Anonymoose

Anonymoose Industries Ltd
Alex Toop, Lewis Sweeney and Joseph Thompson

PRIVATE AND CONFIDENTIAL

Contents

1 Mission Statement

Anonymoose is a mobile app which provides highly localised, fully anonymous and self-moderating posting and chat features. Posts from within a five mile radius are shown to the user, who may choose to initiate a chat with any other user who posts within this range. Anonymoose will fully support text, images, audio and video posts.

The goal of Anonymoose is to provide a digital space for local communities, particularly university towns to interact in a completely free and open manner, with no connection to their real life identity.

2 UI and Feature Description

2.1 Units of Measurement

The unit of measurement shown on the distance indicator is based on the locale of the device's OS.

2.1.1 Mile

In the following locales, the distance indicator on each post shall be displayed using the international mile:

Territory	**Code**
United Kingdom	gb
United States	us
Liberia	lr
Burma	mm
American Samoa	as
Ascension Island	ac
Bahamas	bs
Belize	bz
British Virgin Islands	vg
Cayman Islands	ky
Dominica	dm
Falkland Islands	fk
Grenada	gd
Guam	gu
The N. Mariana Islands	mp
Samoa	ws
St. Lucia	lc
St. Vincent and The Grenadines	vc
St. Helena	sh
St. Kitts and Nevis	kn
Turks and Caicos Islands	tc
U.S. Virgin Islands	vi

2.1.2 Scandinavian Mile

In the following locales, the distance indicator will be displayed in Scandinavian miles (6.2 mi/10 km)

Territory	**Code**
Sweden	se
Norway	no

2.1.3 Kilometre

In all other locales, the kilometre shall be used for the distance indicator.

3 Technical Description

3.1 Anatomy of an Anonymoose Post

Post data is sent between the client app and the server using the following format:

Byte Offset	Data Type	Description
0	u8 array	The UTF-8 string "POST"
4	i64	a unique post identifier
12	i32	the UNIX timestamp of the post date
16	f32	the latitude of the post location in decimal degrees
20	f32	the longitude of the post location in decimal degrees
24	i16	the number of upvotes
26	i16	the number of downvotes
28	bool[12]	bytes for various flags
40	i64 array	null-terminated array of post IDs of replies
variable	u8 array	null-terminated UTF-8 string up to 256 chars
variable	raw bytes	any additional data (eg photos, videos, sound) terminated by EOF

3.2 Status Codes / AM protocol

Code	Description
100	initial connection
101	client submitting post
102	client submitting vote
103	client requesting deletion
104	client reporting a post
200	server standing by for IO
201	operation successful
202	closing connection
300	unspecified error
301	connection timeout
302	post format is corrupt
303	database error
304	voting error

3.3 Operation Description

Operation	Description
upvote	after code 102, a zero byte followed by the post ID
downvote	after code 102, a non-zero byte followed by the post ID
delete	after code 103, the post ID
report	after code 104, the post ID

3.4 Algorithms for Calculating Distance

3.4.1 Imposing the Five Mile Limit

'Under the bonnet', Anonymoose uses decimal degrees of latitude and longitude to calculate the position. The client reports its current position and the server returns the 30 latest posts within five miles. The algorithm for performing this is thus:

```
/*get the latitude and longitude from the location given by the client*/

float current_latitude = get_GPS_latitude();
float current_longitude = get_GPS_longitude();

/*degrees of latitude are approx. 69 mi and we want a limit of 5 mi*/

float min_latitude = current_latitude - 0.073;
float max_latitude = current_latitude + 0.073;

/*degrees of longitude are more complex, in order to impose our 5 mi limit we need
to use some trigonometry and the current latitude. First we get the length of a
longitudinal degree in miles*/

float miles_in_longitudinal_degree = cos(current_latitude) * 69;

/*now we need to figure out how many longitudinal degrees are in five miles
at the current latitude*/

float longitudinal_degrees_per_5_miles =  (1 / miles_in_longitudinal_degree) * 5;

min_longitude = current_longitude - longitudinal_degrees_per_5_miles;
max_longitude = current_longitude + longitudinal_degrees_per_5_miles;

/*assuming the database is correctly sorted in order of post times, we search it for
posts in the corresponding range*/

int counter = 0;
struct posts_array[30];
foreach(posts_in_database) {
  if(current_post.latitude < max_latitude && currentpost.latitude > min_latitude
  && post.longitude < max_longitude && post.longitude > min_longitude) {
    postsArray[counter] = current_post;
    counter++;
  }
  if(counter == 30) {
    break;
  }
}

return posts_array;
```