Developing a Unified Messaging System

Michael Walden
Advisor: Avi Silberschatz
CPSC 490

April 30, 2012
Abstract

The purpose of this project was to develop a system that would improve communication. We now have the ability to text, send e-mails, instant message, use social networks, VOIP, and etc. However, considering that there are so many ways to communicate there isn’t a system or application that is explicitly designed to allow you to determine how you want to receive messages when you are busy. So in this project I focused on developing a system that will allow users to indicate how they want to receive messages when they are offline.

To complete this project I developed a mobile website that is hosted on dotCloud, uses jQuery Mobile, HTML, PHP, MySQL, TextMarks Lite, the Facebook registration plug-in, and a Gmail account as a makeshift SMTP server. The registration process makes use of the Facebook Registration plug-in, and asks for a user to indicate their name, email address, a password, cell phone number, mobile provider, and their preference for how they’d like to be contacted when offline. The interface for the website was designed primarily using jQuery Mobile. It allows a user to maintain a contact list of people who have given them permission, instant message their online contacts, send offline messages, view message history, and update preferences. Specifically, it allows you to send messages to any of your contacts that are offline via text messaging, email, or both depending on their preferences for receiving messages when they are offline. Additionally, the site utilizes TextMarks and the keyword UMSAPP, to allow users to quickly update their preferences offline.
Original Design vs. Current Design

The original design for the system as depicted here was focused on allowing users to use the UMS website from their desktop, laptop, or mobile device. It would allow a user to chat with friends who are online or send a message to an offline contact via e-mail, text, or VOIP. The UMS server determines how to best contact recipients based on their preferences and delivers the message accordingly. Additionally, the original design specified that the Unified Messaging System would have the ability to handle attachments and deliver them in the appropriate manner.

The actual design of this project generally follows the original design depicted below, but doesn’t have the ability to handle attachments, and doesn’t include VOIP as an option for receiving messages. The actual UMS server determines how to best contact recipients and delivers the messages using the combination of PHP and MySQL.
Implementation

Web Host: dotCloud

The UMS website is hosted on dotCloud, under the url: [http://umsapp-umsadmin.dotcloud.com/](http://umsapp-umsadmin.dotcloud.com/). dotCloud is a cloud web host that allows you to implement two services on their platform for free. The UMS website utilizes PHP and MySQL as the two free services. So the front-end and the back-end of the website are hosted all on one platform.

MySQL Database

The UMS website consists of five tables in its back-end database. The tables include Users (which holds all of the account information), Contacts (which maintains the contact relationships), Mobile Providers (which is a table that consists of all mobile providers and their SMTP protocol), MessageHistory (which records information about every message that was sent using UMS), and InstantMessages (which records information for the instant messaging component of the site).

Registration

The UMS system requires its users set up accounts on its website. The registration process uses the Facebook registration plug-in, to expedite the process. To use the Facebook API associated with the registration plug-in the website had to be registered as an application under a Facebook account. Benefits to registering UMS with Facebook Developer and using its registration plug-in include the ability to access user analytics on the usage of the UMS website, and also because of the easy validation feature included in the plug-in. The registration plug-in used for UMS has the ability to pre-fill the registration form for Facebook users with some of their account information, and also allows non-Facebook users to register as well. The registration process requires that a user indicate their name, email address, password, cell phone number, mobile provider, and their preference for receiving offline messages (via text messaging, email, or both). The mobile provider of a person’s cell phone service is necessary to determine the SMTP protocol for sending text messages from the UMS server. The email address of a user serves as the unique identifier, which means that an email address can only be associated with one UMS account. Moreover, if someone tries to register with an email account that is already associated with an account the registration will fail and the person will be notified of the error. The registration plug-in passes the data to the UMS server as a signed request, with the signature being the Application Secret set up when registering UMS.
with Facebook. The UMS server reads the signed request, which is stored in a JSON object using PHP. Facebook actually provides a sample of PHP code that demonstrates how to read the signed request, and this was adapted to fit the needs for UMS.

The remainder of the website utilizes jQuery Mobile for its interface, PHP for the server-side scripting, and MySQL as the back-end database. jQuery Mobile was used to make the UMS website easily accessible and user-friendly on laptops, desktops, and mobile devices. The interface is relatively simple and consists of the following jQuery Mobile features: a header, a navbar with tabs (for Your Contacts, Contact Requests, Online Contacts, Contacts, Message, and Preferences), internal pages, buttons, forms, listviews, and control groups.

Your Contacts

After login a user’s online status in the User table is automatically changed from “Busy” to “Online”, using PHP and SQL. After logging out, a user’s online status is automatically changed from “Online” to “Busy”. After login the user is automatically directed to the “Your Contacts” page. In order to send someone a message using UMS, you must have permission from the desired recipient. Thus each user will have a list of contacts that they have permission to contact, and who have permission to contact him or her. The contacts page consists of a list of contacts and their contact information including cell phone number and email address. A person’s contacts are found using PHP and SQL to query a join between the contacts table and the users table. The results are then stored in a jQuery listview. The contacts page also makes use of a jQuery button, which can be clicked to add a contact. This button initiates a process that uses PHP to query the database and find all users who are not currently listed as contacts with the current
user. The user is then able to select one of these people, and send a contact request to them.

![Contact Requests](image)

**Contact Requests**

As described above, to add another user as a contact permission must be granted by that person. Upon the sending of a contact request, a record is added to the Contacts table for this potential relationship with the status of “pending.” Upon approval, the status of the record is updated and set to “approved.” Upon rejection, the record is deleted from the Contacts table. A user’s contact requests can be viewed on the Contact Requests Page, which uses PHP and SQL to find all of the records in the Contacts table in which the current user is in, and where the status is currently “pending”. The requests are then stored in a jQuery listview, and display the user it was sent from along with their email address and phone number.
Online Contacts/Chat

Next, there is a page designated for Online Contacts. On this page, a user can view all of their contacts who are online, and initiate a chat session if desired. The process for initiating a chat session has 3 components. First, AJAX script queries the database to get all of the user’s online contacts, then once a user has decided to initiate a chat session with someone AJAX script runs queries in a PHP file on the database to both send and receive messages. All of the contacts are viewable in a jQuery listview that is generated dynamically with JavaScript and the results of the AJAX call. Each of the list items corresponding to an online contact have click events that are bound with opening a “send instant message” dialog to that respective contact. After sending a message using this dialog, the user is then directed to a chat room where they will be able to receive messages from any of their online contacts or send messages to any of their online contacts. This chat room has a jQuery listview that is dynamically generated by AJAX making a call to a PHP server side file that runs a query, which returns all of the unread instant messages that have been sent to the user. The listview itself is updated on a set interval using this process described above to avoid having to refresh the entire page to receive more messages. Additionally, after a user sends a message it is automatically appended to the listview containing instant messages in their chat session. To reply to a message, a user only has to click on the message and the “send instant message” dialog will appear.

Messages

The next page is the Messages page, which includes Received Messages, Sent Messages, and a link for sending messages to offline users. A user can view a history of messages they have sent and received on this page.
Specifically, two separate jQuery Mobile collapsible sets with inset lists were used to display received messages and sent messages. To retrieve these messages a PHP file runs a query on the MessageHistory table and returns all of the records in that table for the current user. The content of the message is not displayed on this page, only information about the message including sender, recipient, delivery method, and the time at which the message was sent. The most important feature of the Messages page is the “Send A Message” button, which initiates the process for sending messages to an offline contact. Immediately after clicking this button a jQuery Mobile dialog appears that allows the user to select a recipient who is currently offline. After selecting a recipient, the user is redirected to another dialog that allows the user to enter their message and then send it. The UMS server then determines the recipient’s preference for receiving messages when offline using PHP script to run a query on the database. Next, the server will deliver the message based on this preference. If the preference is email the server will deliver the message to the recipient’s email address, if it is text messaging the server will retrieve the SMTP protocol for the recipient’s mobile provider and then send the message. If the recipient’s preference is both, the server will perform both of the aforementioned functions. Because there is a limit on the length of text messages, if a message is longer than 110 characters (note: some additional standard text is always sent) and a user’s preference is text messaging or both, then the server sends a short text message to the user informing them that they have received a long message and it that has been delivered to their email. Because dotCloud only allows two free services, a Gmail account (umsapplication@gmail.com) was used as the SMTP server for sending messages to user. In order to configure the UMS application and dotCloud for using the Gmail account as the SMTP server the Gmail SMTP server name, the port number, and the username and password of the account were necessary and added to the dotCloud Build File.
Preferences

The last page, is the Preferences page which allows a user to update his or her preference for receiving messages when offline. This is done using a form and a PHP file that gets and updates a user’s preference in the Users table. The most important feature related to preferences, is the ability to update your preference when you are on the go via text messaging. TextMarks Lite is used as a SMS gateway, to allow users to communicate with the UMS server from their cell phones. To use this service, the keyword UMSAPP was registered with TextMarks. TextMarks allows users to send this keyword to the number 41411, and it then allows the administrator to designate responses based on server-side script created by the administrator. For this project, the initial message sent to a user after texting UMSAPP to 41411, is “To set your preference for receiving messages offline rply with 1 for ‘Txt Msg’, 2 for ‘Email’, or 3 for Both.” Based on the user’s response, the UMS server will update that user’s preference for receiving messages offline. The server is able to identify the user by using TextMarks to obtain the phone number from which the text was sent, and then verifying that this number was used in one of the UMS accounts. In general this feature addresses the issue of determining what a user’s availability is when they are offline because it allows them to freely indicate or change how they want to receive messages without even having to log on to the UMS site.
Where To Go From Here

First off, given more time the most obvious upgrades to UMS include adding all of the features that were in the original design, but not include in this design. Specifically, adding the feature for handling attachments and delivering them is obligatory as sending attachments is a very, very common task. Next, the other feature that was left out from the original design was the ability to deliver messages via VOIP. I would also shore up the security of the system to safeguard against attacks like SQL injection.

As far as additional features, it is definitely vital to add the ability to send one message to multiple contacts at the same time just like you can with text messages or emails. Also because the Facebook Registration Plug-in was used, it would be useful to have UMS automatically add all of a registrant’s Facebook friends who use UMS to their contact list.

Another extension is sort of related to the concept behind Foursquare. A mobile app for UMS could be developed, and it could allow a user to create a list of preferences for receiving messages, based on a common set of places that they visit. So for example, the app could record the location of places like home, work, church, and etc. and allow the user to designate a different preference for receiving messages for each of these places. The app would then automatically recognize when a user is in one of these places, and then update their preference for receiving messages accordingly.

Conclusion

Developing this Unified Messaging System was a very insightful project and taught me just how much effort and hard work was and is put into maintaining social networks like Facebook. This was my first foray into completing a web project that consisted of so many different moving parts including PHP, MySQL, AJAX, jQuery Mobile, a SMTP server, an SMS gateway, and dotCloud. I’d had no formal introduction to some of these platforms and so completing this project gave me some nice experience in these areas.
Resources

- dotCloud - https://www.dotcloud.com/
- PHP
- MySQL
- TextMarks Lite - http://lite.textmarks.com/
- Gmail
- Facebook Developer
- jQuery Mobile Framework
- Build Your Own Database Driven Website Using PHP & MySQL 4th Edition by Kevin Yank