

Software Design Document

1. Introduction

Skwad is a mobile application that makes use of GPS location to bring together users 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, whether or not the pack has a geofence, and the radius of the pack in the case of an enabled geofence. User-generated packs will have a time limit of 24 to 48 hours depending on their popularity, while admin packs are permanent. Currently, the scope of the project only involves user-submitted text posts.

2. Design Considerations

2.1 Assumptions

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.

2.2 Constraints

One constraining aspect of our system would be our backend, which will only provide as many interactions as we are able to pay for with our cumulative \$250 in Google App Engine credit. When the limits of our \$250 are met, the group will need to decide whether the service should be shut down or if more money should be contributed to keep the service running. The group's ability to pay for Google App Engine will also determine the amount of persistent disk storage that we are able to have on the server.

2.3 System Environment

For the front-end of our system, the software will be an application running on iOS written in Swift. This application will provide the user with a UI which allows for interaction with the system, including viewing posts and packs that have already been created in addition to being able to create new posts and view the user's history of anonymous activity. The aforementioned posts and interactions will be stored and managed through our backend, which will be written in Python and at this point is planned to be Google's App Engine. Google's Cloud Platform, within App Engine, will allow for the storage of all users' posts and packs. The hardware and software that comprise Google's App Engine will handle the storage/management, and our team will choose the settings and configuration for the App Engine via a web interface. It will also handle all the sysadmin work for us, as it will auto scale our physical servers to our needs. The text posts will be stored in a NoSQL database as it is simple, yet still powerful, compared to SQL.

3. Architectural Design

3.1 Overview

The basic architecture of Skwad can be seen in Figure 3.3.1. The architecture consists of four major components: the user-interface(UI), client, server, and database. These components and their respective responsibilities are listed below.

UI - provide a minimalistic and easy-to-use interface to the end-user for interacting with the client.

Client - call functions from the server and provide the returned data to the UI

Server - expose functions able to be called remotely from the client and query the database

Database - store application data in an indexed table and provide a way for the server to query this data

3.2 Rationale

We are using this architecture because it provides a centralized database to store and retrieve our data which we can use to sync users. It is important that all the users are synced to the same database so they can see other users create new posts, comment, vote on posts, create packs, and see packs expire.

3.3 Conceptual (or Logical) View

The component diagram for the system can be seen in Figure 3.3.1 below. The system follows a standard client-server relationship in which we have a centralized server that handles the communication between multiple clients and the database.

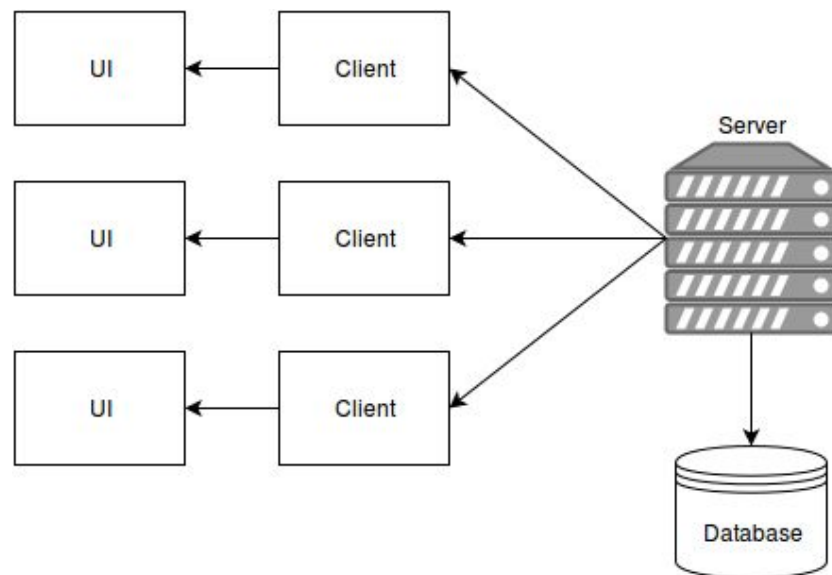


Figure 3.3.1

The deployment diagram for the system can be seen in Figure 3.3.2. The client and UI physically reside on iOS devices since we are developing our app for iOS only. Also, the server exists in another location and runs code to manage the database. The iOS devices will connect to the server through WiFi or a cellular connection.

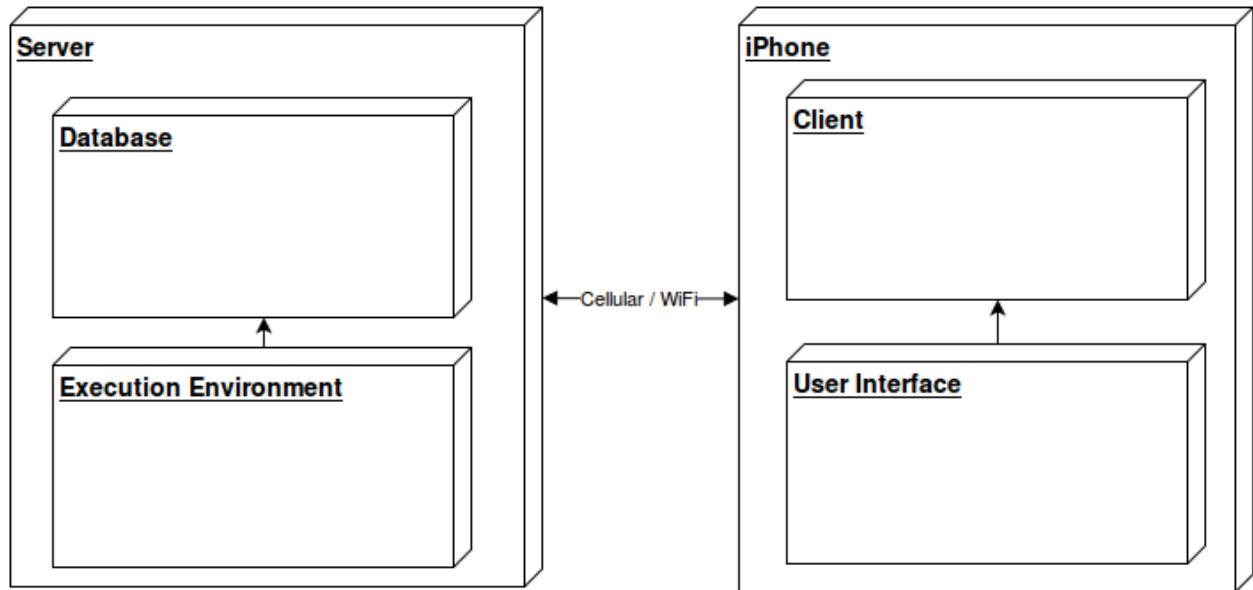


Figure 3.3.2

4. Low Level Design

4.1 Class Diagram

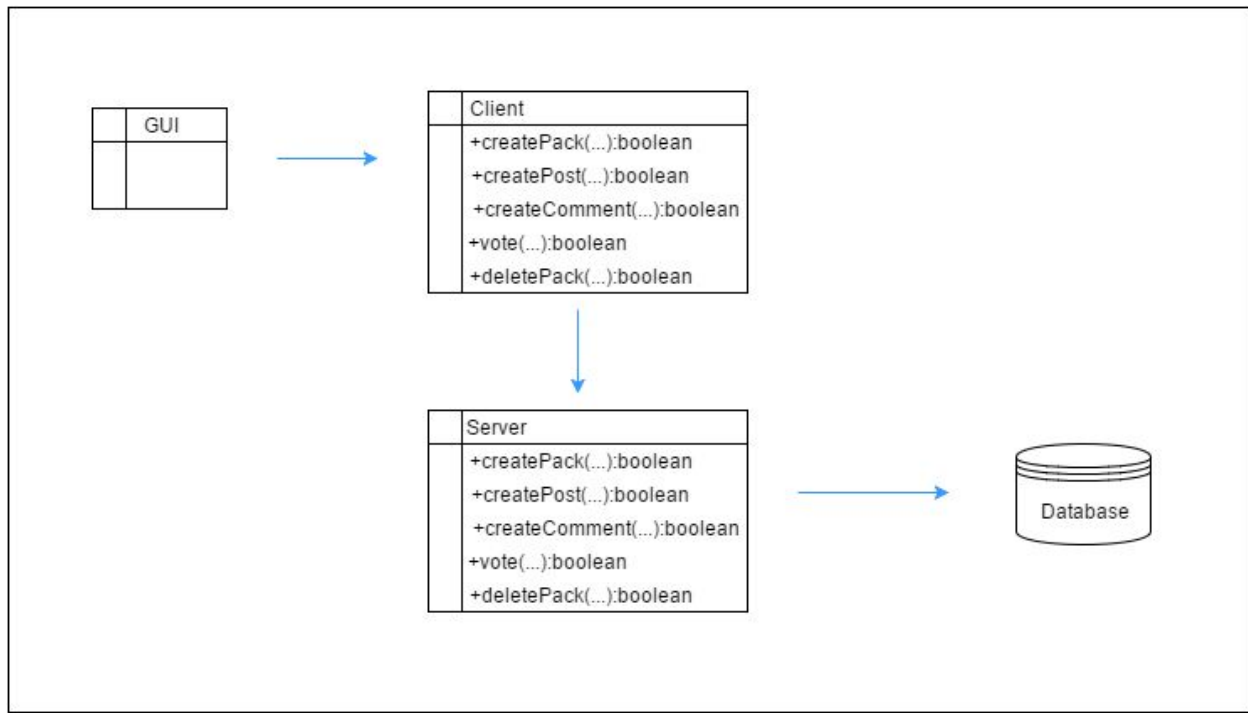


Figure 4.1.1: Class Diagram

4.2 Sequence Diagram

Principle Action: Create/Delete Pack/Post/Comment

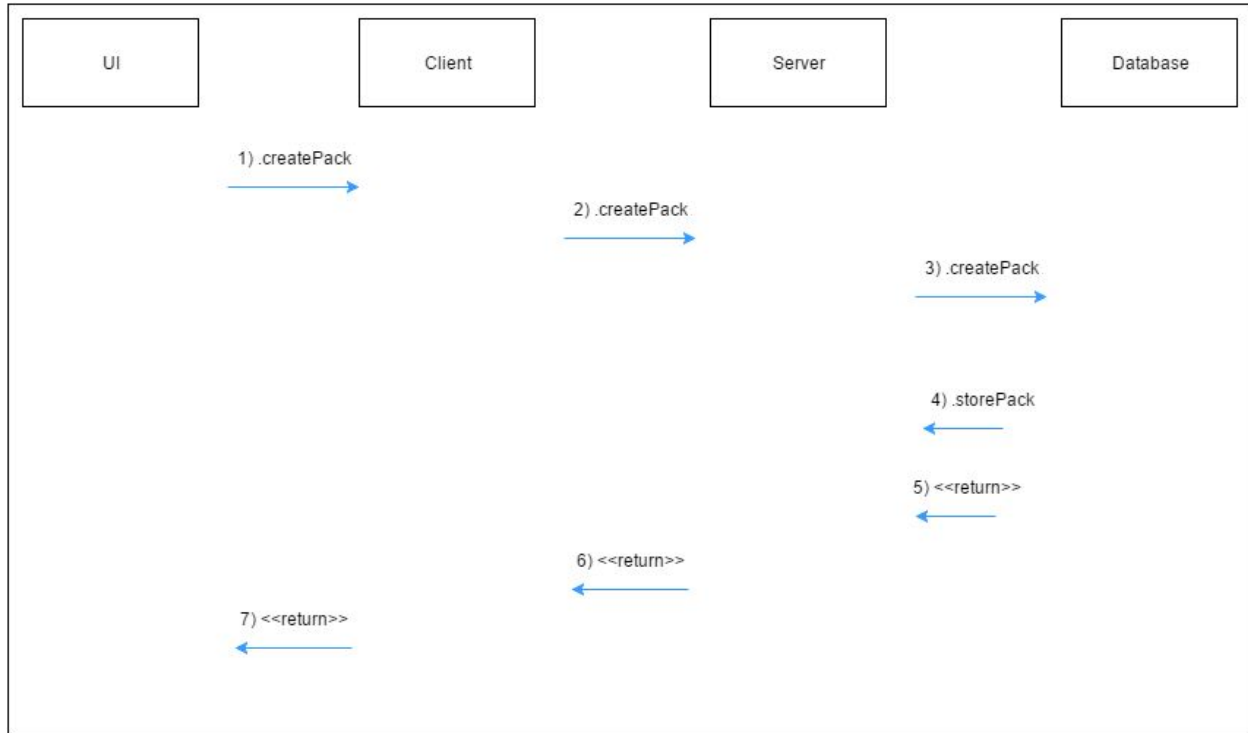


Figure 4.2.1: Sequence Diagram for Create Pack/Post

The sequence diagrams for creating a post, voting on a post, creating a comment, and deleting a pack are identical to the sequence diagram above, with their respective function names.

4.3 State Diagram

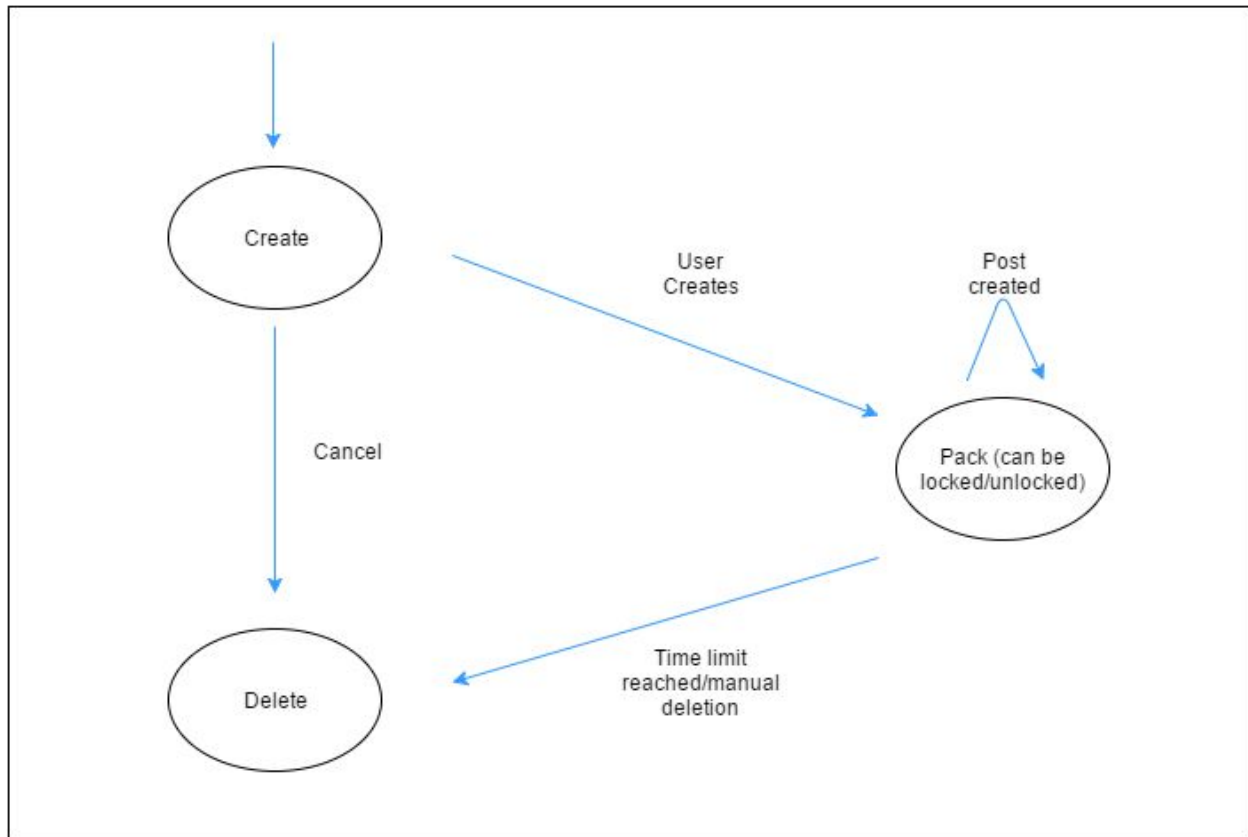


Figure 4.3.1: State Diagram

5. User Interface Design

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 (Figure 5.1)
 - Lists new posts that have been made in your packs
 - There will be a button at the top left of the UI that will allow the user to decide which packs they would like to see posts from in their Feed and how they would like to sort their Feed, by time or highest rated posts. (Figure 5.3)
 - 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
 - There will be a button on the top right that will allow the user to create a post in a pack. The user chooses which of their packs they would like to create a post in, then create the title and body of the text post. They can also choose certain tags to be associated with the post. (Figure 5.2)
- Search Figure (5.4)

- Find packs. Able to sort by distance from your current location, popularity, creation date, and tags
- There will be a button on the top right that will bring the user to a map that shows all pack currently around.
- Create (Figure 5.5)
 - Able to create a new pack
 - Pack Creation: The user is the alpha (creator), and must choose tags and color for the pack. They can also choose whether the pack will be locked or unlocked to users outside the radius and how large of a radius they would like their pack to extend to.
- Den
 - This is the user's "home." Here, they will be able to look at all the packs they are a part of, 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.

Figures 5.1 to 5.6 show examples of how Skwad's UI currently looks and how we plan to have the future "Den" tab look. This is not the finalized UI.

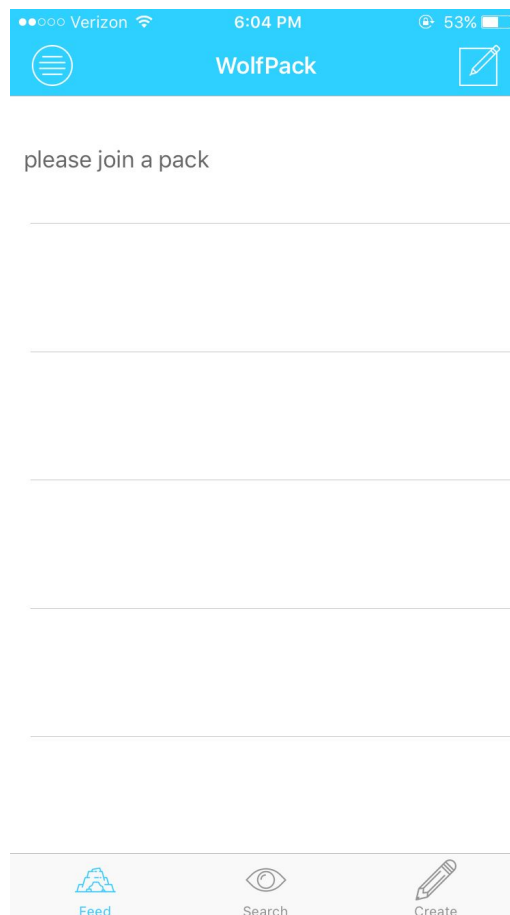


Figure 5.1: The Feed UI where the user is a part of no packs.

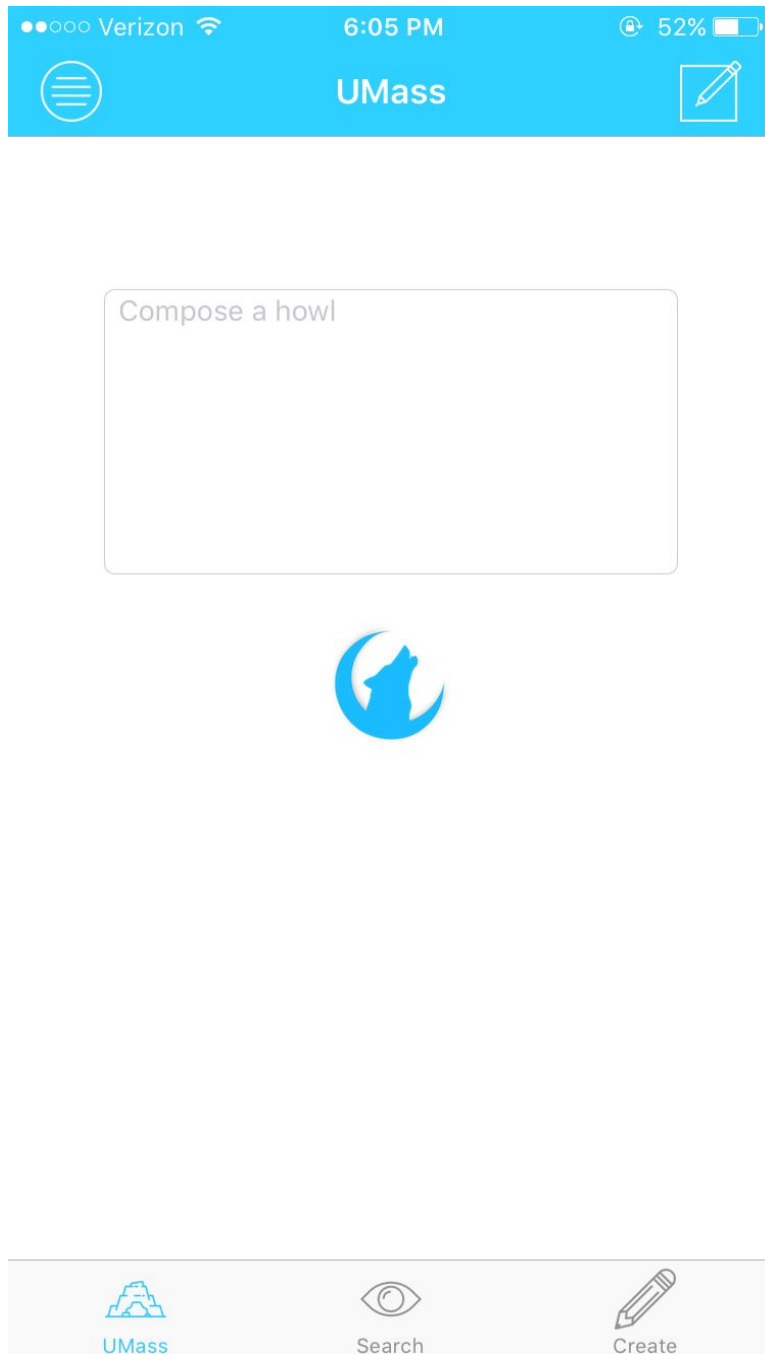


Figure 5.2: Creating a post in the pack "UMass."

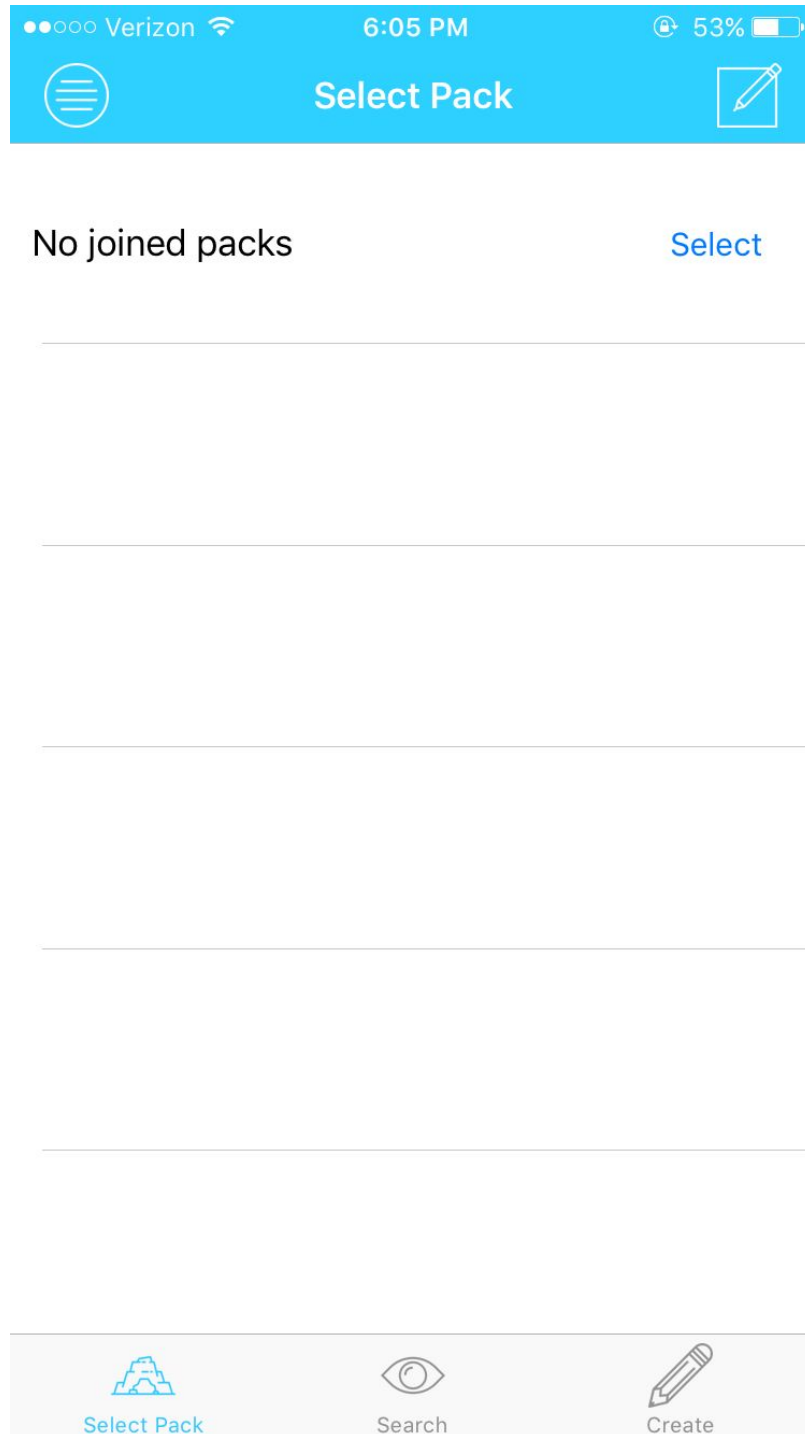


Figure 5.3: Choosing the pack to view posts of said pack from a list of packs the user is a part of.

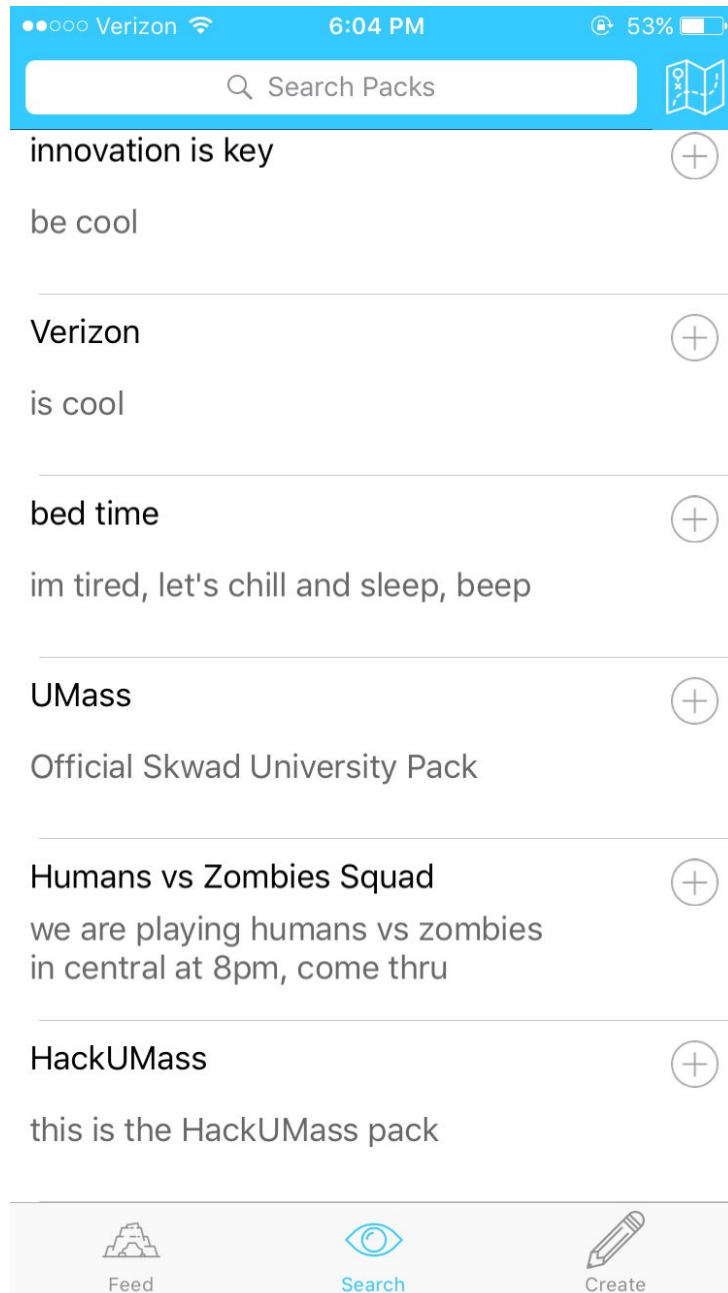


Figure 5.4: The Search tab.


●●○○○ Verizon 6:04 PM 53%


Create Pack

Title of pack

Short description of pack

Create

Feed

Search


Create

Figure 5.5: The Create tab.

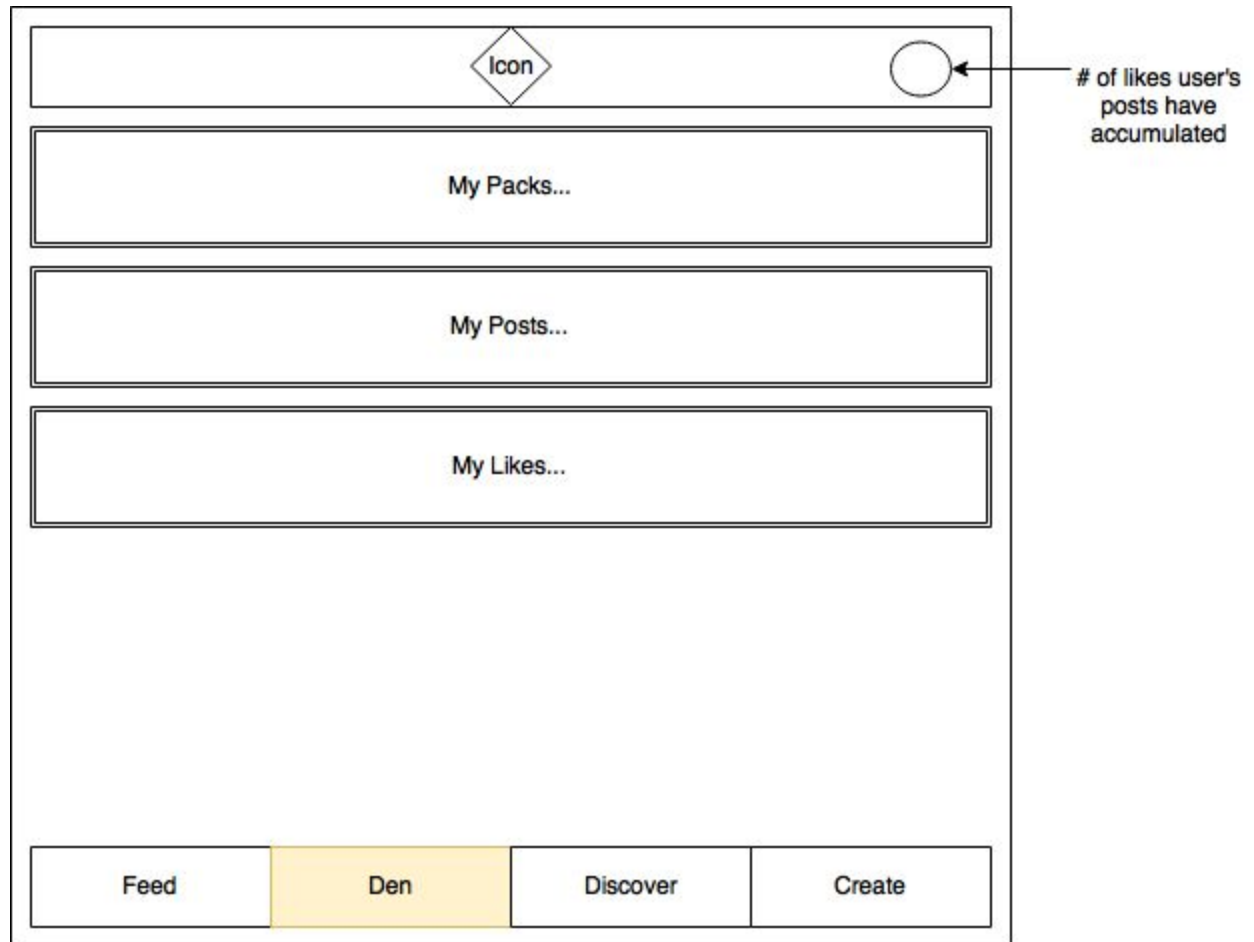


Figure 5.6: The Den tab.