Requirements Document - Skwad

1. Introduction

Skwad is an iOS mobile application that makes use of GPS location to bring together young adults in an environment where they can take part in anonymous chat rooms with people in their area. These anonymous chat rooms, henceforth be referred to as packs, can be admin or user-generated and have a small number of customizable features: the name of the pack, the radius of the pack, and whether or not the pack has a geofence. Admin-generated packs will have no time limit. However, user-generated packs will have a time limit of 24 to 48 hours depending on their popularity. Currently, the scope of the project only involves user-submitted text posts.

2. User Requirements

2.1 Software Interfaces

- Firebase database
    - Store information about packs and posts in the cloud
- Backend
    - Google Cloud Backend
        - Manage Firebase database
        - Flush out expired packs and posts
    - Raspberry pi
        - Manage Firebase database
        - Flush out expired packs and posts
        - Run in M5 and possibly additionally someone's house
- Realms
    - Store post/pack information locally to reduce bandwidth
- Geofire
    - iOS geolocation library to handle all location requests and computations
- Any necessary external Swift libraries for the iOS application

2.2 User Interfaces

The general goal for the user interface is to have it be minimal, clean, and easy-to-use. A user can create a pack, choose the settings for that pack, and become the Alpha. Users can also manage the packs they are a part of, post content in any pack they are in, and browse for relevant or popular content near them. In the UI, there will be four main tabs for interaction:

- Feed
    - Lists new posts that have been made in your packs
    - Can also list posts by highest rated posts in your packs, based on algorithm considering time and upvotes
    - Each post will show the pack that it was made from, the icon and color for the pack it was posted from, the title of the post, the post's content, and the number of likes it has received
- Discover

    - Find packs or public posts. Able to sort by distance from your current location, popularity, creation date, and tags
- Create
    - Able to create a new pack, or a new post within a pack that you have joined
    - Pack Creation: You are the alpha, and must choose tags and color for the pack. You can also choose whether the pack will be locked or unlocked to users outside the radius and how large of a radius you would like your pack to extend to.
    - Post Creation: You choose which of your packs you would like to create a post in, then create the title and body of the text post. You can also choose certain tags to be associated with the post.
- Den
    - This is the user's "home." Here, they will be able to look at any posts they have made and any comments they have posted. They will also be able to see how many upvotes they have accumulated and all the packs they are a part of.

2.3 User Characteristics

The intended audience of the product is young adults, ideally in high school and college environments where their use of our application can spread to many people through word of mouth. No technical expertise is required, as using our application will only require the application to be downloaded, and from there there will be a simple to use GUI that guides users through posting and the creation/joining of packs. While young adults are our target audience, we are not limiting our scope to just them. Our application will be open to anybody who is trying to make social connections with people who are in their area or share the same interests or hobbies as them.

2.4 Assumptions and dependencies

We are assuming that anyone we are targeting with our user base will be interested in our application, and that they will have access to an iPhone with an internet connection and the App Store. We are also assuming that a sufficient amount of people will download and use the application.

Throughout the design of Skwad, we will be dependent on our ability to integrate Google App Engine for our back-end storage, a Raspberry Pi for periodic server management and deletion, and Swift for the front-end iOS application.

3. System Requirements

3.1 Functional Requirements

One main functional goal of Skwad is to maintain anonymity. Each text post should remain anonymous, and the user's identity should never be revealed to other users in any way. Another functional goal is to keep packs secure, ensuring that nobody located outside of a locked pack can see the contents that are being posted within it. We would also like to design Skwad to require little to no ongoing maintenance, since we expect the revenue that it generates to be relatively small. Skwad should also handle the sorting and filtering of search results via an algorithm that takes popularity, time and distance into account. The software should flush out posts and packs that have expired.

3.2 Non-Functional Requirements

One non-functional requirement we have set for ourselves is to launch version 1.0 on the App Store in November.

3.2.1 Software Quality Attributes

One important software quality requirement we have is for Skwad to have a clean, attractive and simple-to-use interface. We would like the interface to be intuitive for anyone who is at least somewhat familiar with the design and layout of typical iOS applications. This will help us retain users and can be tested by our team and through demonstrations to first-time users of the application. Another software quality attribute that we would like to enforce is a good user to server ratio. Our servers will be hosted by Google, and we will monitor the connection speed to ensure that simultaneous connections are not bottlenecking our servers. This ties into the reliability of our application, and we have a goal to have the application and servers accessible 99.9% of the time, which should be possible given our use of Google's servers. The reliability and uptime will be monitored via Google App Engine.

We also want our software to be secure, since it will be used by many people and the consequences of an exploitation can be bad. Although our database will not store any sensitive data, we do not want anyone to get arbitrary access to our data. We will follow secure coding practices such as input validation and sanitization and use strong database access rules.