Shop Social: The Adventures of a Barcode Scanning Application in the Wild

Jonathan Engelsma[1], Ferris Jumah[1], Alejandro Montoya[1], Joseph Roth[1], Venu Vasudevan[2], and Greg Zavitz[1]

[1] School of Computing and Information Systems
Grand Valley State University
Allendale, MI 49401 USA
[2] Motorola Mobility
Libertyville, IL 60048
jonathan.engelsma@gvsu.edu,
{ferris.jumah,alejomontoya,roth.joseph.e,gzavitz}@gmail.com,
venu.vasudevan@motorola.com

Abstract. Mobile retail is a space rich with plausible hypotheses but sparse on longitudinal datasets that give us a corpus of user behavior to validate or disprove theories around the use of digital devices in a physical store. Popular price comparison apps such as ShopSavvy have shown that a smartphone in the aisle is a reality that brick-and-mortar retailers have to contend with. A nuanced, data-driven understanding of a smartphone powered shopper might enable store-based retailers to leverage the smartphone rather than fear it as something that leads to sales erosion. To this end, we built and deployed a novel, mobile retail app that blends mobile, media and social capabilities. In this paper, we describe the user needs and design axioms behind the app, and the data that we've collected over the course of its use by about 5,500 users over the period of a year.

Keywords: mobile, retail, barcode, video.

1 Introduction

The rapid proliferation of smartphones and their associated application ecosystems is reshaping the way consumers shop for products and interact with the retailers who sell them. Smartphone horsepower, 3G+ bandwidth and the availability of high performance barcode scanner apps have made smartphones a powerful comparison shopping tool in the brick-and-mortar aisle. So much so that the retail industry has begun to take steps against showrooming [1], the phenomenon of shoppers using brick-and-mortar stores as a showroom for getting a better look at the product, while ultimately purchasing the product online.

While the retail industry has treated showrooming as a threat, this project began with the hypothesis that retailers might be able to use technology to turn that threat into opportunity for complex, priced goods. In particular the hypothesis was that

D. Uhler, K. Mehta, and J.L. Wong (Eds.): MobiCase 2012, LNICST 110, pp. 379–390, 2013.
© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2013

Instacart Exhibit 1016

beyond price sensitivity, showrooming demonstrates an unmet need amongst consumers to be better educated about products. The superior performance of Best Buy (until recently) demonstrated that within the confines of a single category (brick-and-mortar in this case) retailers that were able to educate, enlighten and de-risk a product to their prospective customer, were more effective in sell through and ultimately overall retail sales.

We wanted to explore whether the combination of mobile, social and rich media on a smartphone could approximate a well-trained sales force in this regard, at a fraction of the cost. The goal was to detect a user's interest in a product (i.e. barcode scan), and to trigger a rich media conversation around the product experience with authentic user generated content. Additionally, social capabilities integrated into the app would enable the user to extend this visual conversation to his social group, a group that social retail research has shown to be highly influential in driving purchase decisions.

Our goal in creating Shop Social was to use an 'in the wild' application to gain concrete insight on the confluence of mobile, social and rich media via in-market learning. In particular, we wanted to understand the relative proportion of usage of these facets amongst various population segments of shoppers, and the potential of app design alternatives to drive adoption of specific usage patterns.

Recently, barcode scanning apps have gone well beyond the basic price comparison concept. [2,3,4,5,6]. However, most of the widely adopted barcode scanning apps focus primarily on price comparison. In the typical use case, the user scans a product's universal product code (UPC), and receives back a list of retailers (both online and near the user's current location) and the price for which each retailer offered the product. We deliberately chose to avoid the price comparison feature, as we wanted to offer users functionality that didn't already exist in other barcoding applications. We also wanted to approach this from the mindset of a brick and mortar retailer, and hence assumed that a price comparison feature would likely not be a priority. We decided early on that the key distinguishing feature of the application would be to focus on delivering the user just-in-time relevant product video based on a barcode scan.

We also noticed that to the best of our knowledge, none of the barcode scanning applications at that time integrated well with social media platforms. Intuitively, it seemed that relevant just-in-time video content along with the ability to interact around products and product content within a user's social graph could be just as compelling as price comparison, and particular helpful for a consumer making product purchase decisions.

In the remainder of this paper we describe the application itself, and the rationale for the various design and technical decisions made in creating it. We provide a brief analysis of over a years worth of application and analytics data collected by the application in the wild, and conclude with a discussion of the lessons learned.

2 Shop Social: An Experimental Barcode Scanning App

Shop Social is an experimental barcode scanning mobile application that locates product information and relevant product video content based on barcodes the user scans. Shop Social leverages the user's social graph along with product-related media

Instacart Exhibit 1016

(product descriptions, reviews, videos, photos, etc.) to help a shopper understand a particular product's potential for them, or for whomever they might be shopping. In an attempt to further encourage engagement, Shop Social also incorporates simple badging game mechanics.

In addition to the native application running on user handsets is a set of non-trivial network services that feed content to the client applications and also serve as a communication conduit to enable users to interact together within the application. In what follows we describe the end-to-end architecture of the application including both the network services deployed in Google App Engine, and the native client applications that were developed for iPhone and Android.

2.1 The Shop Social Backend

We had a number of goals/constraints that influenced the approach we took in architecting the Shop Social experience. These included:

- *Minimum Operational Cost*: Ideally, we wanted to able to deploy and support up to 10K users without incurring hosting fees or at least keeping the costs as low as possible.
- *Scalability*: While we are a university lab without a lot of marketing muscle to promote the app, we wanted our implementation to be capable of auto-scaling up to large numbers of users without having to re-architecture our backend should the opportunity arise.
- *Persist Client Data in the Cloud*: We wanted to keep the client applications as simple as possible, caching data locally only when necessary for performance reasons. A further benefit of having the data in the cloud is that it gave us the visibility we needed as researchers to understand how users were using the application.
- *Integrate with an Existing Social Network Destination for User Authentication and Access to Social Graph*: It made sense to piggyback our application experience on top of an existing social network destination. Not only did this simplify the services we had to build and support in the backend, but it also made it easier for users to automatically discover which of their friends were already using the application. It also allowed users to share application content with friends who were not already using the app, and thereby extending the reach of the application. Facebook was the obvious social platform to integrate with.
- *Utilize Existing Product Data and Product-Related Media*. Not being a retailer ourselves, we had no product database of our own. Fortunately, there are large existing databases of product information searchable by UPC via web service interfaces that can be used to retrieve product metadata. Over the course of the project we integrated with several different product databases, and are currently using Google's Search API for Shopping. YouTube's public search APIs coupled with some simple heuristics was used to find relevant video content. We used Flickr APIs to search for relevant product photos.
- *Adopt a "platform" Approach*: Our goal was to make it as easy as possible to swap out architectural components moving forward. e.g. keep all Facebook-related

Instacart Exhibit 1016

integration in the backend within a single adapter, so in the future a different social destination could be used with minimum impact on the code. All sources of product data were integrated via a generic adapter component, so we could readily switch to another database in the future with minimal impact on our code base. Paying attention to this early on turned out to be very important as we switched product data sources several times to-date.

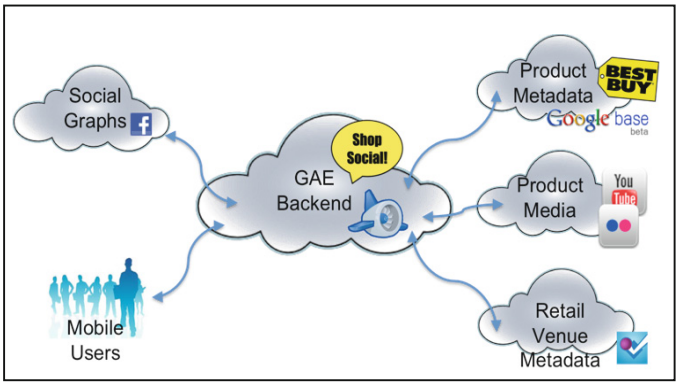

Fig. 1. The end-to-end Shop Social Architecture

Figure 1 above provides a high level view of the end-to-end Shop Social architecture. It consists of the following components:

Mobile Users: End users install the native Shop Social application via the appropriate application store (Apple iTunes App Store and the Google Play Store) on their mobile phone, and then proceed to scan product barcodes and interact with product content and other users via their Facebook friends. Users of the Shop Social are allowed to interact in two different modes: anonymous and authenticated. In anonymous mode, the basic app features are available such as scanning products, viewing relevant videos, etc. If the user authenticates they can also keep track of their favorite products, share app content, view their social dashboard, and earn badges. Sharing extends the reach of the app beyond the users who have installed it on their phone. For example, Facebook users who may not be using the Shop Social application on their phone, might interact with the application when they encounter content generated on-line (e.g. a user recommends a product or comments on a video posted via the app on another user's Facebook wall, via the Shop Social mobile app.) Our backend system also tracked these sorts of interactions on Facebook by having all links in Shop Social postings redirect through our server.

GAE Backend: The Shop Social backend services were implemented on Google's AppEngine (GAE) platform. The backend component coordinates and persists all end user interactions in the application. This includes mobile users running the Shop Social client application as well as online users who happen upon content generated by the application in their Facebook activity stream. The backend also is responsible for efficiently generating a user's social "dashboards" on demand, and locating

Instacart Exhibit 1016

relevant product media (videos/photos) via a set of heuristics that operate on the product/media metadata as well as the historic user interaction data persisted over time.

Social Graphs: In terms of social network destinations, the current backend integrates with the Facebook platform. The backend integration is maintained as a generic "adaptor" to facilitate future integrations with other social platforms.

Product Metadata: When products are referenced, via UPC scans, clicked on from social dashboards or from scan history lists information on that particular product needs to be retrieved from the network. Currently, the backend adopts a two-step process. First, the backend performs a lookup via Best Buy's Remix API the first time it sees a new UPC code. If no data is found there, the backend then "punts" the request to the Google Search API for Shopping. Remix provides better and more consistent metadata for many popular consumer electronics products, as well as user reviews.

Product Media: Currently the backend utilizes the public web APIs of YouTube and Flickr to locate videos and photos relevant to a particular product. This process will be described in more detail below. Once again, the backend integration with these services has been carefully factored out into a set of adapters that can easily be exchanged with alternative product media sources moving forward. The client integration (e.g. when photos/videos are actually viewed on the client) is also decoupled (e.g. operates on URLs) and would not be difficult to direct to a different media source.

Retail Venue Data: The application attempts to encourage in-store interaction around products by offering various badges for in-store interaction. In addition, the application locates retail venues near the user's current location and encourages them to check-in. The latest revision of the client utilizes the FourSquare Venue API and provides a wide range of nearby retail establishments to the end user. Backend integration is kept to an absolute minimum, in that it only needs to tag interactions with venue ids when they occur within a store, as well as a track of how many times each user visits a particular venue in order to award badges.

Google Analytics Instance: Though not shown in Figure 1, both of the client apps are carefully instrumented with Google Analytics to generate detailed analytics as user interact on the clients. The data collected by Google Analytics as well as the user activity stream persisted in our GAE backend will be analyzed in the next section.

2.1.1 Locating Relevant Video Content

Relevant videos for a given product are discovered via heuristics that take both product/video metadata into account as well as prior user video interactions and the requesting user's social graph. The social component is based on the intuition that videos discovered in the context of a given product, are likely to be more relevant to you if your friends interacted with that same video in the past.

The video search begins by locating previous video interaction for the same barcode, and scores every video v previously associated with a UPC code or that turns up when YouTube is searched with the product name. The score s_v for each video v in this initial set of videos is computed as follows:

Instacart Exhibit 1016

$$s_v = YTW_{name}(v, pname) + \sum_{k \in A_{v,upc}} SW(k_{age}, k_{type}) * FW(uid, k_{uid})$$

where YTW_{name} is a weight function which returns a positive non zero value if v turns up in a YouTube search on product name (pname), or zero otherwise. $A_{v,upc}$ is defined as the set of all the past activities that involved video v in the context of the product upc, and SW is an initial weighting based on the age and type of the activity. Possible activity types in $A_{v,upc}$ can be view video, share video, or rate video, and each of these is weighted differently. FW is a weight function that will be a positive non-zero value if the activity k is tagged with a user k_{uid} that happens to be in the social graph of the current user uid, and returns zero if the activity was not tagged by a friend. The idea behind SW is to take into account the nature of the past interactions, and also to gradually reduce their importance as they age. The FW function is the component that incorporates social relevance into the scoring.

Once this initial set of videos is located and scored, it is sorted and the N highest videos are selected. If there are < N videos total at this point, then YouTube is searched with increasingly broadening terms, initially by a more sanitized version of the product name (e.g. one that has punctuation, etc. stripped out), followed by a search by product category, followed by a search by product manufacturer. Each video that turns up in these broader searches is scored by a constant weight that decreases as the search broadens. If a video turns up from the set of previously scored videos described, the scoring is accumulative. After each addition search, the current list of candidate videos is sorted, and if there are < N total, the search continues.

This approach almost inevitably returns videos. The situation in which no videos are discovered has been discussed. One possible action in this situation is to simply return a list of obviously irrelevant videos (e.g. Charlie Chaplin black and whites) with a message "Sorry, nobody seems to have made a video of this product, but we thought you'd might enjoy these instead." Since in practice this rarely occurs, at the moment if this were to happen the user simply gets no videos.

Photo content was added to the application as an afterthought and at the moment we simply do a brute force search on Flickr using the product name. In practice we've noticed the photos are often irrelevant and this is confirmed by the analytics data capture, as very few users have interacted with the photos the application turns up.

2.1.2 Efficiently Generating the Social Dashboard

GAE's persistence layer is optimized to handle requests for a single entry. It is not convenient to aggregate data or request a set of random entries. To display the social tab in Shop Social, the application needs to query the database based on a user's list of Facebook friends that is dynamically fetched from a Facebook API. Internally, App Engine converts the SELECT IN query on the array of user ids into a separate query for each user id which is not going to be very efficient in our context. According to Facebook, the average user has 130 friends [7]. Running 130 queries is not an optimal use of resources, and some individuals have thousands of friends. In most cases, when bootstrapping a new application like this, it's likely that most of the user's friends are not yet using the app, so a significant amount of work would be done to return a result of a small number of "hits".

Instacart Exhibit 1016

To reduce the number of queries against the datastore, the user ids of authenticated Shop Social users are stored via a Bloom filter [8]. A bloom filter runs a series of hashing functions to set bits in a bit string. The basic idea is to tolerate very infrequent false positives for enormous efficiency gains. The Bloom filter is persisted in GAE's memcache for fast access and backed up in the datastore for integrity. Before querying the datastore with the array of user ids, they are passed individually to the bloom filter to identify which ids are in the datastore. The test in the bloom filter is quicker than a query against the datastore and it also does not count towards GAE quotas. Since the bloom filter uses a hashing function, there is a small probability for a collision that will produce a false positive. False positives have negligible impact since the filtered list of user ids will be queried against the datastore to obtain the actual friend data and an occasional invalid id will still result in the correct data and only cost an extraneous access to the data store.

Before using the bloom filter, users with over 500 friends could experience significant lag when loading the social tab. Occasionally, the request would exceed App Engines CPU time limit and fail to complete. Even with smaller numbers of friends, the backend would quickly consume its daily free quota and we'd end up with a bill from Google. With the implementation of the Bloom filter to pre-screen the friend data queries, dashboard loading is very efficient and typically independent of the number of Facebook friends a user has, as most users have a relatively small number of friends actually using the application.

2.2 The Shop Social Mobile Application

The Shop Social mobile application was implemented for both the iPhone and Android phones. The iPhone version of the application was made available to the general public for free in the Apple iTune App Store. The Android version was made available for free via the Google Play Store, as well as the Amazon App Store.

The application supports a variety of features including, product lookup by scanning or keying a product UPC. The UPC is forwarded to the GAE backend which returns the product description, product reviews, and a list of relevant videos and photos. Recently scanned products are retained temporarily in the scan history, and the user also has the ability to favorite the product adding it to their "My Stuff" list.

The user's social dashboard consists of a list of their Facebook friends, a gallery of badges that each friend has earned to-date, and a list of their favorite products. All of the functionality available to users when they scan a product directly is also available to users when they encounter products on their social dashboard. In addition to watching product videos, users are able to share the videos via Facebook, and/or rate videos with a thumbs-up or thumbs-down as shown in Figure 2. The fact that a user watches a video and/or rates it is ultimately taken into account in future video relevancy scoring as described in the previous section.

Instacart Exhibit 1016

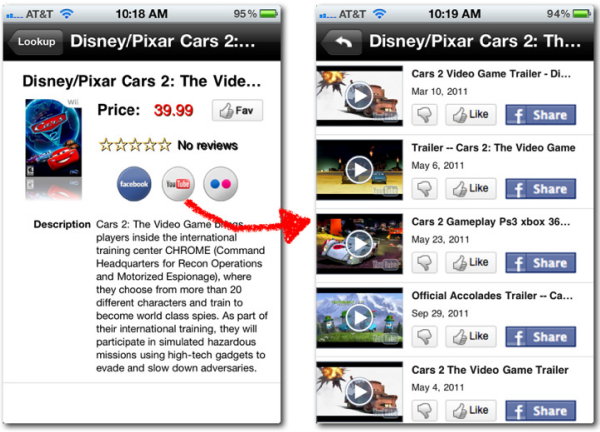

Fig. 2. A barcode scan of the Cars 2 Video Game brings up product data and a list of relevant YouTube videos

We were also interested in learning whether or not people would interact with product information via the application while they visited brick-and-mortar retail establishments. In order to accomplish this the application uses its current location to search the FourSquare backend for nearby retail stores. Since FourSquare has many non-retail venues, the lists of nearby venues obtain from FourSquare was filtered by venue type, and only retail businesses were displayed. Users could then "check-in" to the venue, much like they do in the FourSquare app, and all subsequent product activities (as long as the user stayed in the general vicinity) would get tagged with that venue id. In order to incentivize check-ins, the user who checks in to the venue the most becomes its "Best Badger", which is analogous to the FourSquare concept of "mayor". We also introduced a variety of additional badges to try to encourage users to engage with the application.

3 Results

This section describes what has been learned from the usage data that the application collected via Google Analytics as well as the data captured on the Shop Social backend.

3.1 User Participation

The total number of unique mobile users from 15 October 2010 – 23 November 2011 was 5,499. Of those, 3,804 of them were using Android devices, and 1,695 were using iPhones. Of the overall user base, 527 of them actually authenticated with their Facebook credentials, which allowed them to interact with friends using the app and share application content with friends on Facebook whether or not they were using Shop Social. This had an impact on how the users interacted with the app, which will be discussed in the next section. The chart in Figure 3 below shows the total number of unique users using the app each day during that range. The tall sharp peaks early on in the Android plot correspond with updates in the respective application stores.

Instacart Exhibit 1016

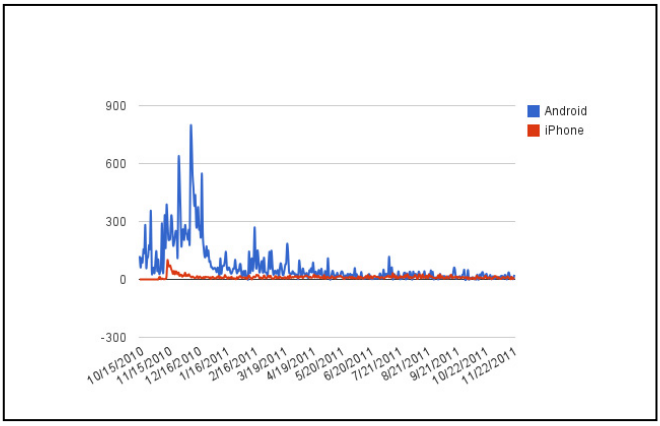

Fig. 3. Number of participants overtime using Android and iPhone smartphones

3.2 User Interactions with the Application

A user interaction with Shop Social is referred to an "activity". In the current implementation, the following activity types are defined and tracked:

- scan: a barcode is scanned.
- video_view: a video associated with a product is viewed by the user.
- share: an item (product, video, badge, photo, etc.) is shared by a user via Facebook.
- checkin: a user checks into a retail venue.

Figure 4 contrasts the frequency these various activities occurred. Barcode scanning is by far the most popular activity, followed by video viewing. These two activities are available to users independent of whether or not they authenticate with Facebook. Sharing and checkin activities were only possible once users authenticated.

In terms of the breakdown of activities over the two population segments of the user base (authenticated vs. non-authenticated), the former had a much higher activity level than the latter. The authenticated users who represent around 10% of the overall user base generated approximately 43% of all activities. Hence, authenticated users of the application are more engaged than non-authenticated users. While we've learned from this and other applications we've deployed in the wild that it is very important to allow users to use the application in meaningful ways without authenticating, this data suggests to us that it is important to incentivize user's to authenticate with Facebook or some other popular social destination. Intuitively, once a user authenticates with Facebook in Shop Social they can begin to see what product content their friends are interacting with, which presumably is more engaging to most users than product information void of any social data.

Instacart Exhibit 1016

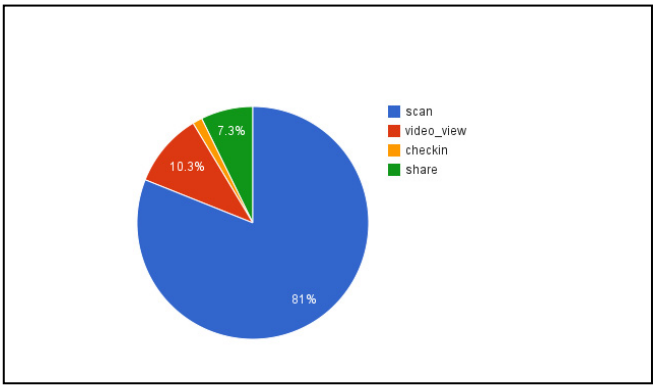

Fig. 4. Breakdown of the types of activies users performed

3.3 Social Sharing

While scanning barcodes was by far the most common activity in which users engaged, users did actively share product content they encountered while using the application. Approximately 80% of the shared items were products, followed by videos (17%). The remaining 3% of the shares were product photos and badges earned by users.

We also broke down the products that appeared in the aggregate activity stream over the past year into specific product categories to determine if a particular type of product was interacted with more than another. Figure 5 shows the various product categories that the application activities were involved in and shows in particular the percentage of actions overall in a particular product category as well as the percentage of overall share-related activities in that category. Looking at the data in this way we can see that while products in the food, book, and health & beauty categories generated the most interest in general, users are more inclined to share information in the consumer electronics and video games categories. One possible explanation for the dominance of the book and food related activities is the fact that the UPC codes on these products typically persist overtime on the food container or book cover. When users load the application up for the first time, the nearest barcode is probably on the nearest book or food container.

Another interesting set of data the backend captured was the response to the shared data. When users shared product and video information on their Facebook activity streams from the application, any click-through events by any Facebook users, regardless of whether or not they were Shop Social users, were redirected through the Shop Social backend. The data showed that video content (essentially product relevant YouTube videos) though shared less, than products themselves, on average received more click-through response per share than the product shares (essentially a product thumbnail image with link to a webpage with more info on the product).

Instacart Exhibit 1016

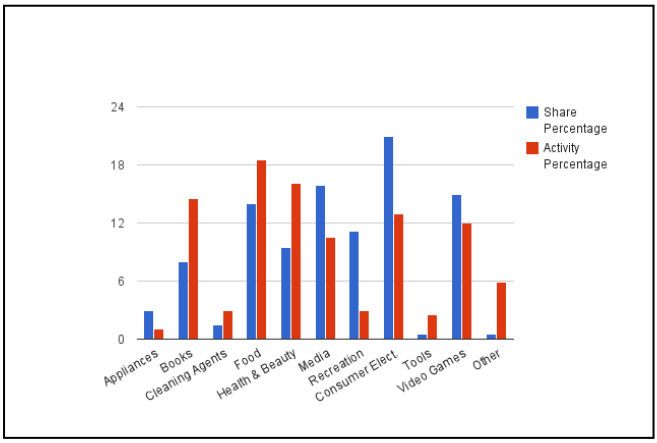

Fig. 5. Aggregate activity vs. share activity by product category

4 Conclusion

Shop Social has been a fascinating learning experience from both a technical perspective as well as an experience design perspective. While we embarked on this journey as an attempt at gaining more practical insight into how a brick-and-mortar retailer might utilize mobile technology, the project ended up taking a life of its own. Along the way we learned a number of lessons that we feel transcend the retail problem space we were looking at, and apply to many different contexts in which one is building mobile applications and deploying them in the wild. Some of the specific lessons learned include:

1. Engaging end users in a mobile experience is very difficult. The vast majority of the users who have downloaded the app seem to bounce, never to be seen again.
2. Android is the more interesting platform from a mobile applications research perspective. We believe part of the success we had with Android was due in part to the speed at which we could iterate on the experience design and immediately deploy in the Google Play Store. The bump in usage every time a revision became available was very consistent. Not only can problems be addressed swiftly, but ideas can be tested and tinkered with. This cannot be done nearly as effectively on iPhone given the approximate 8 days it takes for revisions to make it through the Apple curating process.
3. In terms of social sharing, a few interesting shared videos are worth a thousand product shares. This point was made earlier, and is worth noting once again. While it could be argued we haven't yet collected enough data, it does seem intuitive and is supported by what we've seen in the data so far. We will keep monitoring this moving forward.

Instacart Exhibit 1016

4. Automating product relevant video search on YouTube is reasonably feasible, even without a significant amount of participation. We think the current heuristic works quite well, without a lot of usage data to optimize with. It should get even better with more usage.
5. While attracting iPhone users is a more difficult proposition than attracting Android users, the data collected in this experiment indicates that engaging iPhone users beyond the initial app impression is easier than engaging Android users. This may be due to the more consistent and appealing user interface of the iPhone platform.
6. While its important for apps to offer utility to anonymous end users, authenticated users are more engaged users. In our study, around 10% of the users generated 43% of the activity. This seems to suggest a couple of very important guidelines to app developers. a) Make sure there is a meaningful return on investment for authenticating. b) Make authentication as pain free as possible.

References

1. Eason, G., Noble, B., Sneddon, I.: 'Showrooming': People Shoppin in Stores, Then Researching By Cell Phone, Says Pew Survey (January 31, 2012),
   http://abcnews.go.com/Technology/pew-internet-showrooming-half-cell-phone-users-research/story?id=15480115#.T90OJitYui2
2. Dorman, K., Yahyanejad, M., Nahapetian, A., Suh, M.-k., Sarrafzadeh, M., McCarthy, W., Kaiser, W.: Nutrition Monitor: A Food Purchase and Consumption Monitoring Mobile System. In: Phan, T., Montanari, R., Zerfos, P. (eds.) MobiCASE 2009. LNICST, vol. 35, pp. 1–11. Springer, Heidelberg (2010)
3. Adelmann, R.: Mobile Phone Based Interaction with Everyday Products - On the Go. In: Proc. of 2007 International Conference on Next Generation Mobile Applications, Services and Technologies, pp. 63–69 (2007)
4. Bulusu, N., Chou, C., Kanhere, S., Dong, Y., Sehgal, S., Sullivan, D., Blazeski, L.: Participatory Sensing in Commerce: Using Mobile Camera Phones to Track Market Price Dispersion. In: Proceedings of UrbanSense 2008 (2008)
5. Gao, J., Kulkarni, V., Ranavat, H., Chang, L., Hsing, M.: A 2D Barcode-Based Mobile Payment System. In: Proc. 3rd International Conference on Multimedia and Ubiquitous Engineering, pp. 320–329 (2009)
6. Deng, L., Cox, L.: LiveCompare: Grocery Bargain Hunting Through Participatory Sensing. In: Proc. 10th Workshop on Mobile Computing Systems and Applications (2009)
7. Burbary, K.: Facebook Demographics Revisited (March 7, 2011),
   http://www.facebook.com/note.php?note_id=197149076992338
8. Bloom, B.: Space/Time Trade-Offs In Hash Coding With Allowable Errors. Communications of the ACM 13(7), 422–426 (1970)

Instacart Exhibit 1016