

Anonymoose

version 0.0.3

Anonymoose Industries Ltd

Aaron Walker, Alex Toop, Farouk Jouti and Joseph Thompson

PRIVATE AND CONFIDENTIAL

Contents

1	Core Features	2
2	General Architecture	3
2.1	Broadcast Range	3
3	Client Specification	3
3.1	General UI	3
3.2	General Client Backend	3
3.3	General Theme	3
3.4	Photo/Video Filters	3
3.4.1	Photographic Filters	3
3.4.2	Speedometer	3
3.4.3	Temperature	4
3.5	Language Support	4
3.6	Units of Measurement	5
3.6.1	Mile	5
3.6.2	Scandinavian Mile	5
3.6.3	Kilometre	5
4	TCP Server Specification	6
4.1	General	6
4.2	Operations	6
4.2.1	Bans	6
4.3	Web Interface	6
5	Use Policy	7
6	Protocol Description	8
6.1	Initial Connection	8
6.2	Posts	8
6.3	Status Codes / AM protocol	9
6.4	Operation Description	9
6.5	Algorithms for Calculating Distance	10
6.5.1	Imposing the Five Mile Limit	10

1 Core Features

Anonymoose is a mobile app which allows users to broadcast pictures and both live and recorded audio/video to their local areas. It is a combines the spontaneity of Snapchat-like apps with the anonymity and locality of YikYak-like apps.

- Users may post images, audio and video which will be seen by all other users in the geographical vicinity.
- Users may livestream audio and video to all other users in the geographical vicinity.
- Users earn points based on the popularity of their posts, users with more points can broadcast to a wider audience.
- Users can post either with a username or anonymously, and anonymous posts contain no identifiable data.

2 General Architecture

There are several components to Anonymoose:

- The TCP server is the only component visible to the client app, it handles incoming connections, serves the appropriate content and streams to clients based on location. It also allows operators to control the system including inappropriate content reports, setting up exclusion zones and managing the database and contents servers.
- The Database Server runs MySQL and is responsible for storing all persistent post and user information.
- The Content Server stores all non-textual information such as images and videos.

2.1 Broadcast Range

The list of possible broadcast ranges is 5mi, 10mi, 25mi, 50mi, 100mi, 250mi and 500mi, with rough km equivalents displayed in fully metric territories (the actual code will use decimal degrees). Sub-national (England, Wales, US states ect), national and supra-national (EU, Commonwealth Realms, North America) capabilities will be implemented for special posts such as events and unavailable for ordinary users.

3 Client Specification

3.1 General UI

The UI must display the posts on three separate feeds: the local feed, the home feed and the global feed. Each post will display the time passed since it was posted, the distance from the client in the local unit of distance and the number of total votes. Posts will be displayed sequentially (as in like Facebook Yik Yak or Instagram rather than Snapchat or Whisper). All toolbars and UI elements other than the posts themselves will move out of the way when scrolling down, showing a clean interface. A slider will allow the user to select the geographical range of posts they want to see.

3.2 General Client Backend

The backend must implement the Anonymoose protocol outlined in this document and communicate with the TCP server to load posts. It will be implemented in Java on Android and Swift on iOS.

3.3 General Theme

The stylistic choices will be largely white or transparent with a single, changeable RGB colour for emphasis. A "night mode" will replace white with black for more comfortable viewing at night, and change based on either a toggle or automatically (default behaviour is toggle).

3.4 Photo/Video Filters

3.4.1 Photographic Filters

There will be a few generic filters to change white balance, colour tone ect., and also the option to make custom filters along the same lines as Instagram.

3.4.2 Speedometer

This filter will display a white analogue speedometer dial with the current GPS speed of the phone, it will have mph on the outer dial and km/h on the inner dial.

3.4.3 Temperature

This filter will display the current temperature with a white analogue dial, it will have degrees Celsius on the outer dial and degrees Fahrenheit on the inner dial.

3.5 Language Support

The initial version of Anonymoose will be written in strict British English, avoiding Americanisms where possible. When ready for release, British English, American English, Spanish, French and German translations will be produced with more languages to follow. Welsh will be considered if the University offers support for Anonymoose. If a translation is unavailable for a locale, the app should default to **British** English.

3.6 Units of Measurement

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

3.6.1 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

3.6.2 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

3.6.3 Kilometre

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

4 TCP Server Specification

4.1 General

The server shall fulfil the client's requests as per the Anonymoose protocol outlined in this document. It will be written in Rust to take advantage of the language's high performance, ease of use and reliability. It will interface with a MySQL database and have the ability to modify this database whether it exists locally or on a separate server. The server will be multithreaded and easily scalable to run on modern cloud-based server solutions. The server will also provide a command-line interface for various administration tasks such as purging an area, creating a new database, creating 'autopurge' zones for schools ect., and reporting/banning users.

4.2 Operations

In general, the syntax follows UNIX norms. A zero entry for radii generally causes an operation to be global, and the server should warn the operator before any global changes are made.

Action	Options	Description
ban	ban type (-id, -n, -ip, -im)	bans a user from posting
purge	centre (-c), radius (-r)	deletes all posts in the radius (miles)
autopurge	centre (-c), radius (-r)	deletes all existing and future posts in the radius (miles)
promote	PostID (-i), centre (-c), radius (-r)	raises a post transparently in the defined area
populate	path to XML	loads sample data into the database
search	(string), centre, (-c), radius (-r)	looks for a post or posts in the database
dump	PostID (-id)	prints the contents of a post to the screen
edit	PostID (-id)	allows on-the-fly editing of posts
delete	PostID (-id)	removes a post from the database
reports	centre, (-c), radius (-r), mode (-s, -f)	dumps reported posts and allows deletion
bot	add (-a), remove (-r)	loads and removes bot configurations
scram	shutdown (-sh), secure (-sc)	purges all data indiscriminately

4.2.1 Bans

The server shall implement bans of differing severity depending on the offence, these may be activated either manually or automatically by the server.

- UserID bans prevent a certain UserID from posting. These bans have a maximum length of 24 hours as all UserIDs are reset periodically to retain anonymity.
- Username bans prevent a certain username from posting, affecting both 'loud' and anonymous posting.
- IP bans prevent a certain IP address from posting, these are usually ineffective but are useful for denying organisational networks access, which can be used alone or in combination with an 'autopurge'.
- IMEI bans prevent access to the entire device. These are an obvious 'nuclear option' as keeping records of IMEI numbers is extremely detrimental to privacy. The server and database must therefore **ONLY** store a list of banned IMEI numbers, not collect them from all users.

4.3 Web Interface

In time, a graphical web application to control the TCP server will be developed, however the command-line interface must always remain complete and up to date.

5 Use Policy

The rules of the service

- All content that is not safe for work must be marked as such.
- No conduct that threatens the integrity of the service.
- No threatening, harassing or unnecessarily aggressive conduct.
- No conduct which could lead to any user's identity being revealed against their will.
- No marketing activities without the agreement of Anonymoose staff.

The Anonymoose Privacy Policy

Anonymoose makes no use of personally identifiable data besides usernames. The only data we could provide if compelled or compromised is listed below. We will only release the data to a third party if one of the following conditions are met:

- Anonymoose is compelled to release the data by a court order in the United Kingdom.
- Anonymoose staff feel there is a clear and present danger of an Anonymoose user committing a serious criminal offence or placing themselves in mortal danger.

At Anonymoose we feel strongly that free speech is an important right, and as a result we refuse to co-operate with any third party seeking to censor, astroturf or otherwise corrupt our service unless ordered to by the courts in the United Kingdom.

In the interests of transparency, the data stored by Anonymoose beside the post content is simply a latitude, longitude, timestamp and a unique post identifier which cannot be tied to a user. Anonymoose relies on a unique user ID to differentiate the original poster and allow chat, these are randomly generated and are not based on your real-life identity. We do not log IP addresses except those blacklisted from our servers. All chat sessions are encrypted to at least 2048 bits.

6 Protocol Description

Anonymoose uses a custom protocol to transfer post data between client and server. This is to save bandwidth on the limited connections often encountered in the mobile world.

6.1 Initial Connection

Byte Offset	Data Type	Description
0	u8	The UTF-8 string "INIT"
4	i64	unique user ID
12	f32	decimal latitude
16	f32	decimal longitude
20	i16	range in miles, 0 for global
21	i8	connection type
22	u8	user name as UTF-8 string

6.2 Posts

Post data is sent between the client app and the server using the following format (all numbers are serialised in the big-endian format):

Byte Offset	Data Type	Description
0	u8	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
34	i64	post ID of parent comment (0 if top level)
42	i64	unique user ID.
50	i64	content server number
58	u8	256 UTF-8 chars

6.3 Status Codes / AM protocol

Code	Description
100	initial connection
101	client standing by for IO
102	client submitting post
103	client submitting vote
104	client requesting deletion
105	client reporting a post
106	acknowledge IO
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
305	user is banned

6.4 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

6.5 Algorithms for Calculating Distance

6.5.1 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;
```