Shout! Design Doc

Identification
The working title of the project is Shout!. The product manager is Cameron Porter.

Meet the team:

Andrew Grasso ‘15
Cameron Porter ‘15
Cole McCracken ‘15
Julian Griggs ‘15

Overview
Shout! is going to be a service for sending and receiving anonymous messages to and from nearby users. The target user base is college students. The primary focus will be on an iOS application, but a web interface will be developed as well if time permits

The application will run Django on a Linux server, with MongoDB for a database.

Functionality
One purpose of the app is to enable social interaction between nearby strangers. For example, a user attending a concert would be able to discuss the show with other fans, without having met any of them before. A student could ask a question in a lecture about the material presented without disrupting the class. We have thought of many potential uses, but believe the biggest strength of Shout! will be in its versatility to facilitate the many imaginative uses people will come up with.

As it is a messaging application, the main uses of Shout! will be twofold: to send messages, and then to receive them. In a basic “sending” session, a user will open the app to say something to nearby people. He or she would then write the message, set a distance corresponding to the geographical area in which they would like the message to go, and then the message would be delivered to all people using Shout! in the specified area.

Below are some features that we think we want to add to the basic app:

- Upon registering the app your contacts would be added so you could have “friends” in Shout!.
    - This requires Shout! registration to either look up Facebook friends using Shout! or to require use of the user’s telephone number when registering

- An option to send a private message, to a Shout! friend (or group of friends) in your area.
    - This option leads to the notion of Shout! groups where Shout! friends can gather in a special group so as to quickly and easy send Shouts to one another
- Option to choose anonymity or reveal identity
- All Shouts can be given a thumbs up or a thumbs down
    - Number of thumbs up vs. thumbs down will determine a user's the quality of the users shout or how "Heard" they were.
- Configurable Shout! notifications. User can decide if they want to receive Shouts whenever they arrive, check for Shouts every 30 minutes, every hour, etc.
- Due to the risk of cyber bullying, we plan to implement a feature for reporting malicious users / offensive messages. Too many reports will result in suspension from Shout!.
    - If user is offensive we can "Mute" he or she for being "Too Loud".

Our initial implementation will only allow messages containing solely text to be sent however we envision allowing photos and videos once we have the basic messaging implemented. In addition there is possibility of also providing an interface similar to HeyTell so as to send recorded audio messages.

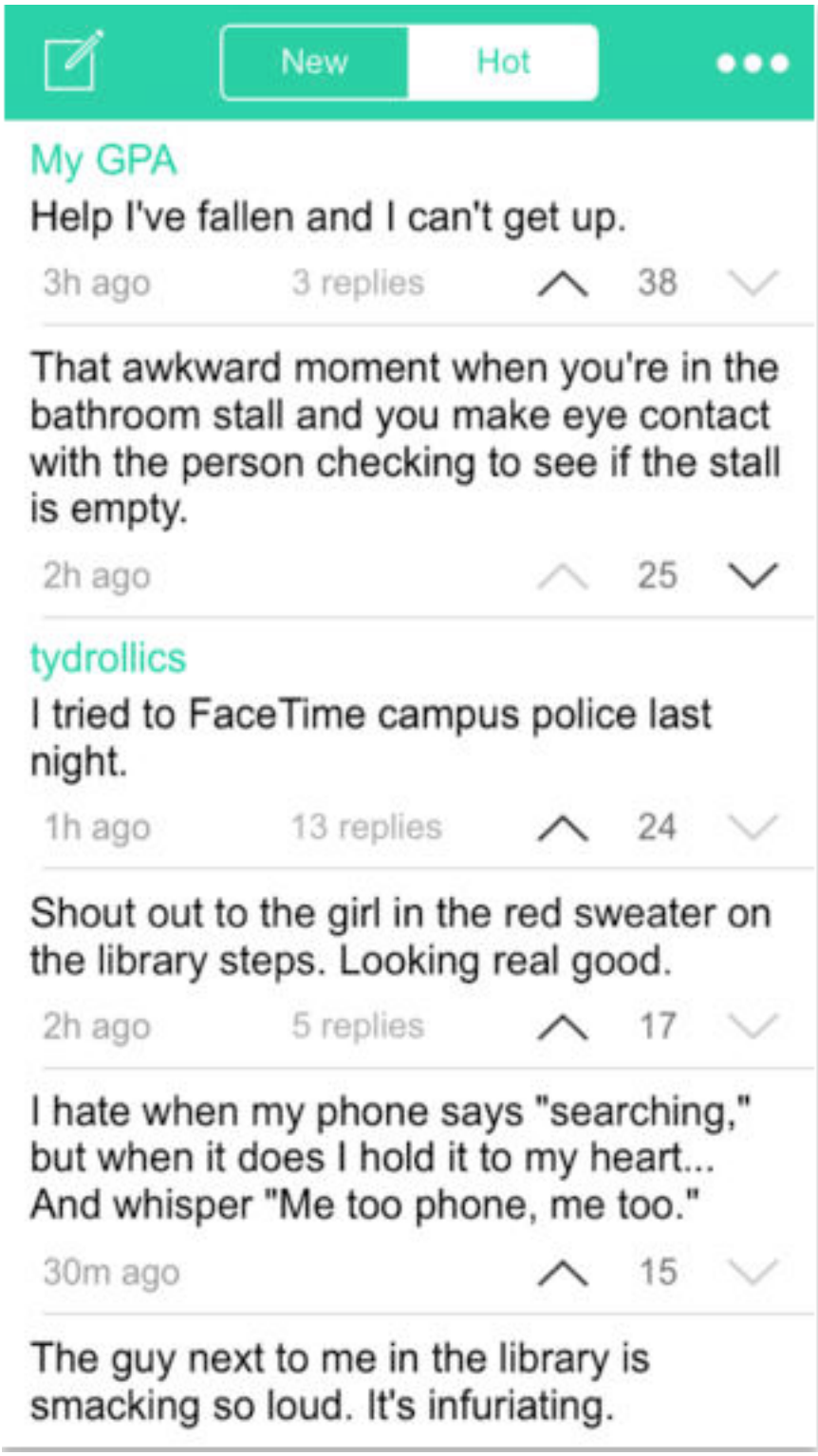
As we develop we also will add in other features we think may be desirable including hashtags (#) and "Shouting" at (@) other users.

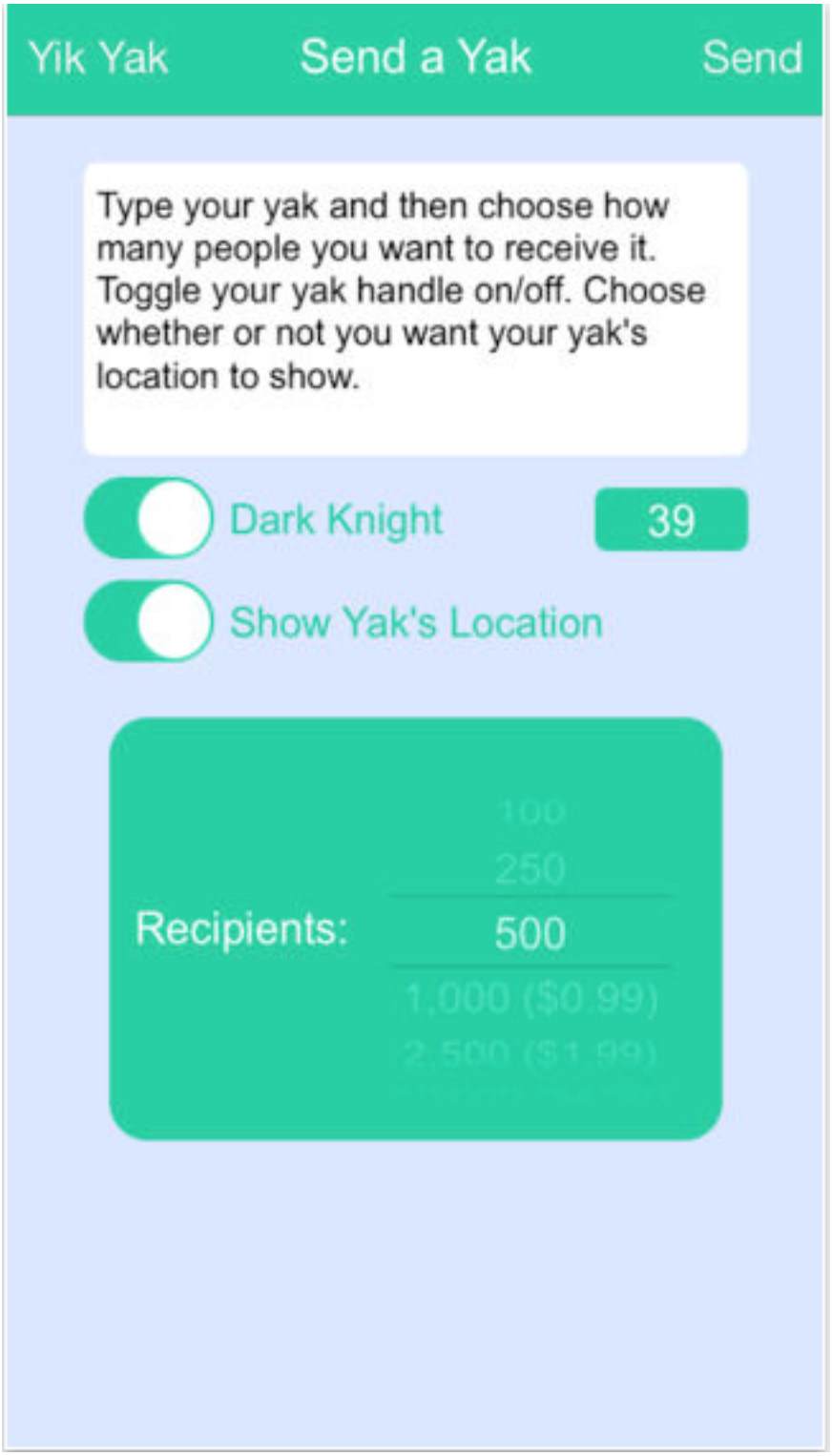
Design

On the server side, we will run Django on a Linux server. Data will be stored in a MongoDB database. The data stored on the server will be of predominantly two kinds: User data and Message data. Initially, User data will include: username, password(hashed), Shout Quality, and number of Shouts, and number of complaints regarding offensive Shouts. As we build up we plan to add data such as elements pertaining to user profiles(School, Hometown, Profile Picture). Much of this data will ideally come quite naturally through a Facebook integration. In terms of Message data, we plan to store each message sent along with the geographic location it was sent from and the distance it was set to send to. Much like Snapchat, messages will expire after a certain amount of time. This time has yet to be determined.

Clients will communicate with the server through a JSON API. Clients will make a request including their current location, and will receive a list of recent messages from within a certain radius.

The primary focus will be on an iOS application. The main page of the application will be either a timeline of recent messages, or a plot of the messages on a map.


New
Hot
My GPA
Help I've fallen and I can't get up.
3h ago
3 replies
38
That awkward moment when you're in the bathroom stall and you make eye contact with the person checking to see if the stall is empty.
2h ago
25
tydrollics
I tried to FaceTime campus police last night.
1h ago
13 replies
24
Shout out to the girl in the red sweater on the library steps. Looking real good.
2h ago
5 replies
17
I hate when my phone says "searching," but when it does I hold it to my heart... And whisper "Me too phone, me too."
30m ago
15
The guy next to me in the library is smacking so loud. It's infuriating.



The images above are both taken from a competitor, Yik Yak. We plan on implementing a User Interface that is simple and clean like the images above. Note that the Message Sending UI will emphasize the text more than the example above. Also, we will not be using the hideous green you see above.

A web app will also be developed if time permits. The app will make asynchronous calls to the same API used by the iOS app to keep things simple. The app will be built using Bootstrap for layout and jQuery for scripting.

Milestones/Timeline
Feel free to change around these dates.

March 7, 2014: Initial meeting
Met with Professor Kernighan to pitch the idea and go over potential features

March 25 - 28, 2014: Server set up
Write the code for the server and get that running.

March 30, 2014: Minimum viable product
Allow text messages to be sent with a radius the user chooses.

April 11, 2014: Project prototype

April 25, 2014: Alpha test

May 2, 2014: Beta test

May 13, 2014: Dean's date

Risks and Open Issues
Firstly, none of us have experience using Django. However, we do have a good background in Python, so this should not be too difficult to overcome. We also have minimal experience with iOS development. This will likely be our largest hurdle, but we think it worth it to learn iOS development along the way. Not only do we plan on taking the Stanford Course on iOS development, we plan on attending the Hackathon

For now, we'll be running the application on a linux box in a dorm room. This is a free and simple solution, but will likely not scale well. Eventually we may consider moving to AWS or Heroku.

Similar apps have had issues with negative content, to the point of being banned in some schools. We plan to initially restrict membership, and to implement features like silencing malicious users to combat this, but it may still be a problem.