Requirements Analysis and Project Plan

Our Brief: Low-income countries often have network coverage in rural areas, but the devices and data contracts available to subsistence farmers make voice and SMS more practical then web services. You need to design a subscriber service that provides automated weather, health and market updates which users can personalise to their needs from a basic feature phone. You may need to use SMS gateways, off the shelf Interactive Voice Response techniques, and public APIs for information sources. Keep in mind the need for security, accessibility to low literacy users, and simultaneous support for a large number of local languages

General Investigation

We have chosen Twilio as our main API / platform for SMS & IVR functionality.

Assumptions

We chose to implement the service in one country - the Republic of Uganda - to allow us to really focus on gathering correct data, and to demonstrate how it would function.

Republic of Uganda

- Capital: Kampala
- Population 35.6 million
- Area 241,038 sq km (93,072 sq miles)
- Languages English (official), Swahili (official), Luganda, various Bantu and Nilotic languages
- Major religions Christianity, Islam
- Life expectancy 54 years (men), 55 years (women)
- Currency Ugandan shilling

We chose the Republic of Uganda because it is a country which relies heavily on subsistence farming, and one in which this service would be useful to a fairly large proportion of the population. We were able to find comprehensive data to support all two parts of the service (weather and markets), making it a suitable country to implement a first version of the service for. Furthermore, both of the major languages spoken (English and Swahili) are supported by translation APIs, facilitating the support for multiple local dialects.

Once we have our system built, it can be easily rolled out into other countries / continents. Apart from data sources providing market information, which may pertain to specific countries or regions, the translation API, data source for weather, as well as the website that delivers custom messages, are not country-specific. We plan to develop a multilingual setup interface and initial greetings that automatically change language based on the user's country-code/location, prioritising genericity and covering as many languages as we can support. Once the user chooses a preferred language, his or her preferences would be

saved in our database and all future communications will be based on that language. We envision that daily weather updates should work generally for users across the globe, while availability of health/market-related functionality can be rolled out across more regions as we incorporate additional region-specific data sources.

Assumptions Relating to Data Available

In terms of weather data, the weather APIs we will use have support for locations globally, so choosing a specific country here was not that important. However, for market data, the API we have found has good support for crops which would be of interest to subsistence farmers (such as maize and rice).

Assumptions Relating to Multilingual Support

We have chosen to integrate off-the-shelf translation APIs as our main method for providing regional / multilingual support. It is unreasonable to support languages and regions not yet covered by mainstream phone manufacturers / operating systems, hence we deem the coverage provided by Twilio & Google Translate sufficient. This means that our coverage will not extend beyond the major lingua francas of the world, and will exclude esoteric / minor regional languages. This corresponds to the assumption that phone users do have the level of literacy required to text and read in at least one of the major lingua francas of their region.

We are aware of the common scenario where people prefer typing and speaking in different languages, so we plan to design our system to incorporate such bilingualism.

Assumptions Relating to Mobile Device

We assume that the users have a phone of some kind with which they can call a local number and talk into the phone, as well have the ability to enter decimal (English) numbers - which prevail across all phones around the world. This would allow them to work their way through an IVR. Additionally, we have an expectation that the user has the ability to send messages through texting - whether on a flip phone or on a smartphone. This may be in local scripts or in a roman alphabet, since some newer SMS standards support unicode texting.

Assumptions Relating to Location Information

Getting exact locations is a crucial aspect of our system but is a significant challenge. The prefered input method will be using what3words, giving memorable versions of lat-long coordinates, which could be distributed as the phone number to our service is distributed. The second, less favoured input method will be to allow them to enter the name of the nearest significant settlement/town, to get a rough position using the Google Maps API.

Investigation into Data Sources

The first step was to undergo research into finding out what data would be relevant, and is readily available for us to use.

Weather
Upon doing some research into the most valuable weather information to Farmers, we concluded that data the API would need to contain is:

- Minimum / Maximum temperature of the day
- General weather (sunny, showers, snow, etc.)
- Precipitation rate
- Warning of hail, frost, storms
- Other desirable information, but not essential, includes cloud coverage, humidity, wind direction and pressure.

There are a vast majority of Weather APIs, all very well documented. So we narrowed it down to 2 potential APIs; Openweathermap and Darksky.net. Both provide all the functionality we would need and seem to work fine for obscure towns entered.
Both also provide multiple language support across a large range languages, which may be useful as multilingual support is a requirement.
We choose to use Darksky.net in the end since the format, and amount of data an API call retrieves works a lot better with what our system needs.

Markets
Before deciding which market prices to support, it is important to know what are the main farmed crops across the word, according to Business Insider, the 10 most important crops in the world are:
1. Corn (large in Sub-Saharan Africa, South America)
2. Wheat
3. Rice
4. Potatoes (Europe, China is largest producer world wide)
5. Cassava (Africa, South America)
6. Soybeans
7. Sweet Potatoes (South America, majorly produced by China)
8. Sorghum
9. Yams (West and Central Africa)
10. Plantain

It proved difficult to find data for market prices, especially in low-income countries. Many data sources exist, such as price graphs of common commodities across many countries, however there is no way to easily export this data, and we do not know how often it is updated.

The one API we managed to find for crops in a wide variety of (non western) countries was Food Security Portal. It provides monthly data updates with the prices for Maize, Rice and Wheat for countries such as Brazil, DR Congo, Haiti, Nigeria and Uganda.

Crop Health
We failed to find any data sources that provide even remotely up to date information about crop health, aside from some yearly datasets. Instead, we have decided to implement a system for users to report issues with their crops, to then be analysed by some NGO or

government worker. To send out any advice the analysts may have, we would need to provide an interface for them to broadcast messages across our system.

Functionality provided

The main goal of the system is to provide informative messages, which will be messages containing weather and market information relevant to the user's location. In this specification we will call these messages 'update messages'. These messages will be sent in whichever language is prefered by the users, from a set of Lingua Francas supported by the Google Translate API. The users will be able to decide which crops they are interested in, so that they aren't sent unwanted information.

To enter these preferences, there will be a configuration system which will collect this information when a user first registers. The messages sent by this system will be referred to as 'configuration messages'.

The update messages will be sent out daily, but the user will be able to prompt the system to send a more up to date message, by ringing or SMS messaging the system. In addition to the automated collection of data to produce messages, there will also be a web interface for local NGO workers or government agencies to send messages to specific areas.

Users will also be able to ring our system to report issues they are having with their crops. They will simply say what is wrong, and this will be recorded as an audio file to be sent to the web interface. This interface will have a page which displays a map with pins in it to show issues that have been reported, and clicking on these pins will play the audio recorded when a user reported their problem. Using this interface a NGO/government worker can decide to send advice if they believe there is an outbreak of some pest in that region.

Overview of major components

- Data Gathering
    - Collects weather and market data
- User Database
    - Stores and updates user preferences
    - Integrates multilingual support across Twilio, Google Translate, and message-generation server
- Message Generation
    - Packages messages and alerts into text
    - Queries the data gathering system
- SMS Sending & Text Translation
    - Uses Google Translate and Twilio to translate and deliver messages
- IVR & Voice Translation
    - Handles phone calls using Twilio's multilingual text-to-voice
- SMS Receiving
    - Delivers messages from Twilio to configuration system where appropriate
    - Prompts message generator when appropriate
- Voice Receiving
    - Reads out messages from the generator
    - Delivers crop reports from Twilio to web server
- Registration Protocol
    - Handles the setup required for a new user, as well as the need to update / reset preferences
- Crop Report Database
    - Accepts and contains recordings of crop health reports, providing them to the website.
- Web-based Interface
    - Provides a way for authenticated sources to deliver custom broadcast messages to specific regions at specific times
    - Provides a way for authenticated sources to access crop reports

Detailed specifications of components and interfaces

- Data Gathering
    - General notes on its functionality
        - Gathers data from weather/market/health APIs/RSS feeds
            - Weather: OpenWeatherMap, Dark Sky
            - Market: Food Security Portal
            - Data sources should be queried once a day for each timezone, before update-messages are generated and sent.
    - Interfaces provided
        - Data provider
            - Arguments
                - User's information and preferences (e.g. for crops)

                - Returns
                    - Relevant health/weather/market data
- User Database
    - General notes on its functionality
        - Apart from the user database recording user data and preferences, there is also a self-updating database/routine that provides languages we currently support (text and voice) for a country/region
            - I.e. “what languages should we loop through in our IVR setup when user is from ‘+1’ -- US? Respond: English and Spanish, both of which are supported by Google Translate and Twilio…”
    - Interfaces provided
        - Single-user lookup functions
            - Arguments
                - Phone number, and the field in question
            - Returns
                - Relevant field, for example:
                    - If the user is registered in the system
                    - The user’s location (a precise version, and an approximate version?)
                    - Language preference (can be different for the one used by SMS and the one used by IVR)
                    - Crop preference
                    - Timezone
        - User-group querier
            - Arguments
                - Location / country / crop preference / timezone
            - Returns
                - A list of corresponding users sharing that property
        - Single-user update functions
            - Arguments
                - Phone number, and relevant field
            - Returns
                - Whether if the update (or addition, if user is not already registered) is successful
        - Languages of a country
            - Arguments
                - Country (code?)
            - Returns
                - A list of (supported) languages
- Message Generation
    - General notes on its functionality
        - Generates periodically delivered status messages for all users present in the system.
        - When a user phones-in or texts-in and prompts the system for an update, immediately responds with information.
    - Interfaces expected

        - "Data provider" from "Data Gathering"
            - In particular: should provide data when given a user location
        - "Single-user lookup" from "User Database"
        - "User-group querier" from "User Database"
            - In particular: should be able to return a list of all phone numbers currently registered
        - "Message sender" from "SMS Sending & Text Translation"
    - Interfaces provided
        - Custom message deliverer
            - Arguments
                - Message body generated by the "Web-based Update Interface", location to send to, and TTL parameters
            - Functionality
                - Records the custom message, and send it at the appropriate time to the appropriate user-group
        - Immediate responder
            - Arguments
                - Phone number, type of data requested?
            - Returns
                - An info message
- SMS Sending & Text Translation
    - General notes on its functionality
        - Use Google Translate for text-to-text translation
    - Interfaces provided
        - Message sender
            - Arguments
                - Target language, message body (in English?), target phone number
            - Returns
                - Whether if the SMS is successfully translated and sent (through Twilio)
- IVR & Voice Translation
    - General notes on its functionality
        - Works with the Twilio API to respond to incoming calls
            - If a phone number is not registered, invoke the Registration Protocol, and tell the user by voice to switch to SMS? (in a variety of languages)
            - If a phone number is registered, ask user (using the appropriate language) to dial: 1 if they want weather information, 2 if they want health information, 3 if they want market information and 4 if they want to modify their location or language
                - And provide the appropriate response
                - Invoking Registration Protocol in the case of "4"
    - Interfaces expected
        - "Single-user lookup" from "User Database"

        - “Languages of a country” from “User Database”
        - “Immediate responder” from “Message Generation”
            - In particular, should be able to get weather/health/market information messages separately
    - Interfaces provided
        - Country lookup
            - Arguments
                - Phone number
            - Returns
                - Corresponding country according to Twilio API
- SMS Receiving
    - General notes on its functionality
        - Deals with POST requests from Twilio. In general:
            - If the message is part of a registration process, pass it to the Registration Protocol
            - If the message is from a registered user, prompt the message generator to generate a new update message
            - Otherwise
                - If the message received is ‘REGISTER’ in any language of that country, enter the registration system
                - Otherwise, return error message asking them to write ‘REGISTER’ - send this in all languages of that country
    - Interfaces expected
        - “Languages of a country” from “User Database”
        - “Immediate responder” from “Message Generation”
    - Interfaces provided
        - Get incoming message
            - Arguments
                - Phone number
            - Returns
                - Newest unprocessed incoming text message
- Voice call Receiving
    - General notes on its functionality
        - Deals with POST requests from Twilio. In general:
            - If the call is not from a registered user, direct it to the Registration Protocol
            - If the message is from a registered user, prompt the message generator to generate a new update message to read out, and then offer to record an issue with crop health
    - Interfaces expected
        - “Languages of a country” from “User Database”
        - “Immediate responder” from “Message Generation”
        - “Update message” from “Message Generation”
        - “Crop Health Message Input” from “Crop Health Database”

- Registration Protocol
    - General notes on its functionality
        - Procedure
            - Firstly, request correct language with number: eg “If you speak English, enter 1. Si vous parlez français, envoyez 2. …”
            - Interpret response and respond with failure messages if necessary
            - Send message asking for name of town or village in the correct language
            - Interpret response and then send correct message to database systems
        - Needs to consider where this procedure should be ran/written.
    - Interfaces expected
        - “Country lookup” from “IVR & Voice Translation”
        - “Languages of a country” from “Database”
        - “Single-user update” from “Database”
- Web-based interface
    - General notes on its functionality
        - Allowing government workers to issue localised updates to users
        - Generates custom messages to be broadcasted to a group of users
        - Have to create a database to store all users information
        - Includes an authentication system to determine who counts as an “authorized broadcaster”
            - There are two feasible options; Give logins to relevant people, or have a verification system in place where they have to upload an ID card or using an email extension to check
        - Ensuring security is kept as a top priority so that the system is not vulnerable
            - Ensure usernames and passwords are safely stored
            - Limit the number of messages a person can generate within a time frame
            - HTTPS (not actually going to buy an SSL certificate, but worth mentioning that it would be used if we were to implement this system outside of this project)
            - Protect against SQL injection
            - Only accept complex passwords
            - If we decide to get users to upload an ID image, make sure this has the correct file extension, not over a certain MB, and ensure permissions are set so that it is not an executable
        - Provides a map of crop health reports, and allows authenticated users to hear the messages recorded as part of these reports.
    - Interfaces provided
        - A sign in / sign up page for officials to register for the online system
        - A page for the user to enter:
            - The message they wish to send

                - The location they wish the message to be sent to (for safety reasons this has to coincide with the country that they are registered to on sign up)
                - When they wish to broadcast this message
            - An input for new crop health reports
    - Interfaces expected
        - “Custom Message Input” from “Message Generator”

Acceptance criteria for the finished product

To test the finished product, we should run the system for some time and see how it performs. Specifically, we can test it using our 5 phones at the same time, registering our numbers at the same time (to test that the configuration system is concurrent). We should choose different configuration settings (different locations, crop preferences and language preferences), and then run the system for 2 days to test it. The acceptance criteria for this configuration phase is that the interfaces are navigable (ideally we should find someone who speaks a supported language like French or Swahili and ask them if they understand the interface), and the settings were correctly stored so that the rest of the test can succeed.

During these two days of testing the users should get messages at the correct time for their timezone, and be given the correct information for that location. We can validate the weather data by looking up the weather online for that location, and validate the market information by using the APIs by hand.

During the two days we can validate the custom message sender by writing messages and specifying a location that only a subset of the users should receive, and then confirming that those users receive it, translated correctly. We also can test the crop health reporting system by each recording two messages and checking that they are shown on the map, and that our recordings are correct for the right time and location.

A management strategy for the group

The group will be managed in an flat structure, with an emphasis on decisions being made communally. Now that the work has been divided up, each member has responsibility for different parts of the system, and will be focussed on implementing them and their interfaces to other components. Therefore the basis for group management will be on good communication, telling one another when we encounter issues so that we can support one another and redistribute responsibilities from people who have too much, to people who have less to do. To enable this we will meet twice weekly, usually on Tuesdays and Thursdays at 12 noon, to check in and compare our progress against the task list.

We will keep all of our code in a Git repository, hosted on Github. We will each be responsible for our own section of code, so issues like merge-conflicts should be rare as only one person will be regularly editing a file. This allows people to easily read other people’s code whilst working on their own, to ensure the interfaces between the components

work well. The repository will be mirrored in our part of the MCS Linux filespace, which we could set up as a Git server if Github fails at any point, giving us redundancy.

Technical contributions to be made by each member

- Ashwin
    - Components
        - SMS Receiving
        - SMS Sending & Text Translation
        - IVR & Voice Translation
    - Main focus
        - Integration of various components, linking front-end (SMS, IVR) to back-end (translation, Message Generation)
- Benji
    - Components
        - Message Generation
        - Data Gathering
    - Main focus
        - Server-side functionality, providing a back-end for the web-based update interface, presenting data and generating messages at appropriate times
        - Working with distinct data sources and integrating them with a uniform format usable by Message Generation
- Jason
    - Components
        - Database
        - Registration Protocol
- JP
    - Components
        - Web-based Crop Health Report Viewer
    - Main focus
        - Accepting crop health report recordings from the IVR system, and storing them
        - Displaying these reports on a map, using the same security system as the Web-based Update Interface.
- Kyra
    - Components
        - Web-based Update Interface
    - Main Focus
        - Building a secure system, which needs to query the projects main database and send information to the SMS sender.

Project Plan

Section	Task	Person	Time Taken / hours	Week
SMS System	Create function to send message (translated to Twilio API)	Ashwin	2	1
	Create function to translate message to correct language using Google Translate API	Ashwin	3	1
IVR System	Create phone lookup system using Twilio to get country based on phone number	Ashwin	1	2
	Create IVR system to deal with calls including redirection	Ashwin	3	2
IVR Integration	Integrate with data provision systems to give correct data	Ashwin	3	3
	Integrate with data to get system talking in correct language	Ashwin	2	3
SMS Receiving System	Implement system	Ashwin	3	1
Registration System	Implement system without connection to database system	Jason	7	1
	Connect to database system	Jason	3	2
Database	Set up a query table linking the languages supported by Twilio text-to-voice, Google text-to-text, and country-codes returned by Twilio. Formats need to be unified. Should be self-updatable.	Jason	5	2
	Design APIs for getting and setting user information, as	Jason	2	1

	well as querying for groups of users. Figuring out how to incorporate the database system into the rest of the server.			
	Writing the database and queries	Jason	7	2
Custom message system	Make server to accept POST requests to send custom messages written in web interface	Benji	7	3
Regular message generator	Implement system to regularly make and send update messages	Benji	7	2
Message prompt system	Implement system to generate a message for a specific number, when it is prompting the system for a more recent update.	Benji	3	2
Data Gathering System	Write crawler to regularly query various APIs for each user	Benji	2	1
	Plan out and implement database to store required data efficiently	Benji	5	1
	Create functions to interact with message generation system	Benji	4	1
Web Based Update Interface	Design the wireframes and decide how best to store data	Kyra	2	1
	Build website front end	Kyra	4	1
	Build website database for storing user data, with focus on security	Kyra	5	1
	Link to SMS system, so the generated messages get sent	Kyra	3	2

	to required people.			
	End to end testing of the site, including penetration testing	Kyra	3	3
Crop Health Report Viewer	Build database for storing user reports	JP	3	1
	Build website for displaying and playing back reports	JP	10	1-2
	Securing the website to only allow authenticated users to access data	JP	3	2
	Build system for accepting new crop reports	JP	5	3