Strong wind

Reporter : diodinta

Follow up needed : need 2 hours activity

Status : open

Figure 30. Map visualisation for the web application for monitoring

4.3.3. Sending Data from a Mobile Application

Sending data from mobile application to the server is related to a scenario where the tourist guide report to the office if the situation is so unpleasant to the tourist. The communication channel is chosen by taking urgency component and the type of network coverage into account. In the prototype, the mobile device as the sender has a native capability to recognise the connection they have by using Cordova-plugin-network-information that is available via https://github.com/apache/cordova-plugin-network-information/. After the system recognises the internet availability, the system will take urgency parameter that is chosen by the users when they fill the form. The user needs to choose its urgency by sliding the toggle in the form as seen in Figure 31 which also shows the interface for creating a report form that appeared when a location on the map is clicked.

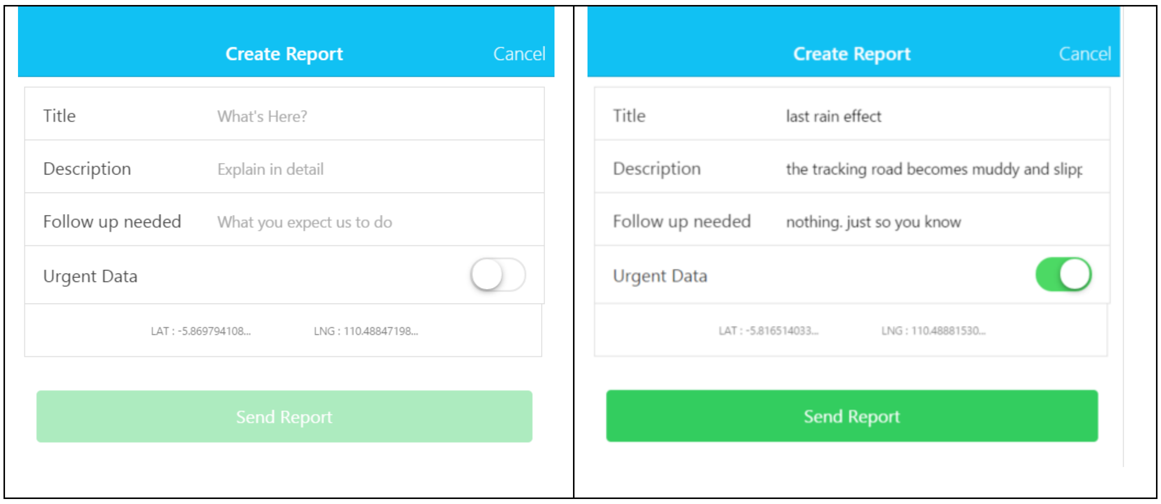


Figure 31. The interface of creating report form

The steps to implement this feature are as following:

Step 1: Create service script to return status of the connectivity.

```
.factory('ConnectivityMonitor', function($rootScope, $cordovaNetwork){
 return {
    isOnline: function(){
      if(ionic.Platform.isWebView()){
        return $cordovaNetwork.isOnline();
      }
    }
```

Step 2: Capture urgency input and create variable of it
Step 3: Put functions save data into local database on send button
Step 4: Create if else script to send the data via chosen communication channel

```
$scope.saveLocation = function() {
        if(ConnectivityMonitor.isOnline()==true){
            LocationsService.savedLocations.push($scope.newLocation);
            $scope.modal.hide();
            $scope.goTo(LocationsService.savedLocations.length - 1);

            $scope.messages.$add({
              title: $scope.newLocation.name,
              content: $scope.newLocation.description,
              followup: $scope.newLocation.followup,
              urgencies: $scope.newLocation.urgencies,
              long: $scope.newLocation.lat,
              lat: $scope.newLocation.lng,
              user: 'diodinta',
              status: 'open'
            });
        }
        else{
            if($scope.newLocation.urgencies==true){
message = '#2#4#'+$scope.newLocation.name+'#'+$scope.newLocation.description+'#'+$scope.newLocation.followup+'#'+$scope.newLocation.lng+','+$scope.newLocation.lat+'';
                console.log(message);
                number = '3197010240043';
                $cordovaSms
                    .send(number, message, options)
                      .then(function() {
                          // Success! SMS was sent
                          console.log('Success');
                      }, function(error) {
                      // An error occurred
                          console.log(error);
                    });//then
            }
            console.log("its not online");
            console.log($scope.sms);
           }
      };
```

The plugin to utilise SMS in the mobile application is cordova-sms-plugin that is available via https://github.com/cordova-sms/cordova-sms-plugin.

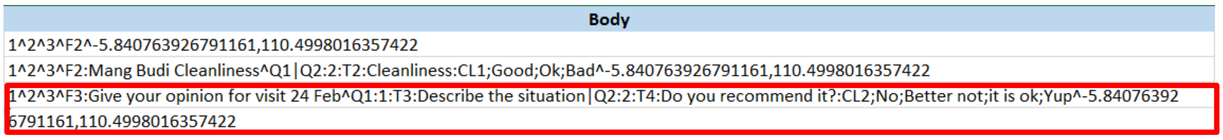
4.3.4. Sending Data from the Office Side

It is also important for the office side to understand the state of client they want to communicate with in term of the connectivity they have. Thus, even though they do not have difficulties in sending and receiving data via internet, they need to know the state of connectivity of the client they want to send the data to. Firebase database has this capability. However, we do not implement it because it requires a lot of time by setting up authentication for the device. By simulating this ability of Firebase Real-Time Database, we have a userStatus parameter that, along with the urgency parameter given by the office user, can be used to decide the communication channel that is used to send the form.

Nexmo SMS API is also used to send the SMS from the web application by using its REST API using following code:

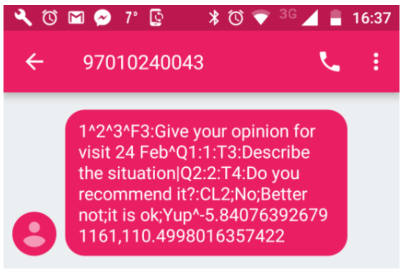
```
window.open('https://rest.nexmo.com/sms/xml?api_key=[redacted]api_secret=[redacted]&from=[redacted]&to=[redacted]&text='+SendSMSFormat+'', '_blank')
```

Figure 27 shows the SMS list that has been retrieved from Nexmo API Dashboard to test the prototype. In Netherland, one SMS cost €0.0761. From the list, we can see capability to split the SMS into 160 characters per SMS automatically. One line represents one SMS that is sent. The red box in Figure 32 indicates SMS that is split into two and the received SMS that is saved in the mobile device that is joined automatically before being processed by the prototype in Figure 33.



Body
1^2^3^F2^-5.840763926791161,110.4998016357422
1^2^3^F2:Mang Budi Cleanliness^Q1|Q2:2:T2:Cleanliness:CL1;Good;Ok;Bad^-5.840763926791161,110.4998016357422
1^2^3^F3:Give your opinion for visit 24 Feb^Q1:1:T3:Describe the situation|Q2:2:T4:Do you recommend it?:CL2;No;Better not;it is ok;Yup^-5.84076392 6791161,110.4998016357422

Figure 32. List of SMS



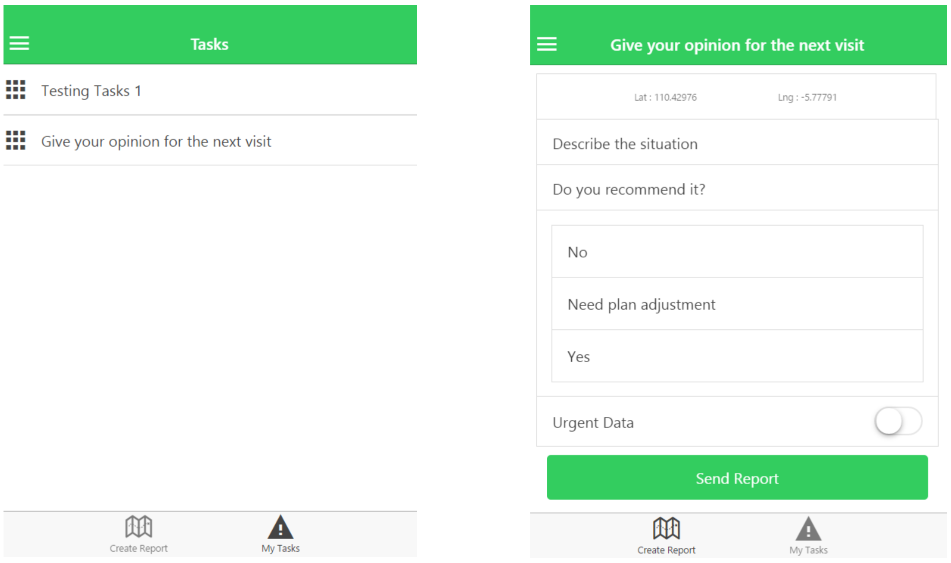
97010240043

1^2^3^F3:Give your opinion for visit 24 Feb^Q1:1:T3:Describe the situation|Q2:2:T4:Do you recommend it?:CL2;No;Better not;it is ok;Yup^-5.840763926791161,110.4998016357422

Figure 33. The SMS received in mobile device

4.3.5. SMS Processing

SMS processing is a mechanism that we develop to parse the SMS and store them into the right structure in the database. The function is developed by using Javascript that uses steps as seen in Section 3.2.4. The parsing method is using a simple string split() method, and iterate the array result based on the structure of the database we need. The complete code can be seen in the appendix while the result can be seen in Figure 34.



Tasks

Testing Tasks 1

Give your opinion for the next visit

Give your opinion for the next visit

Lat : 110.42976 Lng : -5.77791

Describe the situation

Do you recommend it?

No

Need plan adjustment

Yes

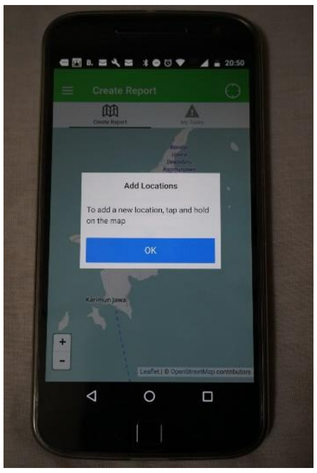
Urgent Data

Figure 34. The form preview after the SMS processed by the system

4.4. Prototype Testing

4.4.1. Testing the Prototype

To test the prototype, the mobile application is installed in Moto G4 Plus with Android Marshmallow 6.0.1 as seen in Figure 35. Functional testing is done to test that the prototype has the expected result as it said in the requirement. By using the scenario we discussed earlier, the testing has been done by acting user that use to work in remote area to collect the data and familiar with mobile application. The testing scenario is described in the appendix. The prototype testing is a success with notes that are discussed in Section 4.4.2. the implementation section where the system screenshot are based on the scenario we did with the prototype.



To add a new location, tap and hold on the map

Figure 35. The installed prototype for the testing.

4.4.2. General Discussion

Even though we do not develop a full-automatic system of distributed geocollaborative system, we develop enough to integrate SFormBD mechanism and test the scenario where this mechanism is used. By developing the prototype, we notice that SFormBD mechanism has several dependencies in compacting the form component. These dependencies are related to finding the best way to use less character for another task components that need to be sent along with the form. The examples in this prototype are the location information and the IDs.

In the SFormBD design, we recommend using indirect referencing to provide location information of the task that needs to be done so that we only need to send the ID of the location information. In the prototype, we implement direct referencing using latitude and longitude coordinate that is directly captured from the map. However, the selection of the location information does not affect the workflow of the SFormBD mechanism, mostly, it affects the number of used characters.

Another dependency is the design of IDs in the application. For multi-user application, there will be a chance of ID conflict for simple IDs like we use in the prototype (an auto increment IDs). In Firebase database, they ensure unique identifier by ordering timestamp and random bits that are resulting 20 characters ID like "-KbvlTwyOERch73pkx97". There are 20 characters for one ID, and yet we potentially need to deliver more than one ID in one SMS. To make potential use of the SFormBD mechanism, we need to pay attention on how we format the ID of the data we use and use as fewer characters as possible.

Thus, despite these limitations, the current prototype is developed with software and hardware requirement that makes it is extendable to another activity that in-field and in-office need with the geocollaborative system. For example, to retrieve and visualise the report data based on temporal and spatial activities, from which we can derive a trend or even acquire useful insight from historical events.

Before implementing the SFormBD into geocollaborative system, we needed to pay attention to two things: (i) the data hierarchy, and (ii) the parser character. First, a careful design of the data hierarchy was crucial. In this prototype, the task was under report because it allows an in-office user to follow up the report if needed, and it would be easier for them to monitor the flow of the tasks. Moreover, this data needed to be delivered during data exchanged. Secondly, we needed to pay attention to the choice of the parser characters. It was necessary to choose a character that is most unlikely to be used in filling out the form. In addition, this character would be such that it did not pose a challenge to coding. For example, in the prototype development, we found that number sign '#' cut the text that is being sent with SMS from the Nexmo REST API.

The successful testing of the scenario that involves the implementation of SFormBD indicates that SFormBD, the proposed mechanism works, and can accommodate the improvement on the new feature we need on the geocollaborative system in areas without internet access. As illustrated in Figure 28, the form is the result of parsing and restructuring mechanism in SFormBD. We tried to test in two scenarios; where the form is already in the SFormBD base data and where the form and its component is not in the SFormBD base data. From this scenario, the statement we have about the completeness of the SFormBD base data is correct: The more complete and update the SFormBD base data in local mobile application, the fewer characters it needs to deliver the information, which then can reduce the cost of sending SMS.

4.5. Summary

In this chapter, we presented a way to evaluate the improvement we need in the distributed synchronous geocollaborative setting by creating a scenario where there was a need for the new feature in the system. The new feature implementation is based on SFormBD mechanism we propose in Section 3. The prototype is a success based on the functional testing, which indicates that SFormBD mechanism we proposed in this work can accommodate the improvement we need to fill in the gap of information flow of distributed synchronous geocollaborative analysis in areas without internet access.

5. CONCLUSIONS AND RECOMMENDATIONS

5.1. Conclusions

Because of a wide scope of implementation of geocollaborative systems involving in-field and in-office users, we narrow down the discussion into distributed synchronised geocollaborative works in data collection. In-field users are in the field to collect data and, simultaneously, in-office users monitor and analysis the data. First, we assessed previous solutions of data collection without internet and how they can cover the geocollaborative component and all links in the information flows. In the end, we found a gap and solve it by developing a new mechanism that allows the in-office user to send a new form to in-field users by using SMS. This new mechanism fills the gap of distributed synchronised geocollaborative system that loses its flexibility to follow up the works during works session.

Following are the results corresponding to the research questions from Section 1.2:

How are typical flows of the information and tasks?
In data collection, the information that needs to be exchanged is the empty form (request) and the filled form (response). The flows are the in-office users sending the empty form to in-field user, in-field users sending back the data acquired to the in-office users. In regard to geocollaborative workflows, the information flows need to be more flexible, because the workflow can start from both sides and both sides can do follow up works. This requirement modifies the typical geocollaborative workflow for data collection as follows: (i) an in-field user has a predefined form to start the works, (ii) an in-office user sends an empty form to in-field user, and (iii) the in-field user(s) are able to send back the result of the new form.

How can we characterise collaborative settings?
Based on empirical research and practices, we point at three critical components in distributed synchronized geocollaborative systems in data collection: (i) use of map interfaces which accommodate location information, (ii) mobile data management which accommodates data storage and retrieval in mobile devices, and (iii) interactive and user-friendly user interface.

What are possible problems for each identified flow in areas without internet access?
Previous research and practices accommodate offline capability as follows: (i) download the form when they have connection or have predefined form in the mobile application, (ii) save the filled form in the local mobile database when they are working in areas without internet access, and (iii) send the data once they find the connection. However, in case of an emergency where they cannot find a connection when the task needs to be finished, then the consequences will be: (i) their work cannot be sent to the in-office users and (ii) there is no collaboration between them and in-office user in the meantime.

What communication protocols do mobile devices have and what are their potentials for information exchange in areas without internet access?

From previous studies and practices, we found three communication channels that have been used to exchange data and information in a collaborative setting. They are voice call, SMS, and USSD. SMS and USSD are the potential use because they are text-based communication channel that we need for the collaboration we discuss earlier

What are their characteristics and their limitations with regards to the identified collaborative workflows?
USSD and SMS can be used as the communication channels to exchange data in areas without internet access, as long as the area is covered by GSM network. From previous research and practices, SMS and USSD are used as text-based data exchange. In regards to the collaborative workflows, USSD covers all the flow while SMS cannot cover the identified flow: an in-office user sends an empty form to in-field user.

What are the options to improve information flows given technological and organisational constraints?
We find two ways to improve information flows in geocollaborative systems. First, by using USSD, we need to find a way to integrate it into the geocollaborative component. Second, by using SMS, we need to add flexibility for the in-office user to follow up the task during the workflow. Considering the limited access to explore USSD technology, we explored on the second option. In the end, we filled the gap in the workflow by creating a mechanism that allows in-office users to send follow-up task, in a new form, to in-field users in areas without internet connection.

Which is an effective and efficient set-up for reliable information flow in geocollaborative setting involving in-field and in-office elements?

Reliable information flow in geocollaborative setting must allow users to exchange data when they need it even if they do not have access to the internet. Especially in urgent situation when the data need to be sent or received immediately. Effective and efficient set up for reliable information flow means that the implementation of geocollaborative system covers all the information flow and other geocollaborative setting components even in areas without internet access.

How can we evaluate the proposed improvement?
We evaluate the improvement design by developing a simple prototype based on a scenario where SFormBD mechanism is needed in urgent situation.

What are the requirements to conduct the evaluation?

In the prototype, we implement SFormBD and test if the office user can send new data collection task to tourist guide through SMS. To evaluate how well the design is, we also implement the other geocollaborative components to see if the implementation of SFormBD integrates well with the current workflow of geocollaborative system.

What is the result of the evaluation?

The success of the functional testing in a scenario where it needed indicates that SFormBD mechanism we propose in this work can accommodate the improvement we need to fill in the gap of information flow of distributed synchronous geocollaborative analysis in areas without internet access.

5.2. Recommendations

Even though the implementation of the prototype SFormBD is limited to text-based data collection, it is a new feature that will improve the workflow of geocollaborative system in areas without internet access. To assess the full potential of the SFormBD mechanism, we recommend integrating the implementation of

SFormBD into the existing distributed-synchronize geocollaborative systems for data collection in areas without internet access.

By implementing SFormBD mechanism in a real use case, we can assess which part of the design need to be extended. It is because the design of SFormBD in this work is limited to a very basic form that might be not enough for some use cases. For example, a user might need to constrain the data type and character length, to set a default value, to set up a hint for every question, or to set the required field. By using the same approach, it is possible to add components into the formatted SMS and adjust the parsing mechanism. Thus, even though this information can be synched later when in-field user has an internet connection, the user might need to consider if it is worth the cost to put the information into the SMS in the first place.

Second, by performing usability test in the real use case where the users have experience with the workflow, we can evaluate how this new feature improves their workflow and their work performance.

LIST OF REFERENCES

Agafonkin, V. (2015). Leaflet - a JavaScript library for interactive maps. Retrieved January 4, 2017, from http://leafletjs.com/

Azavea. (2017). PhillyTreeMap. Retrieved February 9, 2017, from https://www.opentreemap.org/phillytreemap/page/About/

Barjis, J., Kolfschoten, G., & Maritz, J. (2013). A sustainable and affordable support system for rural healthcare delivery. *Decision Support Systems*, *56*(1), 223–233. https://doi.org/10.1016/j.dss.2013.06.005

Barr, R. (2008). *what3words Technical appraisal*. Cheshire.

Bray, T. (2014). The JavaScript Object Notation (JSON) Data Interchange Format. https://doi.org/10.17487/RFC7158

Brovelli, M. A., Minghini, M., & Zamboni, G. (2016). Public participation in GIS via mobile applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, *114*, 306–315. https://doi.org/10.1016/j.isprsjprs.2015.04.002

Cai, G. (2005). Extending Distributed GIS to Support Geo-Collaborative Crisis Management. *Geographic Information Sciences: A Journal of the Association of Chinese Professionals in Geographic Information Systems*, *11:1*, 4–14. https://doi.org/10.1080/10824000509480595

Dasgupta, A., Kamble, R., Ghosh, S. K., & Acharya, P. S. (2013). GeoSMS framework for information acquisition in rural public health management system. *2013 Annual IEEE India Conference (INDICON)*, 1–4. https://doi.org/10.1109/INDCON.2013.6725880

Esri. (2016). Collector for ArcGIS. Retrieved August 11, 2016, from http://doc.arcgis.com/en/collector/

Esri. (2017). Create and share a map for data collection. Retrieved January 17, 2017, from https://doc.arcgis.com/en/collector/ios/create-maps/create-and-share-a-collector-map.htm

Facebook. (2015). Internet.org is connecting the world. Retrieved August 10, 2016, from https://info.internet.org/en/

Firebase. (2017). Retrieved January 25, 2017, from https://firebase.google.com/docs/database/

Freire, C. E. de A., & Painho, M. (2014). Development of a Mobile Mapping Solution for Spatial Data Collection Using Open-Source Technologies. *Procedia Technology*, *16*, 481–490. https://doi.org/10.1016/j.protcy.2014.10.115

GeoODK. (2014). Geographical Open Data Kit. Retrieved December 12, 2016, from http://geoodk.com/

Heard, J., Thakur, S., Losego, J., & Galluppi, K. (2014a). Big board: Teleconferencing over maps for shared situational awareness. *Computer Supported Cooperative Work: CSCW: An International Journal*, *23*(1), 51–74. https://doi.org/10.1007/s10606-013-9191-9

Heard, J., Thakur, S., Losego, J., & Galluppi, K. (2014b). Big board: Teleconferencing over maps for shared situational awareness. *Computer Supported Cooperative Work: CSCW: An International Journal*. https://doi.org/10.1007/s10606-013-9191-9

Ionic. (2016). Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular. Retrieved January 4, 2017, from http://ionicframework.com/

Joshfire. (2014). JSON Form. Retrieved December 12, 2016, from https://github.com/joshfire/jsonform

Kapucu, N., & Garayev, V. (2011). Collaborative Decision-Making in Emergency and Disaster Management. *International Journal of Public Administration*, *34*(6), 366–375. https://doi.org/10.1080/01900692.2011.561477

Kipf, A., Brunette, W., Kellerstrass, J., Podolsky, M., Rosa, J., Sundt, M., … Thomas, E. (2016). A proposed integrated data collection , analysis and sharing platform for impact evaluation. *Development Engineering*, *1*, 36–44. https://doi.org/10.1016/j.deveng.2015.12.002

Lee, M. (2015). Collecting Field Data in an Environment Combining APP and Fusion Tables, 11–16.

Maceachren, A. M., & Brewer, I. (2004). Developing a conceptual framework for visually-enabled geocollaboration. *International Journal of Geographical Information Science*, *18*(1), 1–34. https://doi.org/10.1080/13658810310001596094

Morrow, N., Mock, N., Papendieck, A., & Kocmich, N. (2011). Independent evaluation of the Ushahidi Haiti project. *Development Information Systems International*, 1–36. https://doi.org/10.1109/MIS.2011.52

Nexmo. (2017). Nexmo - APIs for SMS, Voice and Phone Verifications. Retrieved January 4, 2017, from https://www.nexmo.com/

Ochoa, S., Talavera, J., & Paciello, J. (2015). Applying a Geospatial Visualization Based on USSD Messages to Real Time Identification of Epidemiological Risk Areas in Developing Countries: A Case of Study of Paraguay. *Studies in Health Technology and Informatics*, *216*, 396–400. https://doi.org/10.3233/978-1-61499-564-7-396

Puspitasari, Y. (2013). *Emergency information system of IDP (Internally Displaced Persons) needs using SMS gateway for flood disaster emergency response in Sukoharjo Regency, Central Java Province.* Gadjah Mada University and University of Twente.

Quin, L. (2016). Extensible Markup Language (XML). Retrieved December 12, 2016, from https://www.w3.org/XML/

Sa, J. H. G., Rebelo, M. S., Brentani, A., Grisi, S. J. F. E., Iwaya, L. H., Simplicio, M. A., … Gutierrez, M. A. (2016). Georeferenced and secure mobile health system for large scale data collection in primary care. *International Journal of Medical Informatics*, *94*, 91–99. https://doi.org/10.1016/j.ijmedinf.2016.06.013

Sanou, B. (2016). ICT facts and figures 2016. Retrieved July 27, 2016, from http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2016.pdf

Signore, A. (2016). Mapping and sharing agro-biodiversity using Open Data Kit and Google Fusion Tables. *Computers and Electronics in Agriculture*, *127*, 87–91. https://doi.org/10.1016/j.compag.2016.06.006

Smirnov, A., & Ponomarev, A. (2015). Crowd Computing Framework for Geoinformation Tasks, 109–123. https://doi.org/10.1007/978-3-319-16667-4

Suddul, G., Bahadoor, U., Ramdoyal, A., Doolhur, N., Soobul, A., & Richomme, M. (2011). An open USSD enabler to simplify access to mobile services in emerging countries. *International Conference on Emerging Trends in Engineering and Technology, ICETET*, 323–326. https://doi.org/10.1109/ICETET.2011.53

Tayal, R. (2015). Open Data Kit- Use of Smartphone Technology for Surveying, 461–464.

TEXTALK. (2016). Angular Schema Form. Retrieved December 12, 2016, from http://schemaform.io/

The Apache Software Foundation. (2015). Apache Cordova. Retrieved January 4, 2017, from https://cordova.apache.org/

Thomas, J. J., & Cook, K. a. (2005). Illuminating the path: The research and development agenda for visual analytics. *IEEE Computer Society*. https://doi.org/10.3389/fmicb.2011.00006

Triggs, R. (2013). What is SMS and how does it work? Retrieved August 14, 2016, from http://www.androidauthority.com/what-is-sms-280988/

Wang, C., Qiao, Y., Wu, H., Chang, Y., & Shi, M. (2016). Empowering fall webworm surveillance with mobile phone-based community monitoring: a case study in northern China. *Journal of Forestry Research*, 1–8. https://doi.org/10.1007/s11676-016-0230-5

Wouters, B., Barjis, J., Maponya, G., Martiz, J., & Mashiri, M. (2009). Supporting Home Based Health Care in South African Rural Communities Using USSD Technology Supporting home based health care in South African rural communities using USSD technology. *Americas Conference on Information Systems (AMCIS)*, (January 2017).

Wu, A., Convertino, G., Ganoe, C., Carroll, J. M., & Zhang, X. (Luke). (2013). Supporting collaborative sense-making in emergency management through geo-visualization. *International Journal of Human-Computer Studies*, *71*(1), 4–23. https://doi.org/10.1016/j.ijhcs.2012.07.007

X. (2013). Balloon-powered internet for everyone. Retrieved August 10, 2016, from https://www.solveforx.com/loon/

XLSForm.org. (2017). What is an XLSForm? Retrieved January 16, 2017, from http://xlsform.org/

APPENDICES

Appendix 1: Testing Scenario

Scenario	
A tour guide is guiding his/her tourist in an uninhabited island, where ships come every three days. The trip has been set for six days. However, in the day-5, he finds that the wind surprisingly becomes much stronger, there is a very heavy rain for half day that surprisingly makes most of the site becomes extremely unpleasant; the land becomes muddy, and the water is not suitable for water activity. He still reports current situation to the office, even though the weather comes back to normal and the tourists decide that they still want to finish the trip until the last day. The office side receives the report and analyses the situation. Because there will be next trip to the island in the next two days, the office wants to monitor the place, whether it will be a good idea to send their next guess there. The island does not have an internet connection, even though they will find cellular signal in some spots of the island. Even so, the office needs to send the tour guide the monitoring form regarding the current situation of the tourism spots there. The tourist guide needs to send back the results periodically until he/she come back to the base camp.	
User	Tourist guide
Scenario	• Tourist guide create report to in-field user • office user respond with creating task (form) to tourist guide • tourist guide fill the form • send it to office user • Office user end the tasks.
Preliminary condition	• Install application in mobile device • Turn off internet connection on the mobile devices
Testing Scenario: • Open application • Open create report tab in left tab • Select location in the map by long push on the screen. • Create task by filling the form with following text: Title : Description: Follow up : • Toggle on the urgencies • Click submit button	
User	**Office**
Scenario	• Office user send new form to tourist guide
Preliminary condition	• Open application in a web browser
Testing Scenario: • Click on follow up button	

• Select the report from the list • Select the user from the list • Select the form that want to be send • Hit Send Task Button	
User	**Tourist guide**
Testing Scenario: • Process the incoming SMS into form • Fill in form • Toggle on the urgencies • Send Report	

Appendix 2: SMS processing code for the mobile application

```
.controller('resultCtrl', ['$scope', '$stateParams', 'StorageService', 'SFormBDQuestion', 'SFormBDQuestionText', 'SFormBDQuestionChoiceList'
 $stateParams.parameterName
function ($scope, $stateParams, StorageService, SFormBDQuestion, SFormBDQuestionText, SFormBDQuestionChoiceList) {
    var items = [

        ];
    var itemid = $stateParams.input;
    var str = itemid;
    var res = str.split("^");
    var SMSType = res[1];
    if(SMSType == 1){
    }
    var formKet = res[3].split(":");
    var formArray = res[4].split("|");
    var longlat = res[5].split(",");
    var idform = parseInt(res[2]);
    var idTask = parseInt(res[2]);
    $scope.formArray = formArray;
    console.log($scope.formArray);
    console.log($scope.formArray.length);
    var formInputperRow=[];
    if (formKet.length > 1) {
        console.log("it's a new form");
        var formID = formKet[0];
        var formTitle = formKet[1];
        var question = [];
        for(i=0;i<$scope.formArray.length;i++){
            console.log(i);
            var formRow = [];
            question = {};
            var formRow = formArray[i].split(":");
            var Questionitems = SFormBDQuestion.getAll();
            console.log(Questionitems);
            for(var j=0;j<Questionitems.length;j++) {
                var itemQ = Questionitems[j];
                console.log(itemQ);
                console.log(itemQ.QuestionID);
                if(itemQ.QuestionID == formRow[0]) {
                    var QuestionText = itemQ.QuestionText
                    console.log(QuestionText);
                }
            }
            if (formRow.length > 1) { //new question
                console.log("it is a new question");
                if(formRow[1]=="1"){
                    if (formRow.length > 3) { // with new text
                        question = {
                            "QuestionID":formRow[0],
                            "type":"text",
                            "textID":formRow[1],
                            "text":formRow[3]
                        }

                    }
                    else{ //old text
                    var Questionitems = SFormBDQuestion.getAll();
                        console.log(Questionitems);
                        question = {
                            "QuestionID":formRow[0],
                            "type":"text",
                            "textID":formRow[1],
                            "text":QuestionText
                        }
                    }
                }
                else if(formRow[1]=="2"){
                    var QuestionTextItem = SFormBDQuestionText.getAll();
                    console.log(QuestionTextItem);
                    for(var k=0;k<QuestionTextItem.length;k++) {
                        var itemQT = QuestionTextItem[k];
                        console.log(itemQT);
                        console.log(itemQT.QuestionTextID);
                        if(itemQT.QuestionTextID == formRow[2]) {
                            var QuestionText2 = itemQT.QuestionText
                            console.log(QuestionText2);
                        }
                    }
                    var choiceRow = formArray[i].split(";");
                    console.log(choiceRow);
                    if(choiceRow.length > 1 ){
                        console.log("it is a new choice list");
                        var choices=[];
                        for(j=1;j<choiceRow.length;j++){
                            choices.push(choiceRow[j]);
                        }

                        if (formRow.length > 4) {    // new question text
                            var choiceIDSeq = formRow[4].split(";");
                            question = {
                                "QuestionID":formRow[0],
                                "type":"radio",
                                "textID":formRow[2],
                                "text":formRow[3],
                                "choiceListID":choiceIDSeq[0],

                                "choices":choices
                            }

                        }
                        else { //old question text
                                   var choiceIDSeq = formRow[3].split(";");
                                   question = {
                                       "QuestionID":formRow[0],
                                       "type":"radio",
                                       "textID":formRow[2],
                                       "text":QuestionText2,
                                       "choiceListID":choiceIDSeq[0],
                                       "choices":choices
                                   }
                               }

                           }
                           else{
                               var QuestionChoiceItem = SFormBDQuestionChoiceList.getAll();
                               console.log(QuestionChoiceItem);
                               for(var l=0;l<QuestionChoiceItem.length;l++) {
                                   var itemCL = QuestionChoiceItem[l];
                                   console.log(itemCL);
                                   console.log(formRow.slice(-1));
                                   if(itemCL.QuestionChoiceListID == formRow.slice(-1)) {
                                       var QuestionChoiceL = itemCL.QuestionChoice
                                       console.log(QuestionChoiceL);
                                   }
                               }
                               if (formRow.length > 4) {    // new question text
                                   question = {
                                       "QuestionID":formRow[0],
                                       "type":"radio",
                                       "textID":formRow[2],
                                       "text":formRow[3],
                                       "choiceListID":formRow[4],
                                       "choices":  QuestionChoiceL
                                   }

                               }
                               else { //old question text
                                   question = {
                                       "QuestionID":formRow[0],
                                       "type":"radio",
                                       "textID":formRow[2],
                                       "text":QuestionText,
                                       "choiceListID":formRow[3],
                                       "choices":  QuestionChoiceL
                                   }
                               }
                           }
                       }
                   SFormBDQuestion.add(question); //input question into SFormBD base data
                   }
                   else{ //old question
                       console.log("it is an old question");
                       question =  {
                           "QuestionID":formRow[0]
                       }
                   }
               formInputperRow.push(question);
               }
           }
           else {
               console.log("it is an old form");
               StorageService.add({"$id":formID,"title":formTitle,"lat":longlat[1],"lng":longlat[0]});
           }

           StorageService.add({"$id":formID,"title":formTitle,"lat":longlat[1],"lng":longlat[0],"form":formInputperRow
                   });
           console.log(formInputperRow);
           /*items.push({"$id":res[3],"title":res[4],"lat":res[7],"lng":res[6],"form":formInputperRow
           });*/

           console.log(StorageService.getAll());
           //console.log(items);

       }])
```