1 Abstract

Snazzi is a step toward bringing paid crowdsourcing onto a mobile platform. We developed a prototype iOS app that allows Yale students to earn money on the go by performing simple content evaluation tasks. Students use the app to browse fashion styles from retailers around the world and provide feedback by swiping on products. With each swipe, students earn points that can be redeemed for cash and other rewards. Retailers, meanwhile, get valuable data on product appeal as well as engaged exposure for their brand and products.

For this project we had to address cheating and quality assurance: two major hurdles for paid crowdsourcing on any platform. Requiring users to sign up with a valid yale.edu email is a good first step towards accountability, but in designing the app we still had to develop new anti-cheating strategies tailored for a mobile platform.

Our experience leads us to believe that there are unique advantages to smartphone-based content crowdsourcing. Support for multi-touch gestures allowed us to create a clean and frictionless user experience to combat participant fatigue and burnout. Furthermore, we note that users may find paid crowdsourcing opportunities most compelling when on the move or when waiting for something. In these instances, the opportunity cost for engaging in paid crowdsourcing is minimized.

As we continue to test and develop the app, we will explore the potential for mobile-friendly ‘gamified’ and social elements to augment - or even replace - monetary incentives for simple crowdsourced tasks. We also hope to test the limits of paid crowdsourcing on mobile devices by fielding more involved tasks such as content creation and editing.
2 Features

Our app offers the following core features (among others) to users:

- View text and image content provided by third parties
- View relevant contextual information for each piece of content
- Provide binary feedback on content appeal

In order to enrich the user experience, drive engagement, and incentivize desirable user actions, our app also offers the following ‘gamified’ features (among others) to users:

- Earn points for providing content feedback
- Convert points to cash and/or other rewards

Finally, our app also incorporates safeguards and restrictions as a third set of features designed to modulate user actions:

- Restrict user access to rewards

2.1 Displaying Content

Content is displayed to users as cards. Each card contains not only the content to be evaluated, but also a selection of important contextual information (such as the name of the content’s owner) that might help shape the user’s judgement. Further contextual information, including a link to the content in situ, can be accessed from a detail view.

2.2 Capturing User Feedback

Our app incorporates a seamless and intuitive mechanism for capturing binary feedback from users. A user can indicate whether they like or dislike the content of a card by swiping right or left, respectively.

2.3 Rewarding Users

Closely integrated with user feedback is a system for rewarding users with points. Whenever a user provides feedback, the app assigns the user a fixed number of reward points - subject to the restrictions and limitations outlined in the next section.
2.4 Restrictions on User Rewards

In order to maximize the accuracy and usefulness of user feedback, the app safeguards against two types of egregious misuse: (1) entering random feedback in rapid succession with no more than a cursory glance at the content being presented, and (2) entering random feedback at a deliberate pace without examining the content at all.

2.4.1 Reward Preconditions

To safeguard against the former, the app requires that card content be visible to the user (displayed on the user’s screen) for a fixed period of time before a user can earn rewards; providing feedback on a card that has not been visible for long enough will not trigger the reward mechanism. Visual feedback is provided in such cases to minimize confusion.

2.4.2 Challenge Cards

Challenge cards are designed to prevent users from earning rewards without actually viewing content displayed for them. Each challenge card is visibly identical to a regular card in every respect save for one: an unobtrusive visual prompt for a specific user action. If a user interacts with the challenge card in a manner inconsistent with the visual cue, they will receive a strike; after a certain number of strikes, the user is barred from providing feedback on content for a period of time.

3 User Experience

We want to make the features of our app as easy-to-use, accessible, and intuitive as possible—within the limitations of a mobile platform. We opted for a tab-based interface that allows users to access the most important areas of the application with a single tap; additional features are included in a collapsed sidebar that users may access from a conspicuous menu button in the top navigation bar.

3.1 Browsing Content

We aim to provide a frictionless and intuitive platform for users to discover interesting and relevant content to interact with. With this in mind, the primary tab in the application’s interface is a continuous feed (referred to as a ‘deck’) of content cards. Cards are presented in single-column layout to maximize the space available for content presentation, and are presented in reverse chronological order in order to ensure that the ‘freshest’ content is presented first. Scrolling reveals
new cards to the user; additional cards are fetched and queued automatically as
the user scrolls without the need for additional input or user interactions.

Complementing the immediate accessibility and ease of use associated with the
main feed is a browsing interface that allows users to filter content cards by
categories and subcategories. By exploring the second tab (‘Browse’) in the main
interface, the user can view a grid of content categories. In order to facilitate
identification, each grid cell contains an image that represents the associated
category. Upon selecting a category, the user is presented with a deck containing
cards belonging to that category. If that category contains subcategories as well,
those subcategories are listed at the very top of the deck; selecting any of these
subcategories filters the cards further.

3.2 Providing Feedback

Content feedback functionality is integrated seamlessly with browsing and content
display. By swiping right on a card (either in the primary tab or in the ‘Browse’
tab) users can indicate whether they ‘like’ or ‘dislike’ the associated content.
Using these intuitive gestures in lieu of a button-based UI allows for a cleaner,
less-cluttered interface and a more fluid user experience; these in turn help keep
the user’s focus on the content rather than on the app itself. Visual feedback is
provided while the swipe is in progress to minimize user error.

Each time the user provides valid feedback, a visual indicator representing the
user’s current point balance updates itself. This notifies the user that they have
earned points. When the user fails a challenge card (as outlined earlier), they
are presented with an explanatory alert.

3.3 Cashing Out

Upon tapping on the balance indicator (or navigating to “Cash Out” in the
navigation menu), users are taken to a dedicated interface for converting in-app
points into cash. The interface includes a set of different cash out options
structured so as to incentivize users to keep their points within the app as long
as possible.

There are two hurdles to cashing out: (1) the user must have sufficient points for
the option they select, and (2) the user must have registered a valid Venmo ID.
In the former case, the app will present an explanatory alert to the user; in the
latter case the app will prompt the user for their Venmo ID before the checkout
process can continue.

If the user clears both hurdles, the appropriate number of points are deducted
from the user’s balance.
4 Implementation Details

Our app’s backend was built using Ruby on Rails 4.2 with a PostgreSQL database, and adheres RESTful architectural constraints. The frontend is an iOS application written in Objective-C using XCode 6. Image assets are stored on Amazon S3.

4.1 Backend

We used the RSpec behavior-driven development framework and the guard-rspec gem to develop the application’s backend. We used Rails’ built-in fixtures functionality for test data. Gems used in production include aescrypt for AES encryption, oj and multi_json for parsing and creating JSON, and bcrypt for core user authentication functionality. The app is hosted on a Heroku cloud PaaS. To improve app performance, all image assets are hosted separately on Amazon S3.

4.2 Mobile Application

Our app runs as a native iOS app on iPhone 4, iPhone 5s, iPhone 5, iPhone 6, and iPhone 6 Plus running iOS 7+. The data model on the mobile app has five core components: RailsClient for handling communications with the backend; UserSession for maintaining session-specific user data; Configs for dynamic configuration profiles; User for user data; and Card for card content and metadata.

Outside of the scope of this CS490 final project, many features of the mobile app (including an in-app referral system and user profile / settings management) were implemented Yuri Aluja.

5 Launch

On April 26th, 2015 our app was approved by the Apple Review Board, and on April 28th we quietly released our app onto the App Store. We have soft-launched the app with friends and colleagues, and in the coming months we will incorporate user data and feedback into our app in anticipation of a launch sometime early next semester.

6 Future Work

Our app is a small step towards making crowd-based content evaluation more accessible, engaging, and fun. Immediate future work will focus on applying user
Figure 1: Entity Relationship Diagram showing backend model attributes and relationships
data and feedback to improve user experience and to compare the evaluation data that the app produces with evaluation data from traditional channels.

From there, we are keen to explore new ways to drive and enhance user engagement and activity without increasing the cost per evaluation. Building on the basic points-based gamification model we have already implemented, we are interested to see if badges, achievements, levels, and other gamified elements can help complement or even replace some of the monetary incentives built into the current iteration of the app.

Finally, and most importantly, we aim to test whether the model we have developed for this platform can be used for content generation in addition to content evaluation.