Personal Emergency Preparedness Plan (PEPP) Facebook App: Using Cloud Computing, Mobile Technology, and Social Networking Services to Decompress Traditional Channels of Communication during Emergencies and Disasters

Melvin B. Greer, Jr.
Senior Fellow, SOA/Cloud Computing Chief Architect
Lockheed Martin Corporation
melvin.greer@lmco.com

John W. Ngo
Software Engineer, Technical Lead & Developer
Lockheed Martin Corporation
john.w.ngo@lmco.com

May 18, 2012

***Abstract*—Cloud Computing, Mobile Technology, and Social Networking Services such as Facebook and Twitter has become an integral part of society during the event of an emergency or disaster. In the wake and aftermath of a disaster, a tremendous number of people used social networking sites to post and share information. The Department of Health and Human Services (HHS) had sponsored a challenge for software application developers to design a Facebook application to help people prepare for emergencies and to obtain support from friends and families during its aftermath. Lockheed Martin had responded to this challenge by creating a cloud and mobile based Facebook application called the Personal Emergency Preparedness Plan (PEPP). This paper discusses the design and integration of the PEPP Facebook App with the intention of serving as reference architecture for developing social networking applications using cloud computing and mobile technology.**

Keywords- Cloud computing; Mobile; Wireless; Social Networking

I. Introduction

The trend towards cloud computing and wireless mobility can be distinguished on social networking sites such as Facebook. More than 800 million people use Facebook with more than half logging on in any given day. There are 350 million active users that access the site through their mobile phone while 7 million apps and websites are integrated with Facebook [1]. Cloud computing continues to be an enabler for social networking sites by allowing users to extend the site's platform and create apps that provide additional services.

As the number of users on Facebook continue to grow, so will the number of users who access the site through mobile devices on smart phones and tablet PCs. The more mobile devices that are out on the market, the more services need to be hosted on the cloud. Such services will be attracted to the cloud because they must be highly available. Especially since they rely on large data sets that are too large to be processed on hand held device. Therefore, services are most convenient hosted in large data centers [2]. This is often terms as the "Mobile Plus Cloud Computing" paradigm [3].

In response to this paradigm shift in the way online services will be delivered, cloud researchers and engineers must develop appropriate architectures and design patterns to ensure that clouds provide responsive, efficient, and feature-rich services to both native and web-based applications that run on mobile devices, desktop or laptop computers [4].

In this paper, we present the Facebook App, Personal Emergency Preparedness Plan (PEPP), as reference architecture for developing of cloud-based social networking applications that can be accessed using mobile devices or a native web browser. PEPP is designed to help individuals become prepared by allowing users to identify three "lifelines" to provide support during an emergency, to facilitate communication between friends and family during emergency situations, to provide a platform for collecting geographical information, pictures, and video, along with any other data that will help first response manage a crisis.

PEPP is an integrated Facebook App that can be installed by anyone who has signed up for a Facebook account. Once installed, the user will be able to go through a four-step process to becoming prepared: 1.) Identify Lifelines 2.) Create a Personal Preparedness Plan 3.) Send their plan to their lifelines, and 4.) Encourage other Facebook users to be prepared. In addition, the app allows users to receive notification alerts, manage lifelines, and send current location to lifelines.

Our experience with developing the PEPP has been quite positive because the cloud computing platform, the social networking services, and the associated development technologies have simplified the tasks needed to write, debug, and deploy applications. Using the U.S. Patented, Thundercloud™ architectural design pattern, we were able to separate private and public data to enhance security for users. In addition we have used the latest frameworks for development including Microsoft Windows Azure SDK, ASP .NET MVC 3, and the C# Facebook API. The deployment of our application in a cloud computing space made it beneficial for users who are accessing the application via mobile devices by offering 99.9% uptime availability.

In this paper, we discuss the design of our architecture, the reasoning behind our design choices, the strengths of the systems and the limitations that we found during our implementation effort. This discussion will be helpful to anyone trying to build cloud and mobile based applications with social network capabilities to address various mission focused scenarios.

The rest of this paper is organized as follows. Section II provides an overview of the PEPP Facebook App. Section III describes out the server-side components was implemented. Section IV describes the implementation of client-side components. Section V presents a list of future developments for the PEPP app. Finally, section VI presents the paper summary.

II. System Overview

A. Motivation

On August 22nd, 2011, the U.S. Department of Health and Human Services (HHS) and the Office of the Assistant for Secretary for Preparedness and Response (ASPR) had issued a challenge calling all developers, entrepreneurs, public health, and emergency response communities to develop a dynamic application that identifies and connects friends on Facebook who are willing to be "lifelines." HHS identifies a lifeline as someone who will act as the point-of-contact in the wake and aftermath of an emergency or disaster. The department claims that a tremendous number of people use Facebook to post and share information about those who are potentially affected by the disaster [5].

In addition to social networking sites that people leverage in the event of an emergency, people have also turned to apps built by Google such as Person Finder. It is an open source web application that provides a registry and message board for survivors, family, and loved ones that have been affected by a natural disaster. The application allows users to post and search for information about another person's wellbeing and location. The app has been deployed to several natural disaster zones immediately during the aftermath of recent incidents in Haiti, Japan, and Turkey.

Lockheed Martin had also developed an app called Open911 that utilizes mobile plus cloud computing to aid emergency response agencies in coordinating rescue operations with first responders. The application operates by allowing agencies to collaborate with local officials to share information such as maps, reports, pictures, and video. This information can be streamlined in real-time from a command and control center to hands of first responders with mobile devices.

The motivation behind the PEPP application would be to improve upon Google's Person Finder and Lockheed Martin's Open911 web application by incorporating a social networking platform such as Facebook. This would allow the application to leverage social networking capabilities to facilitate the communication between friends and family. Moreover, the PEPP app integrates features that would provide critical information for emergency responders as found in Person Finder and Open911 such as geographical information systems.

In the aftermath of a disaster, traditional channels of communication such as cell phone and land line networks are frequently overwhelmed. Although, SMS messaging has been an invaluable source of communication that has been proven reliable in past disaster relief efforts, it does not connect multiple response personnel to those that are affected. Therefore, integrating with social networking sites such as Facebook and Twitter will decompress those channels of communication.

B. System Architecture

Designing the architecture of the PEPP app requires understanding how the Facebook platform works. This includes becoming familiar with the core features that are available for integration such as news feeds, notifications, platform dialogs, and the social graph. The Facebook website has a section for developers that refer to these core concepts in addition to tools and software developer kits (SDKs).

While reviewing the core concepts, we have discovered that the Facebook platform does not provide a hosting service for apps. Therefore, a developer who is creating a Facebook app must find their own hosting service. After subscribing to the hosting service and registering the app's URL, the next step is to sign up for a Facebook Developer account. Following the registration process, the developer will have to enter the app's URL in the Facebook App settings page. This will allow Facebook to pull the app into the page canvas. Specifically the app will be loaded within an HTML **`<iframe>`** element. The architecture of the PEPP Facebook app is depicted in Figure 1.

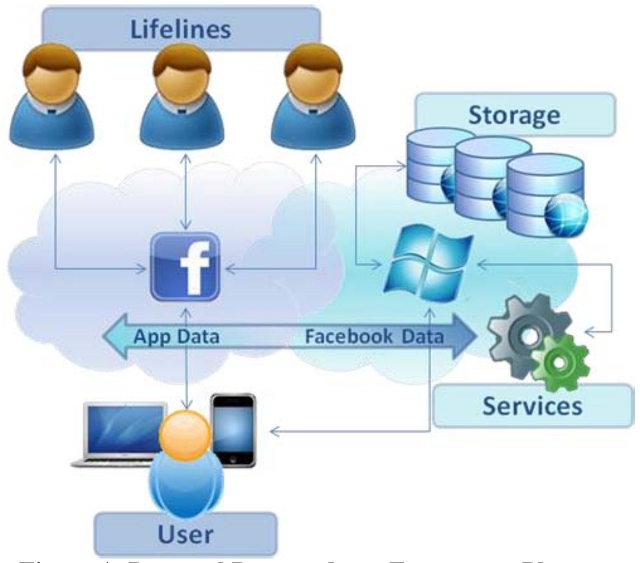


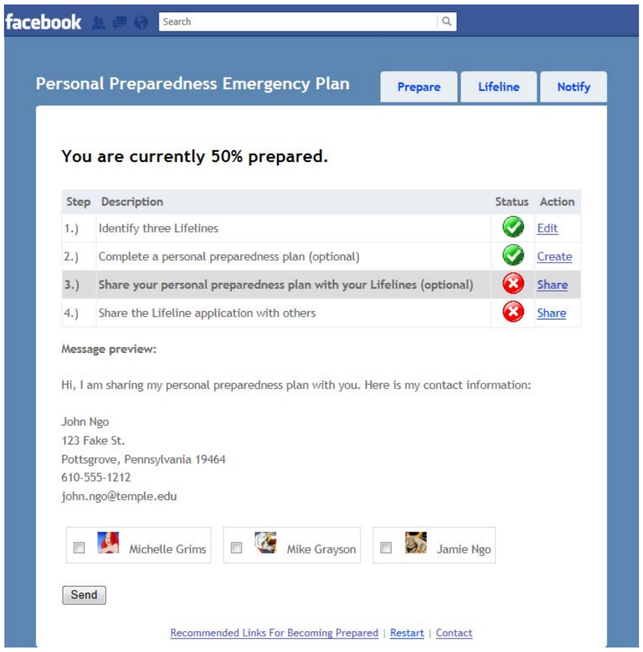
Figure 1: Personal Preparedness Emergency Plan: Facebook Application System Architecture

Figure 1 shows a high level architectural overview of the PEPP app. It shows two interconnected *hybrid* clouds consisting of the Facebook platform on the left and the Microsoft Azure on the right. The composition of the clouds remain unique entities and is joined together by configuration settings that enable data sharing between the cloud entities.

Although the app could have been hosted by any service provider, Microsoft Azure was selected because it offers a highly scalable and available cloud computing environment. This will ensure the app can accommodate the large number of users on Facebook. In addition, the Azure platform can accommodate unexpected spikes of usage.

C. *Web Interface*

The web interface for the PEPP application appears as a native Facebook app. The app is integrated into the Facebook canvas while leveraging core features of the platform such as notifications, authorization dialogs, and the social graph.



Personal Preparedness Emergency Plan

You are currently 50% prepared.

Step	Description	Status	Action
1.)	Identify three Lifelines	✓	Edit
2.)	Complete a personal preparedness plan (optional)	✓	Create
3.)	Share your personal preparedness plan with your Lifelines (optional)	✗	Share
4.)	Share the Lifeline application with others	✗	Share

Message preview:

Hi, I am sharing my personal preparedness plan with you. Here is my contact information:

John Ngo
123 Fake St.
Pottsgrove, Pennsylvania 19464
610-555-1212
john.ngo@temple.edu

Michelle Grims Mike Grayson Jamie Ngo

Send

Recommended Links For Becoming Prepared | Restart | Contact

Figure 2: PEPP App User Interface

The app was created using the following technologies:

- **Visual Studio 2010** - Tightly integrated with the cloud. IDE includes tools to ensure quality modeling, coding, and testing.
- **Windows Azure** - Provides highly available and scalable hosting and storage in the cloud. Allows users to access to data and services from anywhere, anytime. Provides elasticity and load balancing for growing users of the app. Multiple virtual instances can be easily spun up to meet the demands for the app.
- **ASP.NET MVC 3** - Built using the model-view-controller architectural design pattern. Decouples models and views to reduce complexity in design. Increases flexibility and maintainability of code.
- **Facebook C# SDK** - Enables server side invocations using C# to call Facebook API web services.
- **jQuery** - Enables client side invocations using Javascript to call Facebook API web services.
- **HTML 5** - Used for geolocating users of the app. Improves richer web applications on mobile devices while enabling cross-mobile reusability and consistency across various smart phone devices. This will allow us to easily extend this app to a mobile device in future efforts.

D. *iPhone / iPad Interface*

Current the PEPP app works best on wireless devices such as laptops, mobile devices, and tablet PCs. The app makes use of Wi-Fi and HTML 5 Geolocation APIs to detect the user's location. Figure 3 shows the PEPP app using geolocation on the Apple iPad.

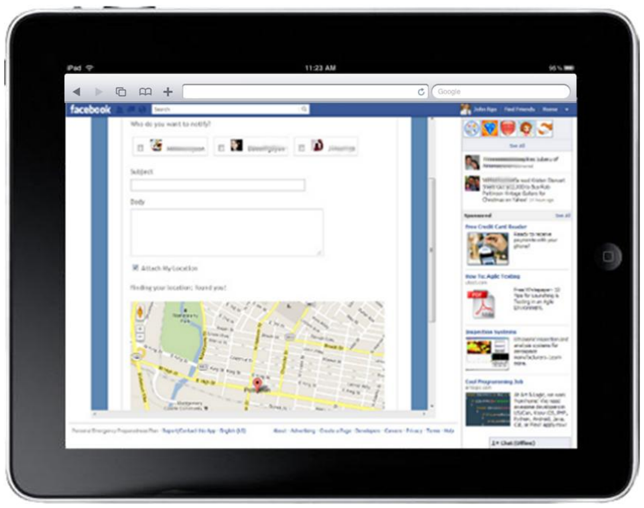


Figure 3: Open911 iPhone interface

The utilization of HTML 5 optimizes mobile web applications by standardizing common use-cases and technologies that are common in smart phone devices. This reduces the need for proprietary markup languages and APIs which allows the application to be portable. Some large companies are starting to realize the need for standards on smart phone devices. For example, Adobe had recently announced they are stopping all development for Flash Player browsers in favor of focusing their investments in HTML 5 [6].

E. *Application Data Model*

The PEPP app needs to store data in a way to represent the relationships between users and their lifelines. Additionally, the notification messages are stored and minimal information about the user. The data is stored using Azure Table Storage (ATS). Although this type of storage is not a relational database, it offers high performance, efficient retrieval, and simplistic scalability. Using ATS, instead of creating a database, all the user has to do is create a class that represents the data model and Azure will build the containers for that model [7]. An example of how data is structured in Azure using containers (or entities) is represented in figure 4.

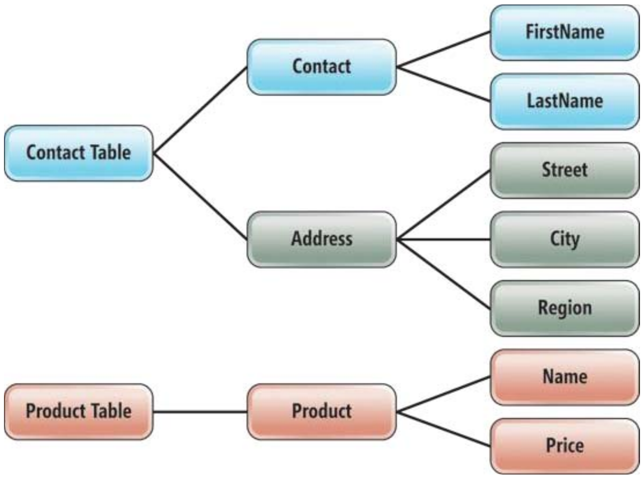


Figure 4: A Single Windows Azure Table Can Contain Rows Representing Similar or Different Entities [7]

The above representation is an example on how data is structured in Windows Azure. For data to be created properly, the classes must define properties and inherit the TableServiceEntity class. It is also recommended that properties such as the Partition Key and Row Key are set for efficient querying, organization, and scaling. The code below is an implementation of the lifeline data model.

```
/* The Lifeline data model represents the
relationships and status between users and their
lifelines. Lifeline status code can be 0:
Accepted, 1: Rejected, 2: Pending */
public class Lifeline : TableServiceEntity
{

  public Lifeline()
  : this(Guid.NewGuid().ToString(), string.Empty)
  {
  }

  public Lifeline(string partitionKey, string
  rowKey)
  : base(partitionKey, rowKey)
  {
  }

  public Lifeline(long UserId, long lifelineId,
  LifelineStatus lifelineStatus)
  : this(Guid.NewGuid().ToString(), String.Empty)
  {
    this.UserId = UserId;
    this.LifelineId = lifelineId;
    this.LifelineStatusCode = (int)lifelineStatus;
  }

  public long UserId { get; set; }

  public long LifelineId { get; set; }

  public int LifelineStatusCode { get; set; }

}
```

Table 1: The Lifeline data model for the MVC pattern. In addition, this class defines how data is stored in Azure Table Storage.

III. IMPLEMENTING SERVER-SIDE FUNCTIONALITY

A. *Cloud Application Server*

Since Facebook does not provide hosting, we have selected Microsoft Windows Azure to host the PEPP app. Azure offers a platform as a service (PaaS) in the cloud that is hosted in Microsoft datacenters. The environment is a cloud operating system that serves as a runtime for applications. Developers can utilize a set of development services, management, and host applications on or off premises. The Azure cloud will be the entry zone for app requests from web browser requests. It delivers both dynamic and static content, thus acting also as an application server.

1) Server Organization

The web role stack is configured with IIS7, which simplifies the process of creating applications using popular web technologies such as ASP.NET and PHP. For the PEPP app, we have leveraged these technologies and took it a step further by developing the app using the ASP.NET MVC 3 Framework. This application framework implements the model-view-controller (MVC) pattern. **Error! Reference source not found.** depicts the organization of the PEPP app in the Windows Azure web role.

The web services for the PEPP app can be accessed using either XML or JSON. Using ASP.NET MVC, we have defined a set of routing rules and controllers that allows app to use read data using the HTTP GET verb or write data using the POST messages. Depending on how these actions are defined in the controller, the method will either return a view such as a Home page or a view containing the data model such as a User.

Figure 5 depicts how request data is handled in Azure and MVC. Based on this model, the When the user invokes a service, for example, GetLifelines, the request will either be in XML or JSON format. Support for JSON is desired because it works best with jQuery's AJAX calls while giving the capability of serializing data into native objects on the client side or the service side. The Azure Compute component has a *web role* defined that is pre-configured with IIS7. This acts as the web server to handle RESTful requests from the consumer. After the web role receives the request, it is then passed to the application's routing system which consists of a set of routing rules. This will map incoming URLs to the application and route them so that the correct *Controller* and *Action* method can process further instructions. Instructions may involve the data model, for example—getting all lifelines for a particular user. The model represents the data that is persisted in Azure Storage Tables. After the data is retrieved from the cloud storage and set in the model (e.g. see Table 1). Finally, the controller instructs the View (a dynamic web page) to retrieve the model and redirect the View back to the Controller which is then returned for the user to see.

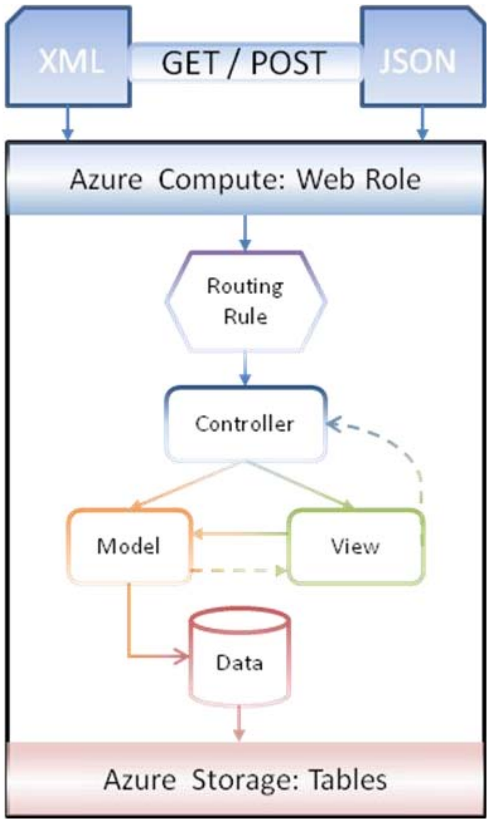


Figure 5: Data flow of a HTTP Request in Windows Azure and ASP.NET MVC

IV. IMPLEMENTING CLIENT-SIDE FUNCTIONALITY

A. The View Engine

One of the new features of ASP.NET MVC 3 framework is the addition of the "Razor" view engine. The engine is focused on the template approach for generating dynamic web pages with clean separation of code. The expressive syntax is easy to learn and makes code easy to read through its compactness and fluidness. Using this engine makes designing web pages much faster since it requires less keystroke and offers a great *IntelliSense*. Additionally, it supports the creation of HTML 5 enabled project templates.

2) Client Organization

The PEPP app client web interface is designed so that it matches the theme of Facebook which gives it an integrated look-and-feel. When the app is loaded into the user's Facebook canvas, the user arrives at the home landing page and is presented by a wizard. The wizard will step the user through the completion of being prepared. Other features include the ability manage lifelines, e-mail messages (via Google's free SMTP server) that pinpoints a user's location on Google maps, and posting messages to a user's wall.

V. FUTURE EFFORTS WITH THE PEPP APP

Additional features that can that can give the PEPP app more added-value is the integration into other social networking twitter such as Twitter. This includes integrating the app with SMS messaging since it has proven to be a reliable form of communication in past disaster recovery efforts.

Tighter integration with mobile devices and maximizing its Wi-Fi capabilities can give the app greater use case scenarios. For example, we can utilize a smart phone's geolocation capabilities and turn the PEPP app into a beacon that announces the user's location if it is ever activated in an emergency.

VI. SUMMARY

As the trend of cloud computing, mobile technology, and social networking continues to grow, architects and developers can now build new communication solutions for the general public and government. In this paper, we present a case study and reference architecture for building applications that integrate these technologies. We have created a Facebook application that can help users prepare for an emergency and disaster. Users of this application will need nothing more than a smart phone and a social networking account. The application integrates mobile, cloud, and social networking technologies to facilitate communication in an event of an emergency or disaster, where traditional channels may not be available.

REFERENCES

[1] Facebook, "Statistics" http://www.facebook.com/press/info.php?statistics, Access date: December 20, 2011.

[2] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., …Zaharia, M., "Above the Clouds: A Berkeley View of Cloud Computing". Electrical Engineering and Computer Sciences University of California at Berkeley, UCB/EECS-2009-28, February 10 2009.

[3] Microsoft Research, "Mobile Plus Cloud Computing" http://research.microsoft.com/en-us/collaboration/global/asia-pacific/programs/mobile-cloud.aspx Access date: December 20, 2011.

[4] Rodriguez-Martinez, M., Seguel, J., Sotomayer, M., Aleman, J., Rivera, J., Greer, M., "Open911: Experiences with the Mobile Plus Cloud Paradigm". University of Puerto Rico, 2011.

[5] Challenge.gov, "The ASPR Lifeline Facebook Application Challenge" http://challenge.gov/HHS/220-the-aspr-lifeline-facebook-application-challenge, Access date: December 21, 2011

[6] Vaughan-Nicholas, S., "Flash is dead. Long live HTML 5" http://www.zdnet.com/blog/networking/flash-is-dead-long-live-html5/1633, 9 Nov 2011. ZDNet. Retrieved: 30 Dec 2011

[7] Lerman, J., "Windows Azure Table Storage – Not Your Father's Database" http://msdn.microsoft.com/en-us/magazine/ff796231.aspx. MSDN Magazine. Retrieved: 30 Dec 2011