QR Conference Capture Mobile App

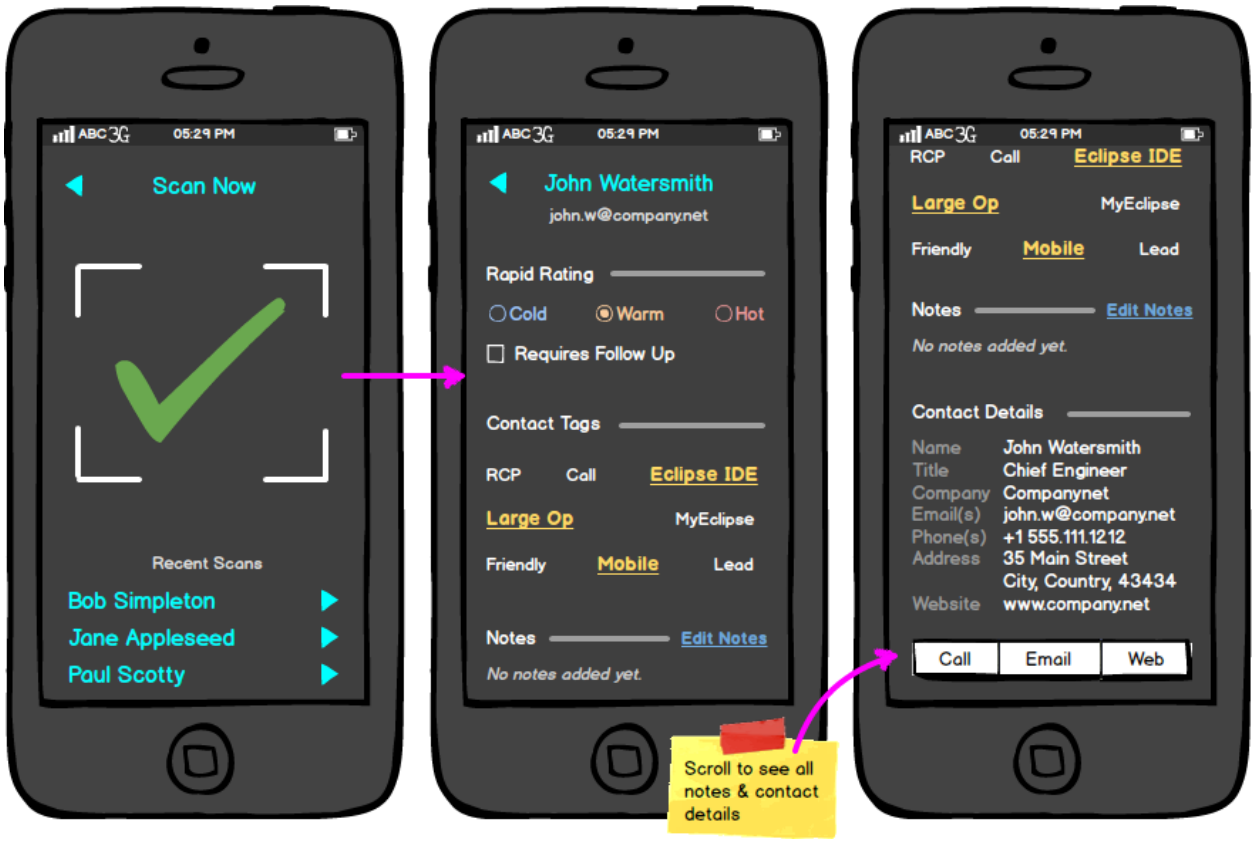
Abstract

Create a small PhoneGap-based HTML5 Mobile application, leveraging offline device storage, to allow QR-Code based capture of conference attendees and then allow adding quick tags or rating the attendee by opportunity size. The app will need to work with a backend web service that allows multiple employees from a company to register attendees collaboratively.

1. Front-End Application Requirements

1.1. Overview

A mobile application will be created that allows capture of conference attendees by scanning the QR codes on the backs of their badges.



The most basic part of the flow is the capture of QR codes. The person scanning the badge should be able to provide additional details on the user, such as the attendee rating, and additional tags and notes on the attendee.

1.2. Example Data

The following QR codes are from an actual conference and can be used for testing the QR code scanning capabilities. *The contacts from these QR codes should not be contacted.*

The QR codes supported should be handled in both vCard and meCard formats, that said, the first conference that this app will be used at will be in vCard QR code format. We should handle all fields that the vCard or meCard supports, but only in embedded form -- we do not need to support QR codes that let you download a vCard from the web.

1.3. Principal Usage Flow

During an average day at the conference, users of the application will scan the QR codes of conference attendees from their badges. During discussion with conference attendees the user will flag data about the attendee including selecting custom tags from a tag cloud, and choosing the hotness of the lead. A user may return to the booth so the application should allow modification of previous scans to augment further information on a given user.

Due to the sporadic nature of internet connectivity at conventions, we will have the application support queuing of scans, and in the background as the app is in use, the app should try to upload scans to the server to allow for lead reviewing and exporting.

The application will receive scans from other users when connectivity is restored and allow perusing of the attendees. In best cases, the user will be able to add data to scans from others from their company, but this is not a hard requirement.

1.4. Project Timeline

The project needs to be completed for initial testing by late February, and be approved and available in the Apple App Store and Google Play no later than March 6th. *This is a hard external time commitment that can not be missed.*

1.5. Implementation Constraints

The technology stack used will require usage of PhoneGap to build the offline-capable HTML5 application, with deployment to both Android and iOS. *The application will be made open source, and can use libraries that are freely available and distributable, such as with jQuery Mobile.* The application should be written using clean coding styles aiming for code simplicity where possible.

The backend of the application will be implemented in Java EE to expose REST-style web services, and the implementation of which will be done by someone at Genuitec. For rapid development purposes, a small PHP backend should be implemented to allow easier testing of early versions of the application, however, the main implementation will be in Java EE and eventually open sourced.

1.6. Data Capture

The data to be captured for each contact include:

- All data from the vCard or MECard
- The Rapid Rating for the contact (Hot, Warm, Cold)
- Whether or not a follow-up is needed
- Tags associated with the contact, customized for each conference
- The employee(s) that scanned the contact
- The time that the contact was scanned
- The time that the last modification to the contact was made

The data to be configured and captured for each conference include:

- The name of the conference
- The start and end date of the conference (to give good defaults for conference capture)
- The list of attendees that were scanned

1.7. User Interface Styling

The UI should be styled using a modern flat look, consistent with iOS 7 and Android operating systems. We don't need extensive styling, and instead of should focus on a nice clean and functional feel.

The mockups in this document are simply to exemplify structure and are not intended to be a proposed style for the content, only for the important content to relay.

A separate PDF will be provided that includes some style guidelines for Genuitec with common colors and button styles that can be used in the interface, though following the guidelines is not a strict requirement if styling comes for free from a toolkit and is not easily changed.

2. Front-End User Interface Functionality

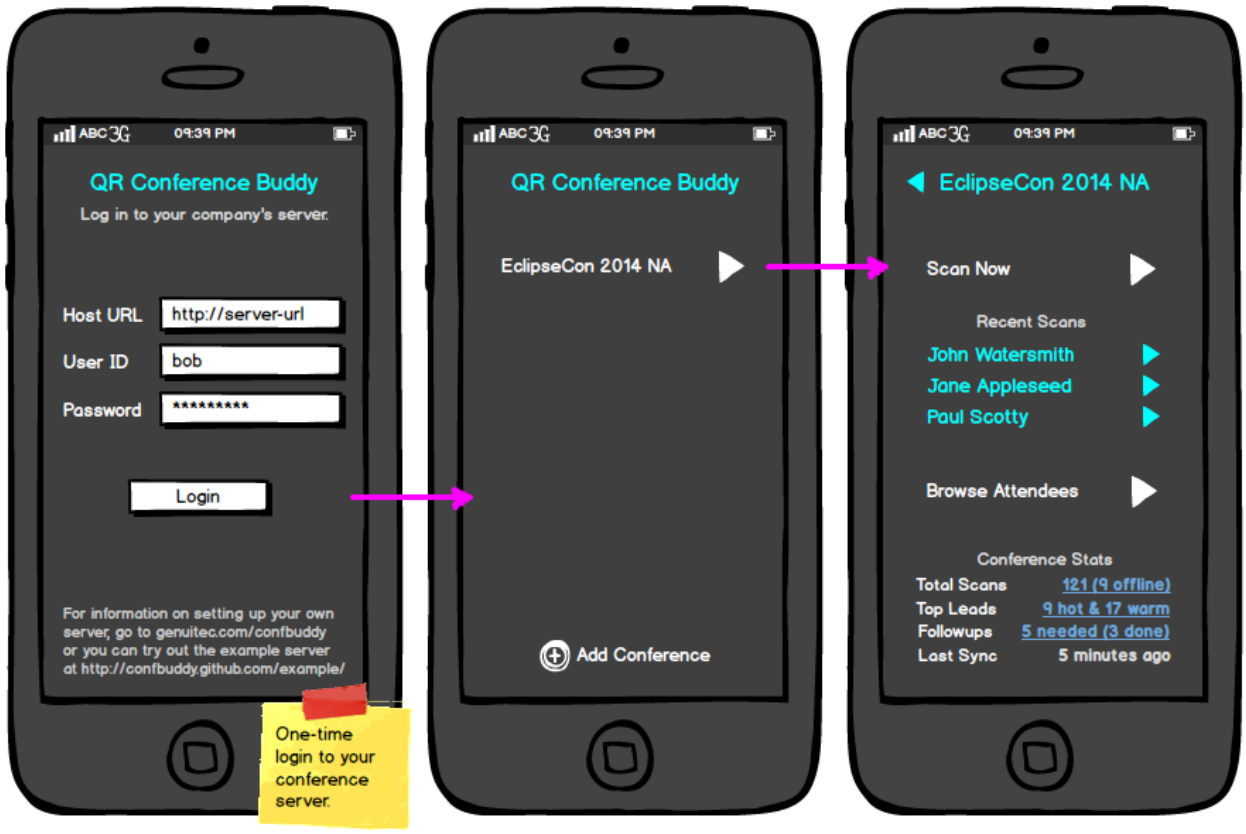
2.1. Server Startup and Login

When the application starts up, it should allow configuration of the server URL to login to, as well as the credentials for the server. The application will use a REST web service to authenticate against the backend and retrieve a list of configured conferences.

The following mock-up gives an example user interface that would address the base needs. Items like the Recent Scans is optional, as is the conference stats. The option to Add a Conference can be implemented via web portal instead of the application, and does not need to function offline.

When choosing a conference, the application will likely want to begin a sync against the server to receive the current attendees scanned for the conference, though the primary need is to make sure the user can begin scanning applications.

Logging in and choosing a conference will require network connectivity.



2.2. Scanning Users

When scanning, the application should be optimized for quick scanning so that a series of users can be quickly scanned in as few clicks as possible. After a scan, an option should be available to change the rating of the attendee, choose to require follow up and select from a series of tags.

The tags will be configured on the Conference object on the server, and should be presented in a way that makes it easy for the user of the application to easily click tags while talking to someone, or quickly right after. The contact should be able to be accessed right after the scan, or when browsing later.

It is not a requirement to show recent scans while in scan mode with the camera. -If- however it is possible to do so, staying in always-scan mode would be great, showing recent scans updated after each scan.

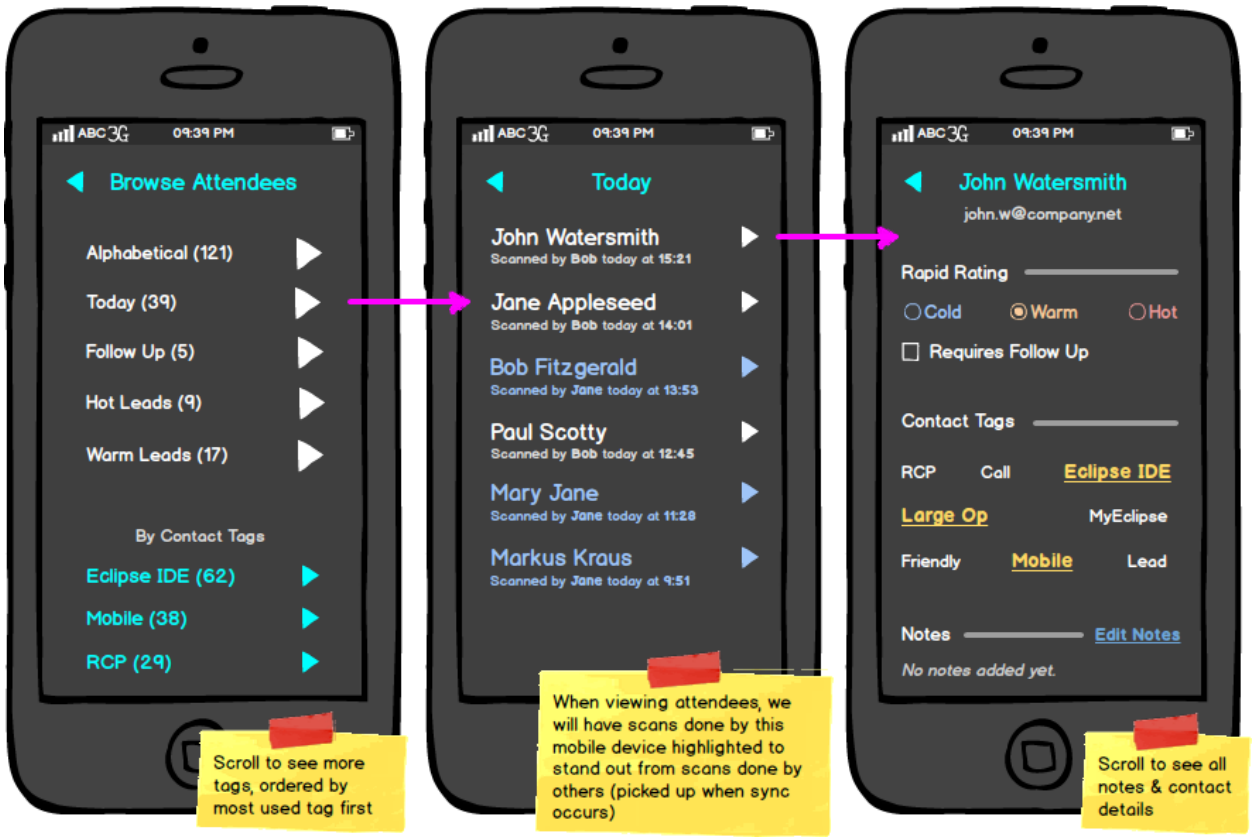
Scanning users must function offline with scan data, and taggings and other contact information queued up so when connectivity is restored, it is sent to the user for processing.

2.3. Browsing and Updating Users

The application should allow browsing of previously scanned contacts. At a minimum a user needs to be able to browse their own scans. The mockup below shows additional capabilities that could be implemented if easily implemented using a SQL backed and if full synchronization comes from the server. That said, the most important feature is being able to view recent scans done on the device (such as from the current day) and allow additional data to be provided for the record.

If, and only if, the application can easily receive data from the server for contacts scanned by other attendees, it would be great to clearly show scans by others such that a user can view all scans based on categories such as those needing follow-up, hot leads or specific tags.

The ability to add additional details to a contact scanned on the same device while offline is required. While browsing of contacts offline (since last sync) is a nice to have, browsing is optional whether offline or connected.



3. Storage Implementation

3.1. Storage Schema

To simplify storage, a simple structure should be used such that each scan or update is saved as an entirely new record with all data in it. Records written on the client will be identified by a field separate from those on the server, and the ID structure should be setup to guarantee uniqueness.

When the mobile app has connectivity to the server, all records updated since last sync (as indicated by a field on the row) should be sent to the server. The server will receive in the request the last sequence number from the previous sync with the server, with the server then able to merge in the latest scans and send back the merged result.

Additional details can be discussed during implementation though the algorithm should focus on guaranteed functionality by segmenting data, instead of heavy architectural complexity. The merging algorithm is identified below.

For identifying a particular scan, the QR code raw data itself can be used as a key (or MD5 of the data), given that two people scanning the same contact will always get the same QR code. This allows for two people with the same name, or other overlapping characteristics to still be scanned effectively.

3.2. Merging Updates

When the server-side receives recent scans from the client, it will use a policy whereby the most specific / most interesting data wins. The following rules will govern the merging:

- The Rapid Rating for the contact: *Hottest rating level wins*
- Whether or not a follow-up is needed: *Most recent change to Follow-up flag wins*
- Tags associated with the contact, customized for each conference: *Union of all selected tags*
- The employee(s) that scanned the contact: *First scan wins ownership*
- The time that the contact was scanned: *First scan wins first scan time*
- The time that the last modification to the contact was made: *Latest time wins for last modification*