**Mobile Web ohmage Client**

Application Overview

Last Updated June 28, 2012

*Principle Authors*  
Khalapyan, Zorayr <zkhalapyan@oit.ucla.edu>

Tangmunarunkit, Hongsuda <hongsudat@gmail.com>

Selsky, Joshua <selsky@cens.ucla.edu>

*Editing Authors*   
Sakabu, Edward <edsakabu@ucla.edu>  
Rocchio, Rose <rrocchio@oit.ucla.edu>

Estrin, Deborah<>

Table of Contents

[I. Introduction 3](#_Toc329038159)

[I.1. Abstract 3](#_Toc329038160)

[I.2 Background 3](#_Toc329038161)

[I.3 Overview 3](#_Toc329038162)

[II. Architecture and Design 4](#_Toc329038163)

[II.1 Client Design 4](#_Toc329038164)

[II.2 Presentation Layer 4](#_Toc329038165)

[II.3 Back End Exposed Services 5](#_Toc329038166)

[II.4 PhoneGap Integration 5](#_Toc329038167)

[III. Conclusion 5](#_Toc329038168)

[III.1 Hybrid Approach Advantages 5](#_Toc329038169)

[III.2 Hybrid Disadvantages 5](#_Toc329038170)

[III.3 Future Direction 6](#_Toc329038171)

[III.4 Screenshots 6](#_Toc329038172)

# 

# I. Introduction

## I.1. Abstract

Advances in mobile devices and web technologies have enabled mobile users to observe, collect, and publish personal or community-related data to the cloud anytime and anywhere. At the same time the increase in diversity of mobile devices has put a high cost on developing robust and powerful mobile applications that are device-agnostic and easily accessible from any multiple devices.

In this paper, we describe our web-based mobile application, Mobile Web ohmage Client (MWoC) that enables users to record in-situ data entered by the user anywhere and anytime using Mobile Web Framework (MWF) and PhoneGap integration. The data will be uploaded and published to ohmage---an open-source, mobile to web platform that records, analyzes, and visualize both human-in-the-loop, and continuous data streams passively collected from mobile devices. The final product is a lightweight, single-source and platform-independent mobile application and can be run on a broader platforms compared to the more comprehensive ohmage native Android application. MWoC utilizes the Mobile Web Framework (MWF) for a robust cross-platform presentation and PhoneGap container for accessing native device features.

We will present our requirements, design and system architecture, and will conclude with general thoughts on building hybrid mobile applications as well as discuss future direction of the MWoC.

## I.2 Background

The increasing ubiquity of smartphones enable mobile devices to be used as tools to observe and understand the patterns, habits, and environments of individuals and communities [] in hopes of helping doctors and researchers to gather and analyze data. To address ohmage is an open-source, mobile to web platform that records, analyze, and visualize data from distributed mobile devices. The platform supports two data stream types: (1) prompted experience samples which are small scriptable survey forms for users to enter data in the field, i.e., in-the-moment experience; and (2) continuous streams of data passively collected from sensors or applications on-board the mobile device (e.g., location traces).

## I.3 Overview

Ohmage users can create a data campaign which is a collection of surveys to be answered by participants. Newly defined campaigns are uploaded via a web browser and automatically become available for participation. To participate, a user logs in via the ohmage client on the phone and downloads the particular campaign(s) in which they will participate. Surveys can be initiated by the user or prompted via time/location reminders. All submitted survey responses, as well as passive continuous data streams are automatically time-stamped, geocoded and uploaded to the ohmage server for further analysis and visualization. Via web browsers, users are able to access and visualize collected data and their analyses in real-time. Ohmage has been used in multiple behavioral research studies[], as a pedagogical tool in high schools[], and as a journaling mechanism to document about communities.

**I.4 General Requirements**

The first version of MWoC is designed to be light-weighted and focuses only on limited features which are user authentication, data capture of prompted experience samples and a simple time-based reminder described below.

**1. Authentication:** Once a user logs in, MWoC authenticates the user with the ohmage server and upon successful authentication, stores a long-term token locally for subsequent communication to the server without prompting the user for a rentering the password. All user data including the token are deleted when the user logs out or changes the password.

**2. Data Capture:** Users interact with MWoC to browse through a set of available campaigns and download specific campaigns that they would like to participate. The downloaded campaigns are stored locally on the phone for better responsiveness and for offline support. To complete a survey, MWoC goes through the campaign definition (written in XML, but converted to JSON on the client side) and renders appropriate sequence of prompts (i.e. question) based on their types (e.g. menus, number, time, free text, image, etc.) and conditions. Upon survey completion, all survey responses are time-stamped, geocoded, and uploaded to the ohmage server. In the case of network outage, it will store survey responses locally on the phone and will try to automatically upload the data upon future launching of MWoC. Users can also perform manual upload of the data.

**3. Time-Based Reminders:** MWoC allows users to configure time-based reminders to be triggered during specific time and condition (e.g. send a reminder to complete a survey unless the survey has been completed within the last 2 hours). Upon reaching triggered criteria a notification in the form of ring tone or vibration will be generated to remind users to complete their survey(s).

# II. Architecture and Design

## II.1 Client Design

The front end business logic of the application has been built using JavaScript that interconnects with the service layer with AJAX communication. The initial campaign configuration is received in XML format and later converted into JSON and stored in *localStorage*. Furthermore, for greater flexibility, reusability, and manageability, each organizational unit such as list of campaigns, individual campaign, a single survey, etc. has been modeled with a JavaScript class that provides certain abstracted functionality and allows avoiding use of raw data.

Data that requires persistent storage space across multiple pages is stored in HTML5 localStorage. When offline, the user’s responses are stored in this storage and later uploaded when Internet connectivity is available.

In order to be fully offline compatible, all JavaScript, HTML, and CSS files have been copied locally inside the PhoneGap integration. With this technology, the user will be able to fully interact with the application even if the device is completely offline.

## II.2 Presentation Layer

Mobile Web Framework’s presentation functionality has been utilized to create a device agnostic user interface, compatible with most mobile browsers. To support MWF in an offline environment, the CSS and JS files provided by the framework have been extracted for individual phones and are stored locally, on the device.

MWF release version 1.2 that was used for the development supports menu lists, buttons, double buttons, page headers and footers along with other mobile optimized UI components. Since generally, MWF entities are generated by either server-side technology such PHP or ASP.NET or are statically programmed, a JavaScript library was built modeling the decorator pattern that facilitates dynamic creation of elements. Utilizing the library, it was much easier to programmatically create DOM elements that resemble the MWF structure and hence, allow more flexible models for rendering prompts, list of surveys, and other views required for the user interface.

## II.3 Back End Exposed Services

The ohmage back end is a Java based REST server that handles authentication, data processing, and data analysis. Communication is based on the JSON data model, but large campaign configurations files are still based on XML. Once authenticated, the user stores a hashed token that is sent with future requests.

## II.4 PhoneGap Integration

PhoneGap is an HTML5 application platform that allows web based applications to interact with the mobile device’s native operating system. Primarily, the platform allows JavaScript running on the client device to access the device’s camera, file system, GPS, accelerometer and several other devices capabilities through a common abstracted JavaScript based API layer.

To deploy an application based on the platform, PhoneGap provides project source code for different devices that are ready to be deployed once integrated with the HTML/JS/CSS files. For example, to deploy an iOS application, PhoneGap would provide an Xcode project model that is configured with the PhoneGap environment and is ready to be deployed on an iPhone.

# III. Conclusion

## III.1 Hybrid Approach Advantages

Using the hybrid approach enabled single source management with native application deployment on several devices. With JavaScript as the underlying scripting language for implementing the business logic coupled with the device agnostic presentation layer supported by MWF, the application provided similar look and feel on almost all devices that were tested.

Furthermore, utilizing PhoneGap’s API for accessing the native device capabilities was fairly flexible and easy to use. We were able to provide seamless image capture capability without coding platform depending programming language – this allowed us to spend more time concentrating on writing the business logic and less time one finding differences between devices.

## III.2 Hybrid Disadvantages

Unfortunately, running web pages within native containers are noticeably less responsive than actual native applications. This is partly due the touch event handling on certain devices that add about a 300ms delay before invoking the ‘click’ event handler on DOM objects­­­­­. To decrease this delay, instead of attaching events to click events, all user interactions should be attached to the touch events such as onTouchUp and onTouchDown. Although this ads overhead to the source code, it produces much more responsive UI.

Another major issue with using the hybrid approach is lack of functionality for supporting background processes. This problem originates from two issues: mutual shared access between native application and the JavaScript application, and lack of support for background running processes by various devices.

With the current implementation of PhoneGap there is no atomic operation for locking data that is shared between the native code and the client side application. Therefore, it is not possible to provide mutual-exclusive shared access to a data structure while running background processes. The second issue is with background process restrictions by the iOS platform.

## III.3 Future Direction

Ideally, all data upload should be done automatically via background processes running seamlessly to the user. Unfortunately, this functionality is not currently supported due to restrictions on background processes by the iOS platform. We will explore other mechanisms (e.g. piggybacking on existing background events) to perform automatic data upload in the future releases. We also plan to extend MWoC to support visualization of survey responses and basic data analysis as we find this will be helpful for users to interact with their personal data. The notification/reminder feature can also be expanded to notify based on geolocation and context.

As for deployment, MWoC is an alternative to its ohmage android counterpart suited for studies that focuses on gathering experienced sample data. MWoC will be deployed in a behavioral research study of health and HIV-transmission risk behaviors of gay men and used as an option in high school students for data collection and analysis lessons [4].

## III.4 Screenshots

Below are screenshots from the application.



























