RainCast: A Voice Based Weather Service for Rural Ghana

Hameedat Omoine, Hui Chen, M. Iqbal Akhtar

VU University Amsterdam, 1081 HV Amsterdam, The Netherlands,
h.o.omoinechogudo@student.vu.nl, h2.chen@student.vu.nl,
iqbalgcuf@gmail.com

Abstract. In the absence of access to weather forecast data in the rural Northern Ghana Region, farmers are unable to make sufficient plans to save their crops from bad weather conditions like storms, floods, etc. With access to weather forecasts at low cost, through existing infrastructure, farmers can get weather predictions and be better prepared for upcoming weather conditions. In this paper, we have developed a prototype for a voice-based weather forecast application that makes use of the current GSM and network infrastructure available in the rural region. The application is able to get weather forecast data through APIs provided by an online weather forecast service, and make this data available to users when they call the app. Our research shows that a voice-based weather application can be built using low cost tools and infrastructure, and users with a single phone call, can get real time weather data using simple GSM phones (non-smart phones) with mobile network connection.

1 Introduction

In many rural areas of Africa like Northern Ghana, farming is still highly dependent on the weather condition, for example the gross rain volume at a certain time of the year. However, farmers at these rural areas don't have any reliable and instant source of weather forecast. The most widely used rain forecast information source is the radio broadcast. Although this information is easy to get from radio, it lacks the accuracy for small locations and at some point the weather forecast is only occasionally mentioned in a radio broadcasts. Therefore, for farmers weather forecast by radio is not an effective and reliable source because it does not meet their needs and expectations. In order to improve the productivity and reduce the damage from unprecedented weather changes, farmers need to know the accurate rain forecast for both coming few hours and coming few days. In some cases, intensity of rain for example light shower or heavy rain etc is also important.

For the purpose of helping farmers in rural Ghana villages to improve the farming productivity, an application will be developed to assist them get rain forecast and weather information for as far back as the next 5 days. Which they can get at anytime using their existing cell phones. In terms of technology, the people in these areas have little to no access to smart-phones or internet.

This also makes it hard to impossible for them to get weather forecasts which are widely available online. In order to overcome this challenge, the app to be developed will mainly be a voice-based app. This app would be able to make use of the existing technologies currently being used in these areas, like the GSM phones that have no 'smartphone' functions. The farmers and even non-farmers will be able to get weather information for their location using a very easy to use voice service interface.

By developing this application, farmers in Northern Ghana would be able to make better decisions and improve agricultural planning, that would enhance their preparedness for various weather changes, leading to reduced crop loss, better agricultural productivity and more work satisfaction.

2 Use Case Description

2.1 Name

The application to be developed named "Raincast" will be used by farmers to get accurate predictions of rain in coming five days. This application will provide a voice based service via cell phone and we also design web based interface that is directly connected to weather Service . Currently this app is able to give weather predictions for only two regions in Northern Ghana, (Tamale and Bolgatanga). However, its is possible to add as many cities as needed. With this app, the farmers in Northern Ghana will be able to check and listen to the rain forecast as well as general weather condition reports for example Tuesday Morning rain predicted "short but intense" or "long but shabby", and temperature "is currently 40 degrees Celsius, very hot" or "is currently 20 degrees celsius, warm", etc.

2.2 Summary of Key Idea

In many rural regions in Africa, local radio and mobile phones are the most important source of information and communication media. Many people have simple and basic models of mobile phones used for chatting mostly through text-messages. The wide availability of these cell phones gives us the opportunity to make use of them in benefiting the local community. The idea is to assist farmers in deciding what action to take according to the weather of the coming days in Bolgatanga and Tamale. Farmers could make a call at any time to a local number that will be picked up by the system and through some simple interactive steps, the farmers can listen to the rain forecast. This forecasts would be retrieved from a top rated weather forecasts provider, which would be made available to the voice application through an API. This on-demand service can be useful to many other businesses that could be dependent on the weather e.g. transportation, tourism, etc.

2.3 Actors and Goals

In this Application three categories of Actors with different goals are involved. These are described in the table 1.

Table 1. Actors and Goals.

Actors	Goals
Farmers and other users depending on weather information	Get reliable weather forecasts in order to be able to make necessary agricultural preparations to protect their crops and improve productivity.
System owner	Provide a working system that is easy to use and understand. Collect data from weather forecast provider and convert into verbal message to provides accurate weather information to farmers.
Weather forecast provider	Provide the system with accurate weather forecast information for the Northern Ghana region.

2.4 Context and Scope

Agriculture in Bolgatanga and Tamale, in Northern Ghana is mainly rain-fed. The cropping and harvesting is done during the rainy season. Meteorological information is very important for farmers in Ghana. By using this information farmers can prepare themselves for upcoming weather by taking necessary preparations and make their land ready for sowing, etc. Weather forecasts that include only temperature detail is sometime mentioned in a local radio broadcast. But farmers are not much interested in the prospect of just temperature. Farmers want to know:

- Rain forecast for the next few days in Bolgatanga and Tamale.
- If it will rain or not, and intensity of rain (heavy or light).
- General weather conditions

The three stakeholders related to this application are thereby:

Farmers: They want to know the Rain forecast in detail for next few days.

Weather Forecast Provider: They will provide Meteorological information to the system.

System Provider: They will receive information from the weather forecast provider and make it available to the users/farmers.

2.5 Use Case Scenario Script

For end-users, such as farmers, using the application is straightforward.

1. The end-user calls the number for the service.

2. End-user selects a preferred language
3. End-user selects a desired location for which they wish to get weather information
4. End-user selects the type of weather information they wish to get.
5. The system sends a request to the weather forecast system using an API to get updated weather information
6. The system gets a feedback and gives a verbal output to the user.
7. The end-user hears verbal weather information, and can either end the call, or request for additional weather information.

Figure 1. in the Appendix shows a call flow for the RainCast application.

2.6 Interaction and Communication

The end-user will call a local number that will be answered by the system and user will get two options such as for English press 1 and for a local language press 2. After language selection, system will respond in the selected language. Once user selects a language, user would be asked to select a location, either Tamale or Bolgatanga for instance. After selection, user is asked to select the type of weather information they are interested in, from a few options. After selecting an option, user will be able to listen to information on weather for the selected location. Figure 2 in the Appendix shows a high level user/system interactions and communications design, where primary user actions and system outputs are shown.

3 Contextual Issues

This application and its corresponding use-case scenario is limited to the farmers who have a mobile phone and their phone is connected to a mobile network. In addition, the system currently can provide English/French language interface. The call service will not be free due to operation costs pressure. The weather forecast information quality is dependent on the quality of information provided by the weather forecast service, therefore the accuracy can not be fully guaranteed.

The connection speed and the system response speed are the key performance measures of the system. The system should be able to stay connected even when the user is in an poor signal environment. That means the system should be active at all the time and communicates frequently with the weather data provider.

The system is built on top of the existing voice based service platform which is already in use in some African countries. The environmental dependencies of the system include network speed, power supply for the server and weather data connection stability. The biggest technical challenge of the system will be the concurrent calls from users. If we set concurrent call threshold too big, the costs of the system will be high but low concurrent threshold will drastically impact the user experience.

4 Theoretical Background

The Northern Region has a low population, of which the majority of live in villages with most of them being farmers. The official language of Ghana is English. Currently there is no reliable system for weather forecast. However, rain forecasts are very important and useful for farmers to plan their activities.

4.1 Introduction to North Ghana

The Northern Region is one of the ten regions of Ghana. It is the is the largest of the ten regions, covering an area of 70,384 square kilometers or 31 percent of Ghana's area . The Northern Region is divided into 26 districts. The region's capital is Tamale[1].

4.2 Bolgatanga and Tamale

Bolgatanga, colloquially known as Bolga, is a town and the capital of the Bolgatanga Municipal District and Upper East Region of north Ghana. Bolgatanga has a estimated population of 66,685 people. Bolgatanga is 161 km to the north of Tamale.

4.3 Weather

Climate of northern region of Ghana is much drier than southern part of Ghana. from January to March is the dry season and the rainy season is between July and December with an average annual rainfall of 750 to 1050mm. The highest temperature is reached in March and lowest in December and January. The temperatures of these areas are from 14 degrees to 40 degrees. The wet season of the year is the best season for Farming, Because in most areas farming only depend on rain water. If annual average rainfall is below 750mm that's mean crops are in less quantity and of poor quality. Due to this reason, farmers want to get accurate weather predictions for days ahead, in order to give them time to prepare.

5 Relevant Literature

ICT4D projects in developing countries, especially in resource-limited contexts, frequently face infrastructure challenges such as unstable power supply, low internet connection speed and low performance hardware. Therefore, a technically and economically sustainable development model is required. Based on multiple technical and non-technical requirements, Victor de Boer et. al. [6] proposed KasaDaka, a rapid prototyping platform for the rural poor. KasaDaka enables local developers to choose various hardware and software combinations and develop voice based services. It works well in low cost hardware and GSM network environments, and makes various service implementation feasible.

The RainCast implementation was mainly carried out using a Django based VSDK which was introduced by Baart [2]. The VSDK provides an easy to use GUI so that even developers with less programming knowledge can build voice services. Before the development of VSDK, developers who wanted to implement voice services on KasaDaka platform must have extensive Python, VoiceXML knowledge. However, in the most needed developing countries, developers with good knowledge of programming skills are sparse. As a result, the voice services development are expensive and inefficient.

6 Solution Design

The RainCast prototype was designed to cover a series of functionalities that will help deliver a simple application that provides the locals with weather forecasts. These functionalities have been highlighted in this section, a user-system interactions model can be found in Figure 2. of the Appendix.

6.1 Select Language

The first functionality implemented for the app is the language selection functionality. The prototype has been designed to be able to interact with the user in English or French. Upon calling the app, users are prompted to select either English or French, then the system goes ahead and interacts with the user in the selected language. The main idea behind including English as a language option is because it is the official language being used in Ghana. Which means that a lot of people in Ghana will be able to interact with the application by select English as a preferred language.

However, in the context of this research, the app is mostly to be used by local farmers in Ghana, more specifically in the Northern part of Ghana. Most of these farmers are uneducated and only speak or understand their local languages. Meaning, most of them will not be able to interact with the system in English. With this idea in mind, we decided that a local language option needs to be included in order to better fit the context. However, there was a challenge to implementing a local language. Which was that there were no available resources or tools for translating the system's content to a local language within the time given. The French language was thereby implemented as it was possible to get translations in French using the 'Google Translate' tool. It was also important to include the French language option because it serves as a way of demonstrating how a non-English language option can be implemented when building this type of application. It helps to show that once resources for translating to a local languages are made available, new local languages can easily be added to the application.

6.2 Select Location

The current prototype has been designed to provide weather information and forecasts for two locations; Bolgatanga and Tamale. Users are given the option

to select either of these location, then the system is able to respond with weather data for the selected location.

The first prototype used just a single location of Bolgatanga, in accordance to the specifications in the use case description. However, based on the feedback given, the final prototype has included a second location, and the location selection functionality. The second location Tamale is included in the this version in order to show that additional locations can be implemented in the application.

6.3 Select Weather Option

The current prototype has been designed to give weather information in two categories:

- Current Weather Information,
- Five days of weather forecasts.

Once user selects a location, user is asked to select to either get information for the 'Current Weather' or information on 'Five days forecast'. The 'Current Weather' option gives user information on the current weather, in terms of the current temperature in degree Celsius, and the weather condition which is provided as either 'heavy rain', 'clear skies', 'cloudy', 'thunderstorm', 'windy', etc. By this, user is able to get the current temperature and the current weather conditions for the selected location.

The second option of five days forecast was implemented based on the weather API available. The Open Weather Map API was used for getting weather information, and we were able to get a maximum of five days of weather forecasts for all major cities in Ghana. Therefore, the application was designed to provide users with five days of weather forecasts.

Based on the feedback received during the demo market, we were made to consider that farmers are not really interested in numbers such as temperature, and are more interested in things such as "there will be rain", "there will not be rain", "it will be cloudy and cold", etc. This prompted the decision to include only weather condition information in the forecasts, and exclude the temperature. The weather condition information will let the users know whether there will be rain or not, as well as the intensity of it (heavy, light, or drizzle). For each day, user can get the foretasted weather condition for 'Morning', 'Afternoon', and 'Evening'. We made use of these descriptions rather than an exact timestamps because the sense of time for the local farmers will probably be more adjusted to these three descriptions, rather than saying 6am, 9pm, etc.

Users will be able to navigate through the app to get information on all five days forecasts starting from the present date. Which means that if a user calls the app in the morning on a Saturday, the user will be able to get the forecast for Saturday afternoon and evening, then Sunday morning, afternoon, and evening, then Monday, Tuesday, and so on. If a user calls back every three hours, the user will be able to get additional forecasts. More on the forecasts data retrieval will be discussed in the next

6.4 Retrieve Weather Informations

An important component of the raincast application is the data source that provides it with weather data. In order to provide the users with real time weather information, the raincast app is designed to get its weather data from Open Weather Map. Open Weather Map is a service that provides weather data. The service provides an API that makes it possible to extract weather data and link with the application.

The OpenWeatherMap service was selected amongst others because it provides weather information and forecasts for all cities in Ghana. And it also provides forecast for as much as 5 days ahead for free, and as much as 16 days for a paid account. Raincast makes use of two separate APIs provided by OpenWeatherMap. One provides current weather data, while the other provides forecasts for 5 days for every 3-hours of the day. The app is designed to extract weather data from both APIs, and make it available to the users through the voice interface. From the API, the app retrieves the temperature, weather condition, day of the week, and time of the forecast. The current weather provides the temperature and weather condition. While the 5 days forecast provides the date of forecast, the time of day (morning, afternoon, or night), and the weather condition for that day. The forecasts are provided for every three hours rather than hourly, for example, a forecast is given for 6am, then next will be 9am, 12pm, 3pm, etc. Therefore, the app is able to continuously provide updated weather forecasts every 3 hours.

7 Implementation and Usage Scenario

This section will discuss how each functionality has been implemented in the development of our prototype. The current link for pointing to our app is: http://raincastapp.herokuapp.com/vxml on the Kasadaka VXML Switcher, and the web interface: https://raincastapp.herokuapp.com.

7.1 General

The final version of the prototype is built using the python Flask framework. This version deviates from the previous prototype that was built using the Django VSDK. The decision to use Flask stems from the fact that the Django VSDK platform did not support the API use and did not give a lot of freedom in terms of adding functionalities. Also, in terms of development skills, we had more experience using the Flask framework in developing python applications, than with Django. This made it easier to rebuild the prototype from scratch and implement all desired functionalities, including the use of the OpenWeatherMap API.

A simple web interface was also built for the application. Which makes use of JavaScript in retrieving weather data from the API and displaying it on the web interface.

7.2 Audio Files

All output provided by the application have been pre-recorded, named and saved as wav audio files. Taking into consideration that the Kasadaka platform does not support text to speech, all audio files that will be needed by the app have been pre-recorded and saved in the format; bit-resolution: 16 bit, sampling rate: 8000Hz, audio channel: mono, and PCM format: PCM signed 16-bit little-ending. The format is saved as required to in order to be compatible with Kasadaka.

7.3 Weather API

In order to get real-time weather data to be used in the application, we needed to connect the application to the to OpenWeatherMap API, and be able to extract required data and output them to the users. In getting this weather data from the API, we made use of python, and stored it in the JSON format so it can be used in the application. Therefore, each time a user calls our application, the application calls the API, gets the weather data, and outputs the response to the user.

When a user calls the application, they request two types of data: temperature, and weather condition. However, in order for the app to be functional on Kasadaka, the app has to respond to the request using pre-recorded audio files. Which means that all possible responses from the API have to be pre-recorded and stored as audio wav files, then extracted and played to the user when needed. For example, when a user calls the application and wishes to get the current weather condition, the response from the API could be 'heavy rainfall with wind' in text format. However, since Kasadaka does not support text-to-speech, this response has to be stored as an audio file, and played to the user. The following data were extracted from the API and have been pre-record and stored as wav files to be used by the application.

Weather Condition The OpenWeatherMap API is designed to store up to 41 different types of weather conditions using unique IDs. For example one weather condition could be 'clear skies' and the unique ID is 800, another weather condition is 'heavy intensity rainfall' with unique ID 500, 'thunderstorm with heavy rain' with unique ID 505, etc. Depending on the current or the forecasted weather condition, the API responds with one of these 41 types of weather conditions by providing the unique ID, and the text description. Hence, the idea for including these weather conditions in the app, was to record all 41 weather condition types, and save them using the unique ID provided by the API. Which means that an audio recording of 'the weather condition shows clear skies' is stored as '800.wav', and an audio recording of 'the weather condition shows heavy intensity rainfall' is stored as '500.wav'.

The next step was to retrieve only the unique ID for each weather condition from the API response, then parse it into the VXML template using python, so that the app can respond with a correlating audio file each time. For example, if the API responds with the weather condition ID '800', the app checks the static

folder and retrieves the audio file '800.wav' and plays it to the user. A example of how the code was implement is as shown below:

```
<prompt>
  <audio src="static/{{context.city_weather.id}}.wav"/>
</prompt>
```

From the code above, context city weather id will be equals to the weather condition ID. Therefore, the app will play the corresponding audio file.

Temperature For the current weather option, the prototype application provides the current temperature in degree Celsius. The API provides the weather in degrees Celsius. Then in order to make it possible to read out this temperature to the user via the voice interface, the temperature have to be rounded up or down to two digits. We took into consideration in our designs that the temperature in degree Celsius will always be between the 0 - 99 degrees Celsius in the Ghana region which is in Africa. The weather in Africa which is usually hot has never been recorded to go lower than 0 degrees, and the weather all over the world has never been recorded to go higher than 60 degrees Celsius. So this prompted the decision to use degree Celsius, round it to 2 digits, then split it into 2 separate digits. Then audio files were then recorded for the numbers 0 - 9, and stored as wav files. This means that instead of creating audio files from 0 - 60, we just needed to create for 0 -9, then pair each individual digits up to make two digits that represents the temperature. To give an example, if the API provides the weather temperature as 35.7 degrees Celsius, this value is then rounded up to 36, then split into 3 and 6, as the first and second digit respectively. These two digits are then passed into the VXML code using python, and the app plays the pre-recorded audio files for '3.wav', and '6.wav', with additional audio files that read 'the current temperature is', and degrees Celsius So that when user listens to the temperature using the voice app, user hears 'The current temperature is three six degrees Celsius.' A sample of the code is as shown below:

```
<prompt>
  <audio src="static/currenttemp.wav"/>
  <audio src="static/{{context.city_weather.fdigit}}.wav"/>
  <audio src="static/{{context.city_weather.sdigit}}.wav"/>
  <audio src="static/degreecelsius.wav"/>
</prompt>
```

From the code snippet above, context city weather digit represents the first digit, and context city weather digit' represents the second digit. Using this method, we have been able to implement a functionality that allows the voice application read out temperature values from the API. However, based on the feedback we got from the demo market, we were made to take note that values such as temperature are not so important to local farmers. This is why we excluded this detail in the forecast option. But we have left in the current weather option just to demonstrate a way that this functionality can be included in a voice-based application when needed.

Day of the Week and Time of Day The next thing that was implemented in the app is the the day of the week, and time of the day for the weather forecasts. The users need to know what day of the week the forecast is for (Monday, Tuesday, Wednesday, etc.), and what time of the day, either 'Morning', 'Afternoon', or 'Evening'.

The weather API however only provides the date and time in a Unix timestamps format, and in a numerical date and time format. Therefore these data had to be converted into the day of the week format, and the time categorized into morning, afternoon or evening. Therefore, each time the API provides a forecast with the unix timestamp is retrieved, converted, then stored as day of the week (Monday - Sunday), and time of the day('morn', 'noon', and 'even'). And in order to be able to output this via the voice interface, audio files were recorded for each day of the week: Monday to Friday, and for time of the day: morning, afternoon, and night, and stored accordingly. Therefore when a call is made to the API, and for instance, the date is 28th of May, 5PM, the app will store day of the week as Monday, and time of day as Evening. Then when this is parsed into the VXML code using python, the corresponding Monday.wav and Evening.wav audio files will be retrieved and played to the user. User will hear: 'For Monday Evening.' followed by the weather condition.

7.4 Detailed Usage Scenario

Figure 3 in the Appendix shows the call flow for the application. A more detailed usage scenario script that runs through one complete branch of the application is outlined below:

1. User calls the application and is first asked to select a preferred language:
   RainCast: For English, Press 1. For French, Press 2.
   User: (presses 1 for English)
2. User listens to welcome message and is asked to select a location:
   RainCast: Welcome to RainCast. We provide you with accurate weather forecast. Please select your location.
   RainCast: For Tamale, Press 1. For Bolgatanga, Press 2.
   User: (presses 1 for Tamale and app goes to http://raincastapp.herokuapp.com/tamale)
3. User is asked to select weather type he/she is interested in:
   RainCast: All forecasts are brought to you by OpenWeatherMap. To get information on the current weather, Press 1. To get the weather forecasts for today and the next four days, Press 2.
   User: (presses 1 for current weather information)
   RainCast: The current weather is cloudy with heavy rainfall. With temperature at three..five..degrees celsius.
4. User is asked if he/she wishes to get more information:
   RainCast: Do you wish to get additional weather information? For 'Yes', Press 1. For 'No', Press 2.
   User: (presses 1 for 'Yes')

RainCast: To get information on the current weather, Press 1. To get the weather forecasts for today and the next four days, Press 2.
User: (presses 2 for 'weather forecasts')

5. Application retrieves and reads out weather information to user:
RainCast: For Monday afternoon, the weather is cloudy with some light rainfall. For Monday evening, the weather is cloudy with no rain. For Tuesday afternoon, the weather shows clear skies.
RainCast: To get more forecasts, Press 1. To quit, Press 2.
User: (presses 2, to quit)
RainCast: Thank you for using our service. Goodbye. (Application ends call).

The above script shows one of the flows that can be found within the app. In the forecasts part, if the user keeps listening to more forecasts and reaches the last one, user is notified that they have reached the last forecast. And that they can call back in three hours or more to get more updated forecasts, before the call is terminated.

7.5 Implementation, Hosting and Porting on Kasadaka

By having access to the application folder, our prototype can easily be setup and running on any local or web server.

The tools that will be needed for implementing the application are:

- Git
- Python 3.6 and above
- Pip
- Heroku Account

these need to be setup and, or installed, and some knowledge on the use of them and the Command Line is required. In addition, a video demontrating the following steps to set up the application on heroku and the Kasadaka based server can be found here 'https://www.youtube.com/watch? v=6I2lNHA5FAg'

Getting the Code As shown in the video, our project code can be found at https://github.com/armydah30/raincastprototype. Navigate to the 'Clone or Download' button, and copy the repository URL.

Setup : The next thing to be done is creating a folder for the application and cloning the repository. This can be done by opening Command Line and creating a folder for the application. Navigate into the folder directory as shown in the video. Then type the command 'git clone https://github.com/armydah30/ raincastprototype.git'. This clones the application into the directory. Next, change directory into the application folder by typing 'cd raincastprototype', before going to the next step.

Hosting on Heroku Once an heroku account has been created, the next step is to log in via command line. Next step is to type in 'heroku login'. Type in heroku email and password as prompted. Once logged in, type 'heroku create' to create a new application. Next, type 'git add .' to make sure all files are selected. Then type 'git commit -a -m "first commit" '. Then type 'git push heroku master'. This will push the application to the heroku server and have it hosted there. Once the application has been created and launched, copy the URL as shown in the video.

Porting on Kasadaka Based Server Now that the application has been hosted on Heroku, the link can be added on the Kasadaka VXML Switcher, and used to point to the application so it can be called. If the link for the created heroku app is 'http://raincastapp.herokuapp.com', add '/vxml' behind it. Making the link for the app: 'http://raincastapp.herokuapp.com/vxml' as shown in the video. The Kasadaka should be set to point to this link, and all calls will be directed to the application. With this, the application can be called and tested. A sample call of the application can be found at 'https://www.youtube.com/watch?v=GoKIj2BCZvc'

Hosting on Local Apart from heroku, one can also view the application locally, however, the voice application will not work locally, only the web interface will be available. In order to setup the local system to have all the requirements to run the application, type 'pip install -r requirements.txt' on command line to install all requirements for the application. Or typing 'pip install flask', and 'pip install requests'. Next, open the 'app.py' file in a desired text editor (e.g Notepad). In line 11, set debug to 'True' for testing purposes. Then read the instruction in line 2048, and make the changes as instructed, and save. Go back to command line, and type in 'python app.py' and press enter. Application should be up and running on the local server.

8 Scope and Fidelity of Prototype

Most of the proposed features of the application have been implemented. It is able to connect to an API and provide current and fore-casted weather information for up to two locations in Ghana. It is able to state the day of the week, and the time of the day for the forecast. Users can actually make use of the app and get real time predictions as if they were receiving the information via a web interface.

The only thing not included in the prototype is the local language option. Due to the lack of time, and access to tools that can provide local language translations, the local language option has not been included. However, if there was more time available, and we were able to get someone that is ready to create audio translations, then if will be possible to include a local language. All audio files can be created in the local language, and saved as wav files, then

implemented in the application, just as we did with the French language option. User will have the option to select the local language, and the system will interact with the user with the local language. If there is access to all audio recordings in the local language, the application can easily be extended to cover the language.

9 Deployment and Sustainability Plan

As mentioned in section 5, RainCast service is accessible through the KasaDaka platform which is already in use in several similar use cases at Africa. At the testing phase, the service was deployed to Heroku, an open source app hosting platform. In order to test if the service could work in the field, specifically, if the service could work on the KasaDaka-based server or not, we used an online server which has same software setup as offline Raspberry Pi. This server runs the Asterisk and VXI stack as it is used in the field. Because the server/Kasadaka does not run any (expensive, heavy, big languages only) TTS, all the speech output needed by the app were pre-recorded and in the .wav format. Before finalizing the report, the service was tested again on the server and it worked well.

At this stage, the RainCast is functionally ready to be deployed in the field without any further configuration or programming. To successfully deploy the service, following steps need to be done:

1. Hardware configuration. KasaDaka server needs to be able to receive calls from users and correspond speedily. At this step, help from KasaDaka specialist is needed. The ideal situation is at the target cities there are already several KasaDaka servers running and there are extra resource for our service.
2. Real-time weather information API configuration. So far we only used one API for testing. If we want to put our service into practice, a stable and high volume API is required. The API connection should be able to work in the low speed environment.
3. Users testing. To validate the service, all the possible scenarios should be tested. The best way is to invite real users to call the service at different level of concurrency.

The RainCast service is a light voice based service which consumes low to medium hardware and network resources depending on the expected concurrency calls.

The biggest financial challenge for deploying and maintaining the RainCast service comes from the hardware costs if new servers and base station is required. To build a brand new base station can be expensive. High costs will inevitably increase the service price. However in the rural communities at Ghana, high price equals to fewer users. Therefore, if the service is deployed in a non-KasaDaka area, funding is required for building a KasaDaka based service center. From the long run, early stage non-profit funding is also very sustainable for the ICT4D projects in the area.

10 Evaluation and Discussion

The current prototype has changed a lot from the initial from the previous versions. The current prototype is able to use real time data from an API, and has been built using the python flask framework. In evaluating the current prototype, one can have a look at its usability and understandability. These two attributes were considered in the development of the application. The call flows has been designed so they are easy to follow and so the instructions are also easy to understand. And the users are able to easily understand and navigate through the app. In terms of learn-ability, users can easily learn how to use the app as a result of the lack of complexity in the call flow. After using the app two to three times, users will be able to navigate through the app very easily. We allowed 'bargain' in the application because once users get familiar with the system, they should no longer have to wait to listen to all the instructions, they can just navigate quickly through the app, and get the information they want. This will allow users get information as quickly as possible, and will also cut down on the airtime and costs associated with it.

In addition, in terms on modifiability, and flexibility, the prototype can easily be extended to cover additional locations, and multiple languages based on the system design. However, with the current codes used in developing the prototype, it will be quite hard and time consuming to expand the application to cover multiple cities. We feel with higher software development skills, the prototype could be built in a way that allows it to be more flexible and scalable. This leads us to the main limitation we had in developing the prototype; level of skill in software development.

The main limitation in building the prototype was the level of skill in software development available. Although the current prototype is able to meet all functional requirements, we feel that it can still be developed further, given more time and more knowledge and skill in software development and Python. With more time and skills, the codes for the prototype can be cleaned up, and made more reusable, while making room for easy scalability and modifiability. More work will continue to be done in enhancing the prototype and expanding the scope of the application in terms on how many cities it covers, and how much forecasts it can provide. Future work on the application will involve adding multiple locations, creating a database that stores all forecasts, and providing users with more forecasts per day.

11 Conclusions and Future Work

By the end of this project, RainCast, a voice based weather forecast service for rural Ghana users, could be able to provide real-time weather information via two languages, English and French. RainCast has been built using the Flask framework to provide an easy to use weather service, especially rain forecasting for farmers who rely on this information to decide what to do to improve productivity. RainCast server side implementation uses Flask framework that

make future expansion and maintenance consistent for other developers. On the other hand, using popular developing framework makes our application reusable in other similar use cases. Considering limited number of developers working in rural Africa, RainCast provides a user friendly web based user interface so that developers with less programming knowledge would be able to customize their own services.

RainCast is a light service, which means it could run on low spec hardware. Because in our target service area that server resources and internet connection are less abundant than developed society, low hardware and network dependencies enable RainCast service to be compatible in the local deployment. In the meantime, low resource requirements could in some extent reduce operational costs, which means more affordable and competitive service price. In other words, RainCast is a sustainable service which has high commercial feasibility to be applied in real business practices.

However, RainCast is yet to be tested by real target users. Weather the voice interface is easy to use or not and the usefulness of provided services should be evaluated by users in form of user survey or user meetings. Concurrency limit is another challenge which has not yet been tested. Local language support is critical to the acceptance of RainCast and expand language options should be considered as top priority in the future work.

References

1. Norther Region Ghana, $https://en.wikipedia.org/wiki/Northern_Region_(Ghana)$
2. André, B. (2016). KasaDaka: a sustainable voice-service platform. $https://w4ra.org/wp-content/uploads/2017/12/Master-thesis-andre-baart_Dec2017.pdf$
3. Kalanda-Joshua, M., Ngongondo, C., Chipeta, L., Mpembeka, F. (2011). Integrating indigenous knowledge with conventional science: Enhancing localised climate and weather forecasts in Nessa, Mulanje, Malawi. Physics and Chemistry of the Earth, Parts A/B/C, 36(14-15), 996-1003.
4. Asenso-Okyere, K., Mekonnen, D. A. (2012). The importance of ICTs in the provision of information for improving agricultural productivity and rural incomes in Africa. African Human Development Report. UNDP Sponsored research Seri
5. Yeboah, S. (2017, July 4). How a lack of access to reliable weather data is hurting African farmers. Retrieved February 06, 2018, from https://theconversation.com/how-a-lack-of-access-to-reliable-weather-data-is-hurting-afr Ican-farmers-80011
6. Lo, A. G., Schlobach, K. S., de Boer, V. (2016). Kasadaka: a rapid prototyping platform for the rural poor. Abstract from ICT.OPEN 2016, Amersfoort, Netherlands.

A Appendix

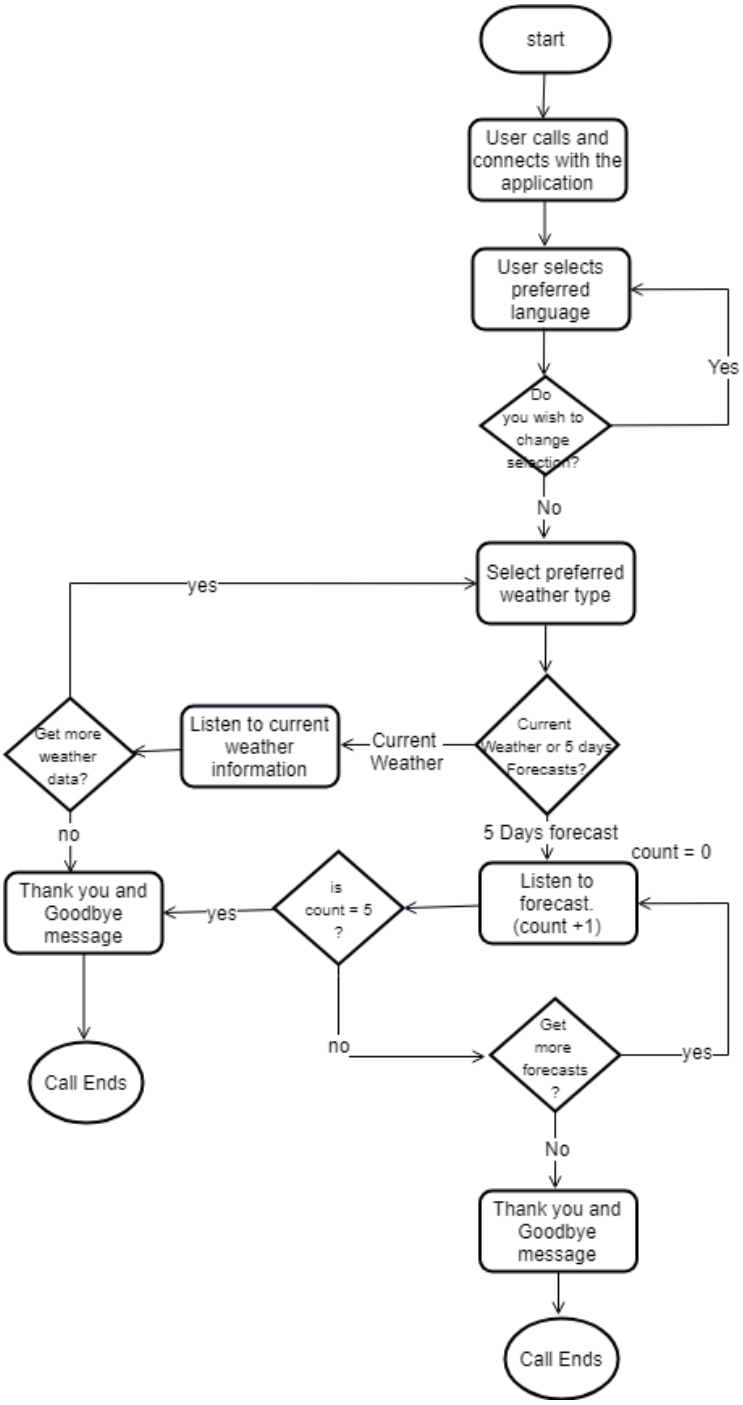

Fig. 1. System Call Flow

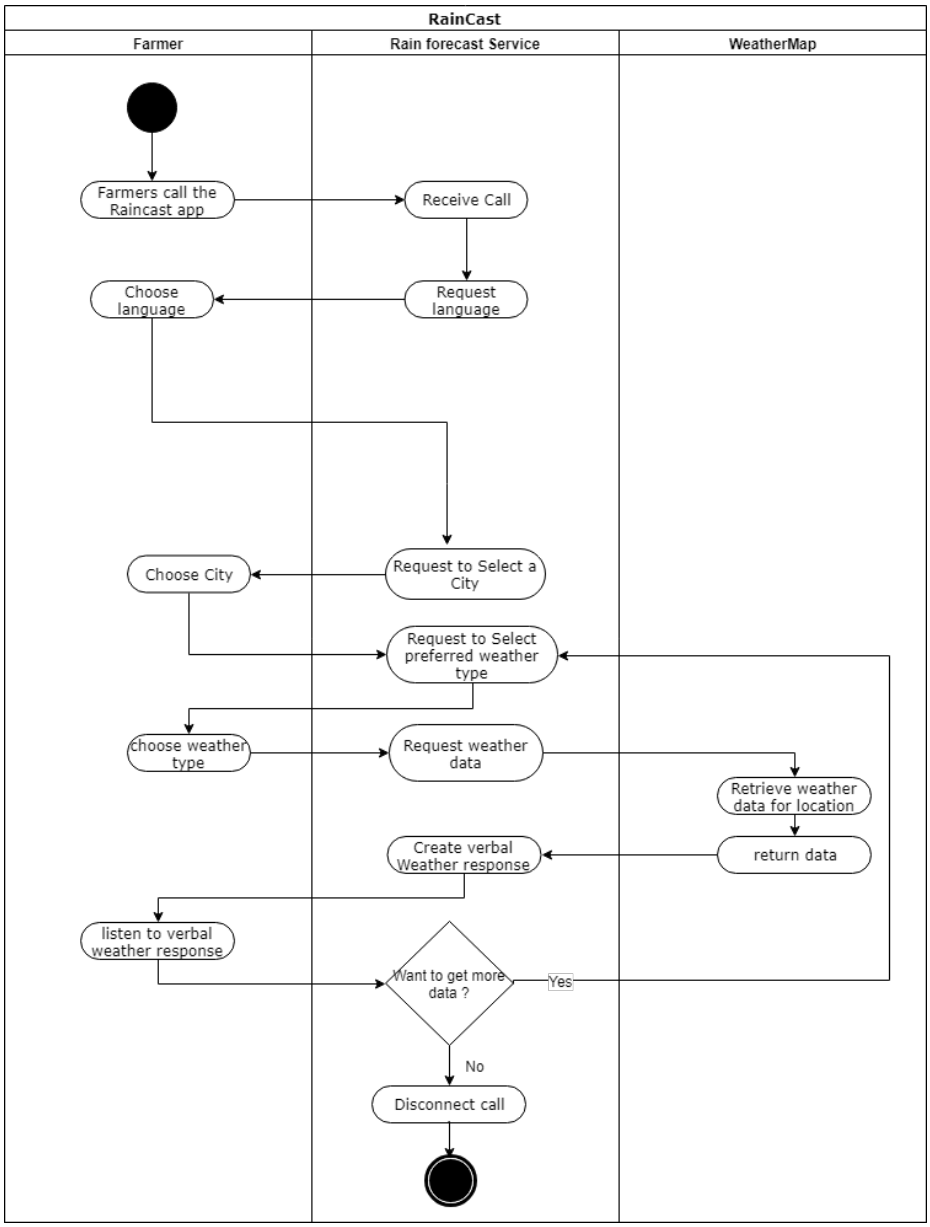

Fig. 2. RainCast Activity Diagram

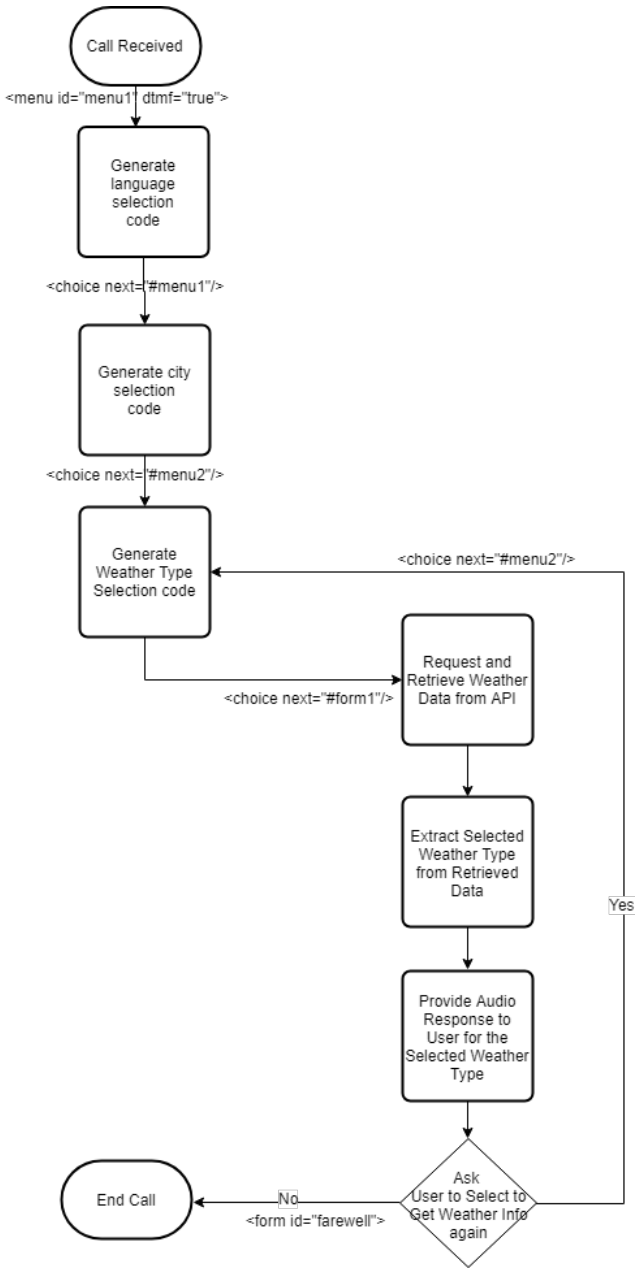

Fig. 3. RainCast Activity Diagram