MULTMEDIA ENABLED CROWDSOURCING

APPLICATION FOR MOBILE DEVICES

By

SAM HONARVARALIJANI

Bachelor of Computer Science

Multimedia University

Cyberjaya, Selangor, Malaysia

2011

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2013

MULTMEDIA ENABLED CROWDSOURCING

APPLICATION FOR MOBILE DEVICES

Thesis Approved:

Dr. Tingting Chen

Thesis Adviser

Dr. David Cline

Dr. Nophil Park

ACKNOLEDGEMENTS

My sincere gratitude goes to my advisor Dr. Tingting Chen who has been a great mentor and an impeccable help throughout my entire thesis.

My appreciation also extends to Dr. David Cline and Dr. Nophil Park for their advice during my thesis proposal presentation and also for serving on my graduate committee.

And last but certainly not least, I would like to express my profound appreciation to the most wonderful, supportive, and tremendously understanding parents and brother, Bahram, Sholeh, and Yasha who have always been there for me and their constant support has provided me great encouragement to go on.

Name: SAM HONARVARALIJANI

Date of Degree: JULY, 2013

Title of Study: MULTMEDIA ENABLED CROWDSOURCING APPLICATION FOR MOBILE DEVICES

Major Field: COMPUTER SCIENCE

ABSTRACT

Crowdsourcing has been a great topic of research and development recently due to the low cost of operation and perpetual availability. Smartphones have also grown to be very popular and pervasive. They are available these days with tremendous processing power equipped with great cameras, microphones, HD screens, GPS, and etc. It has been a while since cell phones and recently smartphones have been exploited for crowdsourcing purposes. There are already apps that provide features to report and help in process of recovering from disasters and natural phenomenon. Before smartphones era, text messages were used as a way of crowdsourcing as well as a source of income. Translation of small pieces of text was a common example of crowdsourcing where the user would receive a small text, translate and send it back and for every text they would make a small amount of money. However there has never been an app that is both multi-purpose and multimedia capable. In our proposed method and application (Crowdesk) the user is capable of choosing the type of their message including image and audio. Since we are dealing with larger file sizes when working with audio and image files, the network connection and transmission rate becomes an issue. We have utilized buffering techniques and breaking down the files into smaller pieces to overcome the connectivity speed problem. Tests and benchmarking have been performed for both cases where connected to a Wi-Fi access point with an Android device, or when 3G/4G is used for transmission of data, and for all cases, the results have been satisfactory. Crowdesk is solely meant to be a representation of what can be achieved by utilizing people's contribution on a mobile platform.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1 Overview

Cell phones have come a long way since their early days. These small devices that nowadays we call them smartphones were initially designed for the sole purpose of oral communication. The very first models were quite a few inches tall and considerably heavy. With the advancements of processors and flat screens, the cell phones started to get smaller, faster, and have much better screens in terms of number of pixels per inch, size and heat. Over the past 10 years this process has been expedited mostly because the processor manufacturers have done a great job by making very fast processors while keeping the size and heat level low such that most of advanced smartphones possess processors that are faster or at least equivalent to those available on laptops at the beginning of the decade. All this has made smartphones to be even more popular than computers recently [Weintraub]. Having such processing power means that more advanced operating systems can be loaded on them and more sophisticated games and applications will be able to run on top of those operating systems. A good example of such operating systems is Android that has the stable core of Linux operating system. We will talk about Android OS in more detail in the rest of this thesis.

Besides being equipped with faster processors, cell phones are shipped with other parts such as remarkable built-in cameras making them a proper device for online video chatting, as

well as satisfying everyday and non-professional photography needs, good processing power, crystal clear and high definition screens, microphones, and reliable speakers.

These features make these small devices into incredibly multimedia-capable gadgets that fit in the palm of hand. As the result of these features in the past few years, smartphones have become one of the top software development platforms and tons of creative apps have been designed for them. As a proof, at the time this documentation is being written, more than 850,000 apps have been officially submitted to IOS app store [Wikipedia App Store] and approximately 800,000 apps uploaded to Google's *Play Store* [Wikipedia Play Store].

One of the great ideas to utilize the capabilities of the smartphones that has recently become popular is the notion of Crowdsourcing [Howe 06]. According to Merriam-Webster dictionary definition of Crowdsourcing is "the practice of obtaining needed services, ideas, or content by soliciting contributions from a large group of people and especially from the online community rather than from traditional employees or suppliers" [Merriam-Webster]. A good example is an actual situation that instead of paying thousands for dollars to professional photographers for certain photos, the required photos' description can be posted on a website and semi-professional photographer will do the same task for much less cost or instead of paying professional reviewers for books that are to be published, the text can be split to smaller pieces and be sent to several reviewers [Doan et al. 11]. Another good instance is a fully human-driven database that uses the power of people for running queries on a database [Franklin et al. 11].

In simpler words Crowdsourcing is same as outsourcing except that there is no specific aimed source and any one with the right tools/knowledge will be able to be one of the sources, hence the word crowd. The use of crowdsourcing stands out when mixed with computers or more

specifically mobile devices whereby a user can participate in an outsourcing activity during their free time. As an example, someone that spends some of one's very day's time on commute, can utilize one's time by translating some texts in a particular language and earning money by sending the translation back through text messaging.

Having the idea of smartphones and crowdsourcing together envisions some very unique applications. Since smartphone are not confined to only text and more importantly have access to fast Internet connection nowadays, they can act very much like a computer and perform variety of tasks. For instance, an app that can help blind people to recognize items in a supermarket or helping them to recognize what is contained in a can. Taking a photo, and using the app to upload the photo can send the answer back in audio format stating the brand and content of the can.

1.2 Problem Statement

In our project, we thought of much bigger picture. All apps available today for crowdsourcing are too specific and are only suitable for certain tasks and therefore might not be very useful. We had the idea of all in one for our project and being able to crowdsource pretty much anything is quite fascinating. There is also the idea of providing enough incentive for users to participate in the crowdsourcing activity which is out of the scope this thesis but there already existing mechanisms that provide good solutions [Yang et al. 12].

We have also categorized the types of posts users can upload based on the timeliness and time sensitivity of the required information/data. Considering a medical emergency in which someone requires fast information on helping someone else that had just a stroke, the user needs a

way to specify that the post is urgent and quick response is required. Such cases motivated us to utilize different mechanisms for different questions based on their priorities.

CHAPTER II

REVIEW OF LITERATURE AND SIMILAR PROEJCTS

2.1 Crowdsourcing to smartphones

"Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing", proposes on the topic of sensing capabilities of smartphones. Since smartphones are nowadays shipped with great sensing peripherals such as microphones, cameras, GPS and many others, it makes them the perfect tool to be used as a sensing device to collect data and use their Internet connectivity to transmit the data. One factor is not considered in many cases is the lack of enough incentives for the user to contribute to the project. Most probably because most users prefer not to enclose their data with GPS and location data and have them be stored and accessible. Therefore the writers are proposing two mechanisms to increase users' incentive, platform-centric model and user-centric model. The paper doesn't provide any mechanism or general architecture on the way system should work rather they propose only methodologies for providing incentive to user by implementing auctioning mechanism and paying user more than it costs them to contribute.

2.2 Crowdsourcing with smartphones

The authors in this paper have very close viewpoint to our aim at having a multifunctional app for crowdsourcing [Chatzimilioudis et al. 12]. They have developed three different apps presented in their paper as follows:

SmartTrace+: provides mobility patterns and movement trajectories according to a single request. Most users are not willing to share their movement trajectories or patterns due to privacy issues. However in case of a specific request the user will be considered anonymous while providing the data. An example provided is in case of bus ride. If someone planning a trip requires knowing the specific bus routes and the user is willing to know whether a specific route is taken by certain number of users, they can send a request to all users and users will voluntarily take part in providing their answer by sending the collected information on the specific route.

Crowdcast: an application that uses smartphones locality capability to find nearest neighbors. This feature can be useful in cases of SOS, immediate help or medical emergency.

SmartP2P: intelligently runs queries according to location and GPS data. As an example if a user sends out a query for finding a picture of golden gate bridge, the app will run the query on the smartphones located in California and more specifically in San Francisco since a smartphone located in San Francisco has a higher chance of containing a picture of golden gate bridge than a smartphone belonging to someone living in Manhattan.

2.3 Crowdsourcing of Pollution Data using Smartphones

NoiseTube is another interesting example of crowdsourcing app for smartphones [Stevens and D'Hondt 10]. There already exist few apps related to traffic [Mohan et al. 08] and noise level [Rana et al. 10] that utilize crowdsourcing concepts. In recent years smartphones have been greatly exploited for environmental sensing [Das et al. 10] using people's smartphones as

the platform with mechanisms [Bruke et al. 06] to promote people's participation [Campbell et al. 08]. Nowadays our lives are polluted with different pollution sources such as noise pollution, air pollution and etc. NoiseTube app is used to collect different types of data from different smartphones sensors (microphone, GPS, user's input) and the collected data is then uploaded to a centralized server, which tracks and records the input data. These input data later is later on converted to maps of pollution areas, which indicate different levels of pollution in different areas. The app uses a signal-processing algorithm that requires the user to record one-second long audio. These picked up samples are then illustrated using numeric and graphs. Each of these recordings then are tagged with GPS and location data and the user is given the option to provide more details on the recording, such as source of noise, subjective impression and etc.

Few of the mentioned problems by the authors include lack of real usability of the app and the way user interacts with the phone. The phone if left in the pocket or purse will not be able to pick up real and trustworthy data and it also may conflict with the normal use of phones. Also one of the most common issues in crowdsourcing is the validity of the responses from the user is discussed.

CHAPTER III

DESIGN AND IMPLEMENTATION

3.1 Overview

The name Crowdesk comes from two separate ideas of crowdsourcing and helpdesk. It pretty much represents the concept of what we have been trying to accomplish in this project. Our main purpose and ultimate goal has been to utilize the power of people and create a perpetual and constantly available service that can help people with their everyday or emergency needs. There are forums and online discussion boards that allow people to ask any type of questions but they normally require a PC, knowledge of computers, knowledge of available forums and most importantly, they don't fit in your palm. With Crowdesk however, one only needs an Android device (tablets, cell phones) and an active Internet connection and they will have access to crowd's knowledge in a few touches. Moreover, replies also lay few touches away. In the other end, for most people with the proper knowledge, a computer is a requirement if they know the answer to a specific question. They have to access that forum online, login, take a look at the list of questions and they won't always be available due to both being away from their PCs as well as not receiving any notifications. With our design however all these hassles have been diminished and availability and accessibility are two key features of it.

Crowdesk currently is only available on devices that run Android platform. The app however has been designed such that it can run on any device loaded with Android regardless of screen size and screen orientation. It would be one of the work-in-the-future tasks to be performed to port the application onto IOS driven devices if this was to be turned into an actual business model and be industrialized and commercialized.

3.2 Design

Crowdesk application is consisted of two major components: a multi-function server and an Android app that communicates with the server mostly through http protocol. In the following sections we discuss each component in more detail.

3.2.1 Server Side

The server side is a multi-function server. The reason we call it multi-functional is because it runs and executes multiple tasks and multiple services. The server major responsibility is to receive and respond to http calls and direct them to the proper services. The services running on the server includes Python service for file management, PHP service for handling database calls and briefly to handle send/receive of files. The server also contains a MySQL server that is in charge of recording and retrieving user information and post information. We will illustrate and explain each of these services separately along with the technologies required for all these communications in between.

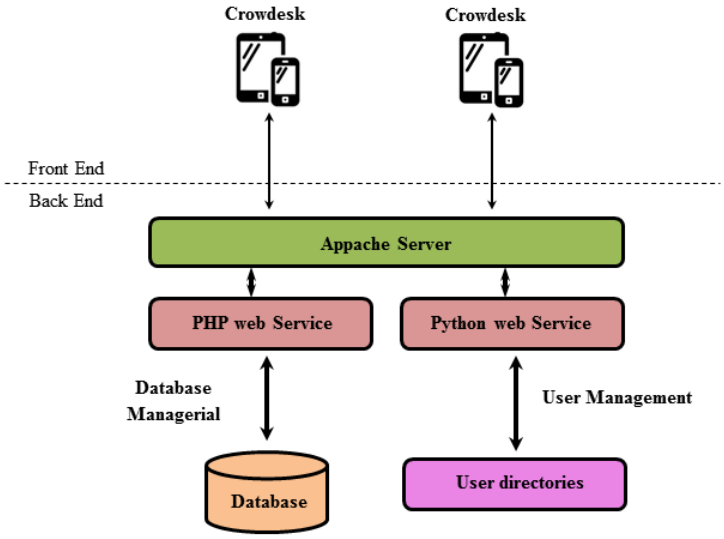

Figure 1. The general structure and communication pathways in the system

Python Web Service: Python recently is vastly being used in web technologies majorly because it ships with out-of-the-box and ready to use libraries that make it remarkably efficient to be used with different technologies. Python also as a scripting language, is armed with super easy operating system API calls to handle files and file-related operations. As an example, Moving, deleting and creating files and directories is an extremely easy task in Python and feels completely natural. It takes few lines of code to write an http server in Python while it requires hundred lines of code to do the same in languages such C++ or Java. Also great web frameworks such as *Django* are available for Python that make web development fast and efficient.

In our design however, we used most basic functionalities of Python for handling the contents of the posts uploaded to the server, which could be of any type of audios, images and texts. In other words Python was acting as our back-end storage engine.

Every file that contains post information are uploaded to the server, initially are located in a folder named *tmp* folder. Then the app calls a Python service module to relocate the file. The

module receives the name of the temporary file, username to whom the file belongs to (this actually gives a lot of information regarding where the file should be moved to) and finally the extension of the file (which determines the type of the file and subsequently the proper folder in the user's directory).

Another Python module is when a user is created. As soon as a new user registers, after updating the database, a Python web API is called that receives the username (email address of the user) and creates the required folders.

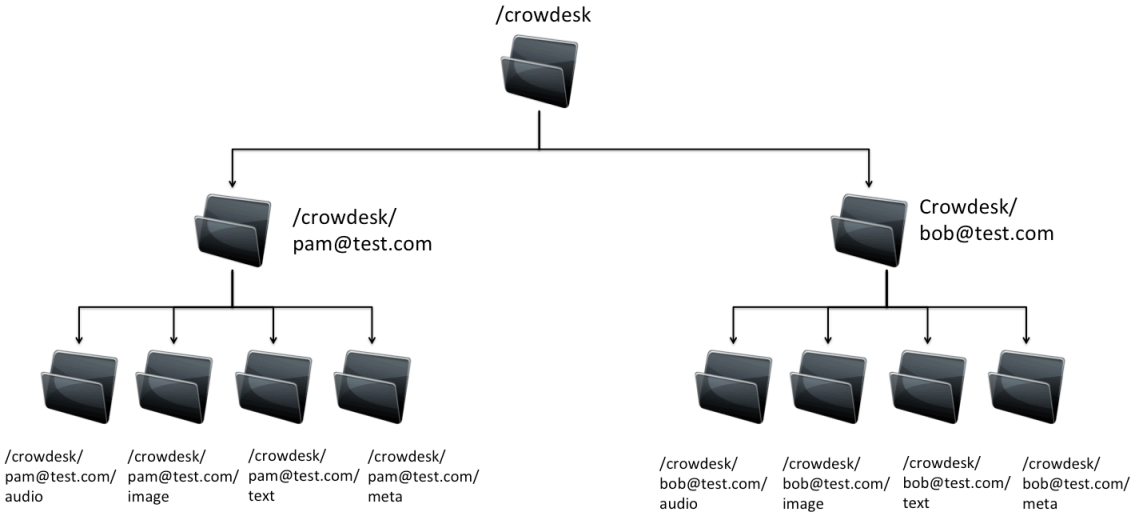

Figure 2. Example illustration of the file structure for 2 different users, "pam@test.com" and "bob@test.com"

As it can be seen in Figure 2, user Bob registers with his full name and email address and password. This information is stored in the database (we will discuss more about the database structure at the end of this chapter) and the email address (i.e. username) is used by the Python service to create a user's folder. Four sub-folders are then created as below:

- *text*: used to store the full description of a text post. This is different than the short description that contains a short summary of what the post is about.

- *image*: all the images from image-posts posted by the user are stored in this folder.
- *audio*: audio files recorded by the user will be dropped in this folder.
- *meta*: This is reserved for further and future work. This was initially intended to contain all the meta data regarding the post and their relationship with the user.

PHP Web Service: PHP in essence is very well established in web architectures for its fluency to deal with back-end databases and generating dynamic web pages. We intensively use PHP for handling SQL queries and briefly for file handling.

Every time a post is submitted, the files uploaded with the post are uploaded and stored in the user's local directory. To do so, a PHP module is called and the file is posted using a POST http request and 2 stream lines are opened at both ends and file is gradually uploaded to the server. This ensures that there is basically no limit (except what is set on the web server configurations, i.e. 200MB) to the size of file to be uploaded and we never occupy the mobile device's memory for buffering the content on it. The PHP module then will return a random filename through a http response message and that is read by the device and passed to Python module explained above to rename and relocate the file to the proper location.

We also utilize the PHP strength to communicate with the MySQL server and any query for updating, inserting and retrieving data from MySQL goes through PHP module.

MySQL server: Very hardly there are actual applications nowadays that don't use databases, and by database we mean any kind of storage and retrieval of data such as XML, and of course MySQL and its derivatives. We use MySQL in our project to store following information:

- User's information and data
- Posts and their related information
- User-Post relationship

The structure of the database we use is consisted of four tables:

- *users*: Stores users' login information, name, UID, and list of subscriptions
- *posts*: Stores posts' information such as submission date, UID, tag, location of attachment and etc.
- *dismissed_posts*: An identical table to posts except that stores posts that are already closed and dismissed.
- Post response tables: These tables are dynamically created by the prefix of "t" and the UID of the post attached. Every post that gets its first response, a new table is created for that post. For example, a post with uid of 195 will have a table with name of *t195*.

3.2.2 Client Side

On the client side, the app is designed for devices that run Android version 2.2 and higher. Every user needs to register their username, password, and full name before they can use the app. Once the user is registered, they will be asked to choose the list of categories that they are interested to see the posts. This list of categories is stored in the database and is used to populate the post feed for the specific user. They are given the capability to update this list at any time.

The user can perform three major tasks; one is to post a new message in the format of text, image, or audio, next is seeing a list of currently active posts and post respond to each of them if interested and last seeing a list of the user's own posts.

In the post feed, all the posts with the proper tags are listed and the user will only see the short description of the post. To see more details or see the actual message in text/image/audio format they need to tap on the post and then app will navigate them to a new page that shows the full post body.

The reason for this is, since we are often dealing with images and audios, there is no need to download all the post's data that are not even user's interest specially when running on a cell phone 3G/4G network. Therefore the content will be queried and downloaded on the devices as soon as the user shows interest in a specific post by clicking on download attachment and the user will be asked to wait for the attachment to be downloaded.

The app also uses Android's internal database better known as SQLite. Every time the user logs in, we store the state of the application in the internal database and the when the user runs the app, the app checks internally to see if the current user has ever logged out of the application. If the user never logged out, the flag is set and the app navigates directly to the front page. Otherwise it would ask the user for their credentials before allowing the user to use the app.

3.3 Implementation

In the previous section we talked about the general design and methodologies used in the system and in this section we are going to talk in details about the implementation and technologies used to implement the design. We'll start with the server side again.

3.3.1 Server Side

As it can be seen in Figure 1 the server is an http Apache server running on Ubuntu 13.04 with LAMP package installed on it. All the Python codes were designed for version 2.7 and PHP version 5 and Apache version 2 was used as the http server.

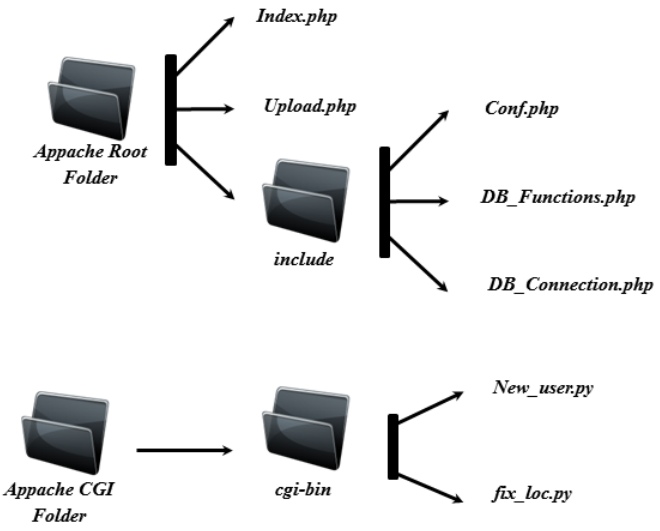

Figure 3. The webserver structure and Python and PHP web services modules. Apache calls to Python/PHP modules and they run and form an http response and send it back to Apache

Python Web service: Python web services or CGI was primarily used for file handling on the server side. Two major modules were the one creating the user and relocating the uploaded files to the proper folder.

- *new_user.py*: When user entered their credential for the first time and clicked on register button, one action is to prepare the user's folder on the server. This was basically creating four folders as described in the previous section and setting the right permission for the Apache server to able to write to those folders. The general mechanism of communication was by sending a POST http request to the right CGI module and waiting for its response. The response would be in form of an http response compatible with *SOAP*[1]. The responses are in conventional http response codes[2]. A "Status: 200

[1] SOAP (Simple Object Access Protocol) is a protocol to transmit structured data between computers in a network

[2] A complete list of these can be found at *http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html*

OK" response would be sent back to calling method in Java and the Java code would analyze the response and will react accordingly. If the parameters composed in the *params* section of the request aren't what expected by the server, the server will reply with a "Status: 400 Wrong Parameter" message. And in any case of unexpected error or exception, the server will issue a "Status: 500 Internal Server Error" message. If this module returns an OK message, the app will go ahead and add the new user to the database.

- *fix_loc.py*: This module is in charge of moving the temporary file to the right folder under user's root folder. The responses back from the module are pretty much the same as the new_user.py but it actually receives three parameters, the temporary filename received from upload module, the new filename and the type of the file (audio, image, text).

There are few other modules for storing configuration and other details that are of minor importance.

PHP web service: PHP is almost always a part of implementation if we are dealing web servers and databases. Although recently a lot of big firms are moving away from PHP but it's still pretty much one of the best server-side scripting languages. We used PHP for the same purpose and the entire set of queries to Crowdesk's MySQL database is handled through PHP scripting. The structure of PHP implemented on the server is illustrated in Figure 3.

include directory includes all the modules required for communicating with the database.

- *config.php*: includes the required information to connect to the database, such as username, password, address of the MySQL's server, and the name of the database.

- *DB_Connect.php*: as the name hints, this module is in charge of establishing a connection to the database. It returns a handle to the database to calling module and from that point on every call to the database will be though that handle.
- *DB_Functions.php*: probably the most important PHP script. Any action required by the application to be done to the database, has to go through this script. Actions such as adding a new user, modifying table fields, authenticating user, adding and removing posts and etc.

Before moving on to other modules, we need to specify here our password storage mechanism. Since the user's privacy is always of crucial importance, we did take that seriously too. Therefore we use a common method of password storage and retrieval mechanism. Every time a new user registers, the password is sent to the database. Then we use PHP's built-in hash function to generate a unique "salt" value and then we concatenate that to the password. Then, we run the password through our encryption mechanism and the encrypted value will be stored in the database. When the user tries to login later on, the password received from Android is sent to PHP code, PHP will use the received password and runs the same procedure on the received password and then checks it against the encrypted password received from the database and if they are a match, the user is authenticated and user is given permission to login or otherwise the error messages is sent back to the app and the login page on the Android will show a "wrong user/password" error message.

On the root of the Apache server's folder, we have few more PHP modules. One and most important one is the *index.php* that essentially handles all of the http requests.

- *index.php*: Every single call to this module should be an http's POST request. The design of this module is such that, initially it looks for a "tag" value in the POST parameters. If it doesn't find, "wrong parameters" message is returned right away. "*tag*" is the actual

way of telling the PHP module what the app is requesting for. Tags such as "*register*", "*login*", "*get_subscriptions*", and "*add_post*" are few examples. This module goes through a very straightforward procedure for every request and is used specifically for dealing with the MySQL server. All the database queries go through this module and it receives the responses back from the SQL server. A tag in this module determines each operation and the tag is concealed as one of parameters of the POST request. Get the "tag", identify the proper action, find the right user, perform the action and commit it to the database and finally, form the response in a JSON object and return the JSON response back to the server. We will talk more about the use of JSON in responses received from the server in the next section. The app will analyze the JSON message and will make decision based on the parameters to go ahead and consider the request as successful or not.

- *upload.php*: is the one that does the important task of uploading files. This module is very sensitive since it needs to be reliable enough and memory efficient. Our initial design was not successful since it was consuming memory to store the files in a buffer and right after the whole file was received, it will flush everything to the disk. That proved to not be very efficient and eventually the current design was implemented. We do have modules in our system that are implemented in Python and they can perform the same action but since the PHP code looks more stable and smaller, it was decided to use PHP. The way this PHP file works is truly simple. It generates a 20 character long random file consisted of alphabets and numbers, then it saves the files received to disk and assigns the file the randomly generated name. Consecutively it returns the filename in an http response back to the Java caller module on Android. Then the Java code calls the fix_loc.py, which was explained earlier, with the temporary filename received.
- *download.php*: this PHP module is a very handy and complete piece of code that takes care of any kind of download. What we needed here in essence was a piece of code

running on the server to provide a download-like mechanism for our Android code to call; upon the call, the file should be downloaded in the proper formatting and be encoded properly according to file type. In other words, this snippet looks for the type of the request and extension of the file that is being requested and makes decision for content-type being provided. The download.php receives three parameters, the file (full path of the file on the disk), the name of the file (content-type decision is made based on this parameter), and an optional mime type that we never use since the type is dynamically determined. Once it receives the request, it looks at the file type and chooses the right file type and determines the length of file being requested. Upon determining the size, it starts reading the file into a buffer and wrapping them in http response packets and sending them piece-by-piece to the requesting end.

MySQL: The implementation of database is very straightforward. It's based on the design explained above there are no database design complication involved.

3.3.2 Client Side

Android development is proven to be not so easy. Although Java is an impeccable programming language and huge portion of programming for Android is done in Java, yet it's far from being simple and/or intuitive. Interface's UI are coded in XML format and when the project is compiled and run, during runtime, the XML files are inflated and compiled in Java format. Also any of the device's peripherals or external resources such as Internet, that are to be used, need to be declared in the *AndroidManifest.xml* file so the app is given the right permission to access those resources.

Crowdesk is designed to be compatible with most devices and screen sizes and any orientation shown in Figure 4. The activities are all scrollable which means even though the screen size is too tiny the user can scroll through the content and perform their actions.

Figure 4. Nexus 7 front page illustration. Left: In Landscape mode, Right: On Portrait mode

The minimum SDK required to run the app is SDK version 8 (Android 2.2.x, a.k.a *FROYO*) but the target SDK is 17 (Android 4.2, a.k.a *JELLY_BEAN*).

There are 12 Android activities in Crowdesk app. Aside from the apps, there is the *Library* modules that are not directly related to Android activities but are used and called by the activities to carry on specific tasks, such as communicating with Database, Parsing the JSON messages received from the http server, *View* and *List* adapters for Android lists, general configuration of the app, such as the IP address of the main http server, and few other. The activities and their connections and relationships are illustrated in Figure 4.

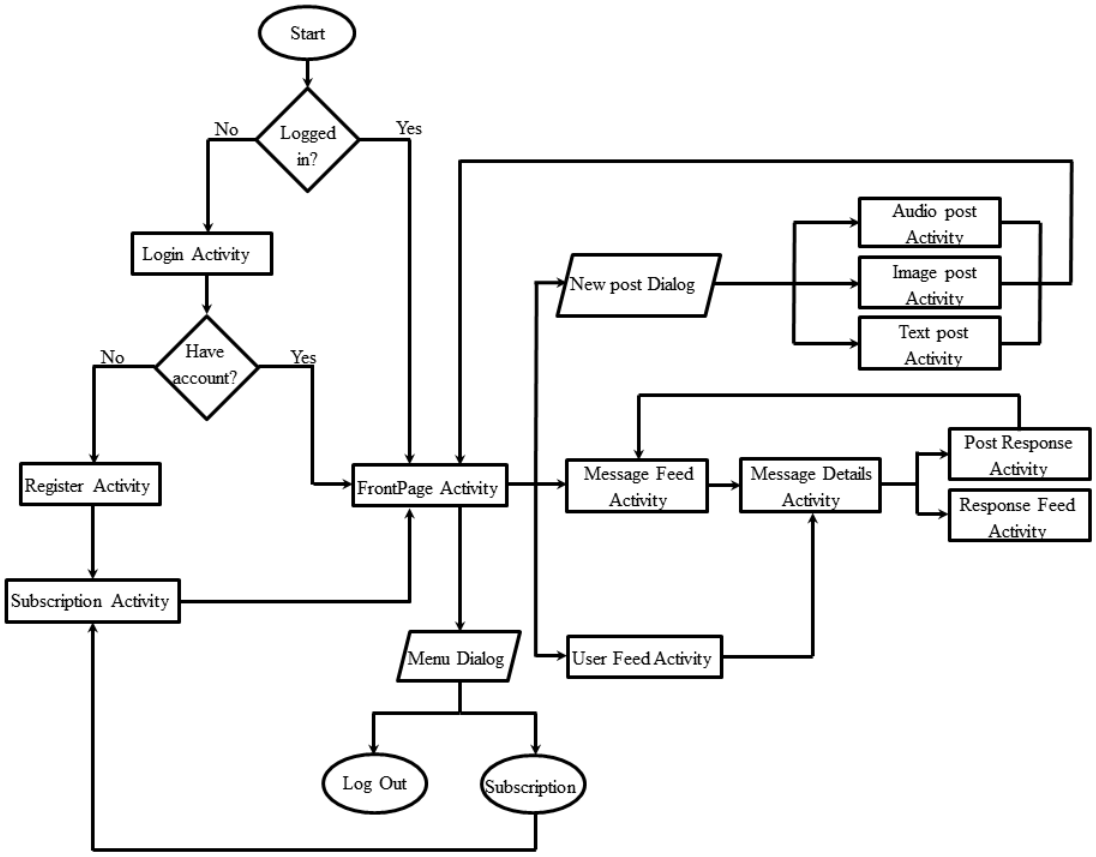

Figure 5. The complete Activity flow of the app. Oval shapes show actions, Diamonds show decisions, Rectangles indicate Activities and Parallelograms demonstrate the Dialog boxes

The entry point to the app is either one of two, *loginActivity* or *registerActivity* depending on the login status. If the user is already logged in, the *onResume*() module will check the app's login status and if the SQLite on the device indicates that the user is already logged on, then the app will directly jump to *FrontPageActivity*. When in so called "front page", the user has three different types of action to perform:

- Create a new post: It provides the user the interface to create a new post and send it to the post feed. Every post in Crowdesk has a "short description" section that is to give users a quick hint on what the post is all about. Users should keep these short descriptions actually short and by default, the app won't allow them to write more than 200 characters. After the short description, depending on type of the post, it's either a bigger

textbox that allows the user to type in a question or problem and post it as a text message. In case of image, user can snap a picture with the device's camera and the image will be sent as part of the post and short description is used to identify what the user requirement is. Audio posts are same as images and the app will record the sound by the device's microphone and will attach that sound file to the post when sent. Every user is also given the choice of indicating their post as time sensitive, means something that requires immediate attention from the other users. Here one good mechanism to use would be to use smart notifications and if there is an emergency post, the user's device (Smartphone, tablet) will keep beeping and/or vibrating to indicate that an urgent matter is waiting for their response. But in case of normal post, a single one-time beep/vibration would be a good notification method. This functionality however is not implemented due to insufficient time.

- View the list of active posts from everyone: This page invokes a database query that retrieves all the posts in the database that are currently active and shows a list of all these posts. However by design, every user will not see their own posts. Every post is clickable and when clicked, the user will be taken to *ViewPostDetailsActivity* activity that shows the details and actual content of the post. It's worth mentioning that in the list of posts, only short description, tag, type of post (text, audio, image) and priority will be shown and nothing else will be queried from server (although each posts UID is implicitly requested and stored to be used later on if the user needs to reply to the post). Since we deal sometimes with big data files such as audios and images, and most users access the app on their cell phone using their 3G/4G networks, therefore the transaction amount will be kept to minimum if data is retrieved only when needed. As the result, when the user clicks on a post, a new Activity called *MessageDetailsAcitivity* will be invoked. This Activity still wont call to the server for the attachment and it will until explicitly requested by the user through the Download Attachment button. There are few minor

details in here where the user navigates from the list of posts to one particular message. Since we don't call to the server again when the user clicks on a post, the post somehow needs to acquire the information it requires, such as the post's UID, type, and the short description of the post. All these are passed through a facility that is offered by Intents structure[3] in Android platform. Any Intent in Android can pass or receive "extras" when calling or being called. Through this object passing mechanism the programmer can pass data from one Intent to another in different data type formats including Strings. We have also exploited this feature to pass the post's UID from one interface to the next and finally when the user in *PostResponseAcitivty* taps on Submit reply, the UID is already available for the Activity. This UID is actually very crucially important since it's the main key to figuring out where and in which table the response should be saved.

For every active post, as soon as the first response is received, a new table in MySQL is created with the UID of the post used as part of the table's name. To be more precise, if the UID of the post is 981, the table created will be "*t981*". This also brings so much ease in the process of the retrieving posts and their related responses later on. In the section explained below regarding user's list of posts, when user clicks on a post, without any requirement to check or run a query to get the table name or related field in the table, the UID is directly used and the table name is constructed and a simple SELECT query is enough to retrieve all the responses. There is also another implementation detail that provides better user experience. Instead of having a general, single UI for all different types of post's details page, the app provides different interfaces based on the type of the post. In fact another reason for using Intents extras has been utilized here. When the user clicks on a post to navigate to posts' details interface, based on the extent received

[3] Intent is an Android activity in its abstract form. When invoked, it starts a new activity and can be connected to a "content" as its user interface.

in the calling Activity (*MessageDetailsActivity*), the app makes a choice between three different interfaces for Audios, Images, or Texts and uses customized features for the specific type of post.

- View their own posts regardless of their status, sorted based on date: This page shows the list of posts by the user. The user will be able to see the content by tapping on the post but will not be able to answer their own post. All posts, active or closed posts will be visible and sorted chronologically and user can scroll through them.

On the front page of the app, the user's menu button also provides two options:

- Logout
- Edit subscription list

As the name expresses, Logout simply sends a command to set the flag that the user is logged out and next time the app is started, the user needs to login before accessing the app. Edit subscription list, allows the user to change their list of interest. Upon that next time user checks on the active posts, they will see the updated posts based on their update subscriptions.

CHAPTER IV

RESULTS AND BENCHMARKING

In this section we are going to investigate on the performance of the system as well as its general scalability measures. First we will talk about the system configuration and platform, and eventually discuss about the measurements and their results.

The entire Crowdesk system, as discussed before, is based on a Multi-functional Server and set of Android devices that communicate with it. The configuration of the system we have used for development and our tests is as follows:

- Server:
  - Hardware: Intel Pentium4, 3.4GHZ with 2GB of RAM
  - Operating System: Ubuntu 13.04 with XFCE4 as the GUI
  - Connectivity: AT&T home Internet, 12MB package
- Android App:
  - Device: Nexus 7, Model ME370T with 16GB Internal Storage
  - Android Version: 4.2.2

We have run tests to measure the responsiveness of our system when only a single Android device is operating and communicating with the server. The tests are based on responses of separate Activities.

Name of the Activity	Nexus 7 on Wi-Fi	HTC Inspire 4G on 4G
Login	212 ms	354 ms
Register	552 ms	424 ms
Get Subscription	176 ms	207 ms
Update Subscription	210 ms	391 ms
Message Feed	201 ms for 23 items	662 ms for 23 items
Post Image	2983 ms for 611KB	6763 ms for 1.2MB

Table 1. Time latencies for access, query, download, and upload of messages and attachments

From the results above, it can be understood that the response times and general performance of the system is quite acceptable. However there are many factors in play that have great influence on the responsiveness or activity rate of the system. As an example, we ran the test on posting an image on the highest possible quality of the internal camera on Nexus 7 device. Some newer devices, such as Samsung Galaxy S3 have better and more sophisticated cameras that provide better quality and resolution and therefore bigger files. Considering that the newer devices also utilize the 4G technologies and better processing power, it's expected that the wait time for uploading images will not be outrageous and relatively acceptable.

It is worth mentioning that this system is purely experimental and in no way is meant to be used as a commercial app with the current configuration. This concept also brings us to the next concern that is shared among all distributed and network systems; the concept of scalability.

There are two broad category of scalability, vertical and horizontal scalability. In Vertical scalability, the system is expanded on top the current system. In simpler words, if in our example, Crowdesk's main storage space is running out of space, the vertical expandability solution would attach more disks and expand the space on the server. Or if the network connectivity of the server is not enough, we provide wider and faster network connection to the system. This method

however cheaper but is not always efficient and is in fact, very risky if for any reason any of the disks fails or if the network on the server shuts down or gets cut out for a while. In contrast, we have horizontal scaling or expansion which tries to keep the performance and activity of every entity up to a certain level but instead having multiple of those smaller entities as backup or redundancy. This method although is impeccably safe and trustworthy however is expensive and has some very fundamental deficiencies that we will discuss below.

Redundancy is always a factor in distributed systems and is a very crucial one. In other words, consider Crowdesk application and all the users' data stored on one single entity. In case of a disk failure, all users' information and data is lost and in no easy way are they retrievable. Losing Terabytes or in some cases, Petabytes of data is something that is not forgivable and all the trust in a system will be destroyed. The solution is simple and is achieved by having multiple copies of the data, so in case of a loss in one of the servers, other copies are available and will be in service right away. It's generally recommended that 3 or more copies of the data provide enough redundancy. But having redundancy will solve one issue and will always bring into picture more problems such as how to keep the copies of data on the severs updated, what happens if one server has old, out-of-date data while the user accesses it?

In industry there are ways of somehow resolving these issues and if Crowdesk was ever going to grow to that size, those methodologies need to be applied to the system. Crowdesk utilizes databases that are common member of any expansion and the conventional methodologies can be applied to the system. To provide an example porting the database to Apache Cassandra will give them enough scalability and will allow the database to be spread out amongst multiple nodes and servers. Load balancers and store-and-forward technology are few other examples of possible expansion and reliability techniques that are viable to be applied to Crowdesk.

CHAPTER V

SUMMARY AND FUTURE WORK

Looking closer at the idea behind crowdsourcing somehow brings to attention that the crowdsourcing and cloud computing are somehow in contrast. Cloud sourcing intensively strives to unify processing and storage units to one single powerful entity (cloud) while crowdsourcing tries to spread out tasks amongst as many entities as possible and try to accomplish tasks by having many smaller/slower units. Cloud computing has made spectacular progress and we strongly believe that crowdsourcing is still at its very infancy and the amount of current on-going research and projects hovering around it indicate that more attention is being drawn towards it.

We also have tried to make the role of crowdsourcing bolder and more obvious by designing a multi-functional, multimedia capable application on a mobile device. Although mobile devices are always limited by their battery life, recently new methods have been proposed to extend the battery life on mobile devices [Cuervo et al. 10].

Crowdesk by no means is a complete, fully scalable project and the sole purpose of such application is to present and demonstrate the possibilities and capabilities of power of crowdsourcing. We have tried to represent the plausibility of such power in our everyday life and providing a platform for further development and expansion. As the result there are so many functionalities and factors that can be added to the system to deliver better user experience, ease of use, and of course more functionality.

As we discussed above, the server side needs a lot of work to be able to handle request more efficiently as the number of incoming requests increases. On the app side however, features such as notification upon a new post, multimedia response to posts, compression mechanism for pictures and audio files to reduce the upload/download time, better and more sophisticated user interface, and of course IOS version of the app for iPhone and Ipad users, and many other are set of possible features to be added to the system.

As we discussed before, providing the incentive for the user is without a doubt one of the most challenging obstacles in crowdsourcing. We initially proposed and designed a mechanism but due to lack of time were not able to implement this section of the system. By providing auction like mechanism such as Vickery Auction, and tagging the posts based on their timeliness level, rewarding and rating mechanisms to reward those users with most related and precise answers, it is believed that Crowdesk will be able to attract more people into contributing and committing to the system. That states that bilateral gain will be possible both for people posting a new request and for people responding to a request and the concept of popularity and auctioning will provide the incentive and drive to both participation as well as more reliable responses.

REFERENCES

[Franklin et al. 11] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin, "CrowdDB: answering queries with crowdsourcing", *SIGMOD '11 Proceedings of the 2011 ACM SIGMOD International Conference* on Management of data, pp. 61-72, NY, 2011

[Doan et al. 11] Anhai Doan, Raghu Ramarkrishnan, and Alony Y. Halevy, "Crowdsourcing Systems on the World-Wide Web", *Communication of the ACM*, No. 4, pp. 86-96, Vol. 54, April 2011

[Chatzimilioudis et al. 12] Georgios Chatzimilioudis, Andreas Konstantinidis, Christos Laoudias, and Demetrios Zeinalipour-Yazti, "Crowdsourcing with Smartphones", *IEEE Internet Computing*, published by IEEE Computer Society, pp. 33-44, 2012

[Stevens and D'Hondt 10] Matthias Stevens & Ellie D'Hondt, "Crowdsourcing of Pollution Data using Smartphones", *UCL Discovery*, 2010

[Howe 06] Jeff Howe, "The Rise of Crowdsourcing", *Wired Magazine*, Issue 14.06, June 2006

[Yang et al. 12] Dejung Yang, Guoliang Xue, Xi Fang, Jian Tang, "Crowdsourcing to Smartphones: incentive mechanism design for mobile phone sensing", *Mobicom '12 Proceedings of the 18th annual international conference on Mobile computing and networking*, pp. 173-184, NY, 2012

[Weintraub] Seth Weintraub, "Industry first: Smartphones pass PCs in sales", "http://tech.fortune.cnn.com/2011/02/07/idc-smartphone-shipment-numbers-passed-pc-in-q4-2010" accessd on May 2013

[Mohan et al. 08] P. Mohan, V.N. Padmanabhan, and Ramjee. Nericell, "Rich monitoring of road and traffic conditions using mobile smartphones", *Proceedings of SenSys*, pp. 326-336, 2008

[Rana et al. 10] R. Rana, C. Chou, S. Kanhere, N. Bulusu, and W. Hu., "Earphone: An end-to-end participatory urban noise mapping", in *Proceedings of ACM/IEEE IPSN*, pp. 105–116, 2010.

[Cuervo et al. 10] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “MAUI: making smartphones last longer with code offload”, In *Proceedings of MobiSys*, pp. 49–62, 2010

[Das et al. 10] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: platform for remote sensing using smartphones", In *Proceedings of ACM MobiSys*, pp. 63–76, 2010

[Bruke et al. 06] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, “Participatory Sensing”, In *World Sensor Web Workshop (WSW’06) at ACM SenSys’06*, Boulder, Colorado, USA, October 2006

[Campbell et al. 08] A. T. Campbell, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, S. B. Eisenman, and G.-S. Ahn, “The Rise of People-Centric Sensing”, *IEEE Internet Computing*, pp. 12–21, July/August 2008

[Merriam-Webster] Merriam-Webster Dictionary, Crowdsourcing definition, “http://www.merriamwebster.com/dictionary/crowdsourcing”, accessed on May 2013

[Wikipedia App Store] Wikipedia on Apple Store, “http://en.wikipedia.org/wiki/App_Store_(iOS)” accessed on June 2013

[Wikipedia Play Store] Wikipedia on Play Store “http://en.wikipedia.org/wiki/Play_Store”, accessed on June 2013

VITA

Sam Honarvaralijani

Candidate for the Degree of

Master of Science

Thesis: MULTMEDIA ENABLED CROWDSOURCING APPLICATION FOR MOBILE DEVICES

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in department of Computer Science at Oklahoma State University, Stillwater, Oklahoma in July 2013.

Completed the requirements for the Bachelor of Science in Computer Science at Multimedia University, Cyberjaya, Selangor, Malaysia in 2011.

Experience:

Worked as a Software Engineer and System Support Engineer in several companies during 2005 through 2013 In Iran, Malaysia, and United States.

Professional Memberships:

Member of ACM Chapter at Oklahoma State University