SensEvent

A Decentralized Media Application

Philip Shibly

What is SensEvent?

- An avenue for sharing media between users which does not rely on ownership of the composite of media.
  - Users can upload media taken from an event and share it with common participants of that event.
- A way to develop a more complete picture of an event, from a time perspective and a spacial perspective.
  - Participants of the same event has access to all of the media uploaded, so a more “complete” picture of an event can be presented to each user.

Motivation

- The need to present a composite of media to all users.
- Remove any notion of “ownership” of the media or the event – presents a persistence obstacle.
- Those who “host” an event would naturally be the ones aggregating the media, but don't forget about the guests as well. How do they aggregate the media without passing around a URL or tagging a user in a picture?
- What if an event doesn't have a “host” or an “owner” such as a Magic basketball game where people from all social groups participate?
  - What if the police need to gather evidence from the event, and an aggregated source of photos from multiple points of view were accessible?
  - What if one circle of users miss important data about an event, but is present in another circle of users data?
- Remove the need for logging into a separate website to create a profile and to see your event's media.

Approach

- Build an app that lets users share events and presents media to all users in different ways.
- Strip JSON metadata from the media such as geo-location and time.
- Backend aggregation of media from all users associated with each event
- Re-host media in different formats requested by the user.

Design Features

- Application login
- TableView of Events
- MapView of Events
- Add new Events
- Delete yourself from an Event
- Take/Select/Upload media
- Backend Server
  - Metadata stripping
  - Thumbnail previews
  - Push/Pull to server via pList web-requests via XML
  - Host media as user requests it

Difficulties

- Needed to learn xcode
- Successfully pull out metadata on the backend... actually... setting up the whole backend.
- Staying within the proper MVC design.

Lessons Learned

- Have to remove ALL warnings in xcode! They will come back to bite you.
- NSZombie excellent for identifying memory leaks – remember no stack/heap difference in xcode, so alloc,dealloc,assign, and release are important.
- Clean code and sticking to MVC prevents dealloc'd object referencing.
- Using web-requests via pList XML is much easier than using CoreData.
- Apple did an excellent job in taking care of all the low-level API (CoreLocation, MapKit... basically all of Foundation).
- UIImagePickerController strips out EXIF latitude/longitude metadata but leaves the tags there. I don't know why Apple does this?
  - Need to use Core Location separately and send that data as well.

Future Work

- Change the way users add Events by adding feature of taking a photo of a QR.
  - Develop a vision system (or integrate existing one) to translate the QR code into an event and link the user to that event.
- Link logged in user with the UUID of the device they are on.
- Allow users to push an event to a specific UUID to “invite” them to an event.
  - Requires subscribing with Apple's push notification service (APNs), getting certificates, SSL, etc.
- Get Rid of the Server!
  - Think outside the box! Create Event and store its info on the cloud... automate in password-protected Google Images or Flikr.
- Use existing API's
  - ASIHTTPRequest instead of NSURL
  - Three20
- Carry the same color scheme throughout the app.
  - Customize the TableViews
- Customize TabBar icons and Navigation Items.
- Implement ability to upload multiple photos.
- Resize/Stretch photos not taken for the iPhone screen size, and scale down quality of photos for iPhone.
- Use Threading
- Pretty much... build an awesome product.