Squirrel.IO: A Social Media App about Squirrels for Students

Fabrizzio Perez, Michael Amaya, Alex Chen, Jessica Wu, Andrew Hong

Overview

Aside from the students and faculty, one of the most iconic figures on campus are the squirrels. With their comical and personable behavior across campus, many students, including ourselves, grew a liking to these little creatures. This cult following for the squirrels can be shown in the countless Reddit posts and even Instagram accounts dedicated to these squirrels. Instead of having these posts scattered across multiple social networks, we wanted to create a central hub to appreciate these squirrels. Our app, Squirrel.IO, solves this problem by hosting a new social network dedicated to these creatures.

The functionality of the app mimics the functionality of other social networking apps like Instagram, but the posts found in our app only contain squirrels. Users can create their own accounts with a unique username and profile picture. From there, the user can begin posting their own pictures of squirrels using their own camera, or by uploading their own photos, along with their own captions. Once posted, other users of the app can view the user's posts and interact with them, either by liking/disliking the post, or providing a comment.

Goals

The functionality of this project was divided into three major milestones: milestone one, milestone two, and milestone three. Each milestone made progressive improvements to our app. Milestone one was focused on developing the UI without any functionality. Milestone one tasks included developing the landing page where the user is able to sign in and/or create an account, creating the home page that displays a scroll view of squirrel posts filled with dummy data, and account functionality where users can create accounts, sign up, and sign up where user information is stored on Firebase. Milestone two focused more on Firebase and the ability to read and write data. These tasks included the ability of the user to post images and the ability to create a scroll view by loading data from the Firebase database. Milestone three focused on increased functionality and interaction in the app. These tasks included deleting posts, interacting with posts through liking/disliking and commenting, and providing a separate scroll for only a user's own posts. We go in further detail of the main goals below.

Along with our three milestones, we also had stretch goals we wanted to accomplish if there was enough time. Stretch goals included filtering profanity out comments, removing posts that do not contain a picture of a squirrel, a daily squirrel of the day where we highlight a popular squirrel post, infinite scrolling of posts, and push notifications to the user.

Cloud Firestore

In order to make a full-stack iOS application, we needed to have some sort of backend to store and maintain our information. Since we learned Firebase in class, we decided it would be best to use this platform, but more specifically, we decided to use Cloud Firestore. Firestore is a NoSQL database that's extremely flexible and allows for simple, effective querying and real time updates. Also, because of its vast and detailed documentation, using Firestone would be simple to incorporate into our app.

Account Creation

Account creation was part of Milestone 1. One major component of any social media app is the ability to make accounts. Accounts allow each user to have a personalized experience. When a user enters the landing page, they have the ability to create an account using our own account creation page, through their Apple account, or their Google account. Multiple options to create an account provides greater flexibility and accessibility of our app to more users. To maintain and store our account, we used Google's Firestore, a mobile development application that helped us manage the backend of our app.

When the user logs in using their credentials, they are able to view their profile picture and account information on the Account tab.

Posting Photos

Another major aspect of any social media app is the ability to share photos. Once the user is logged in, they can add a new post, which requires both a photo and a caption.

Location of Posts

For the GPS requirement, we wanted to incorporate this when creating and viewing a post. Using CoreLocation, we are able to store the user's location (with their permission) when making a post in Firestore. By doing this, we can display the location of each post on the feed.

Displaying Feed

The main aspect of social media is viewing other posts. We wanted to accomplish this in a feed view. By fetching data from the Firestore database, we can obtain each post's information such as the uploader, the upload date, the photo, and the location. With these, we can display the information on the feed. Furthermore, we wanted to sort posts by the upload date in descending order.

Like/Dislike System

To incorporate interactions into our app, we decided to embody a like/dislike system. Likes and dislikes allows users to express their satisfaction or dissatisfaction towards posts. Unlike many other social media apps, we allow users to dislike posts since it can allow us to obtain a better response for each post.

Squirrel of the Day

This was one of our stretch goals. To encourage constant engagement from users, we wanted to have a "Squirrel of the Day" where we display one squirrel post that we liked the most. This would encourage users to make more posts of high quality.

Demonstration

When you first download Squirrel.IO, you're met with a login screen. Here, you can sign in with your Google account or create a new account.

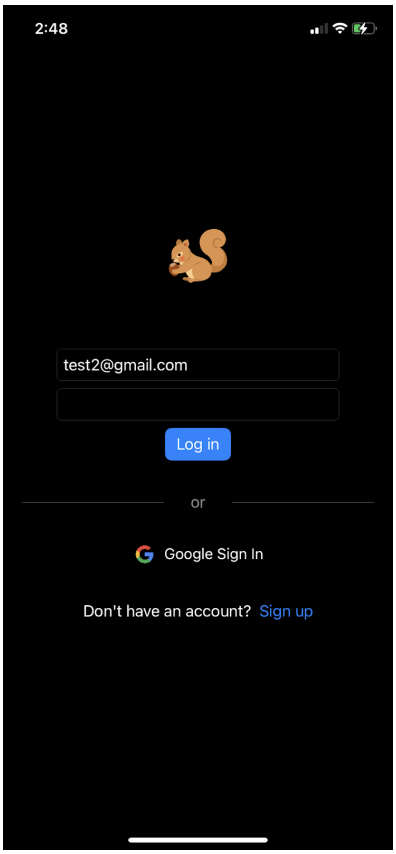


Figure 1: log in

Upon using Google Sign in, you will be met with a location permission request, where you can select what you choose.

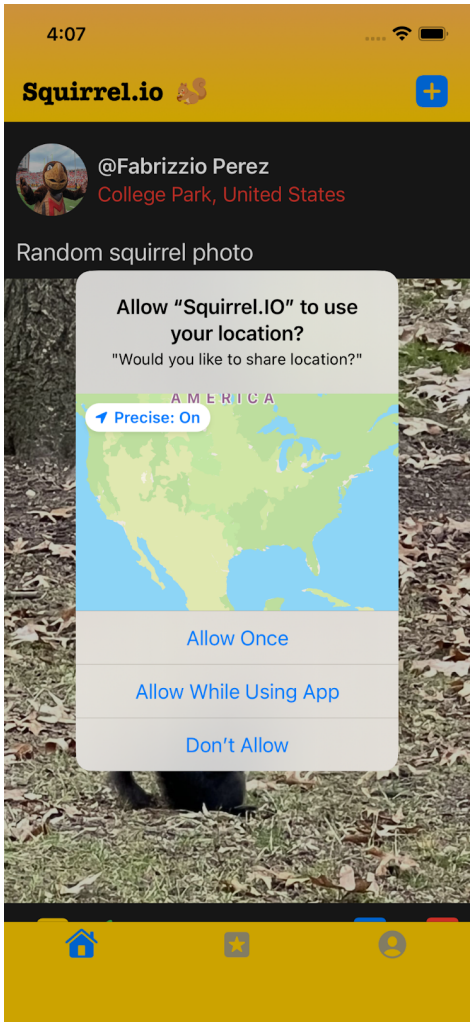
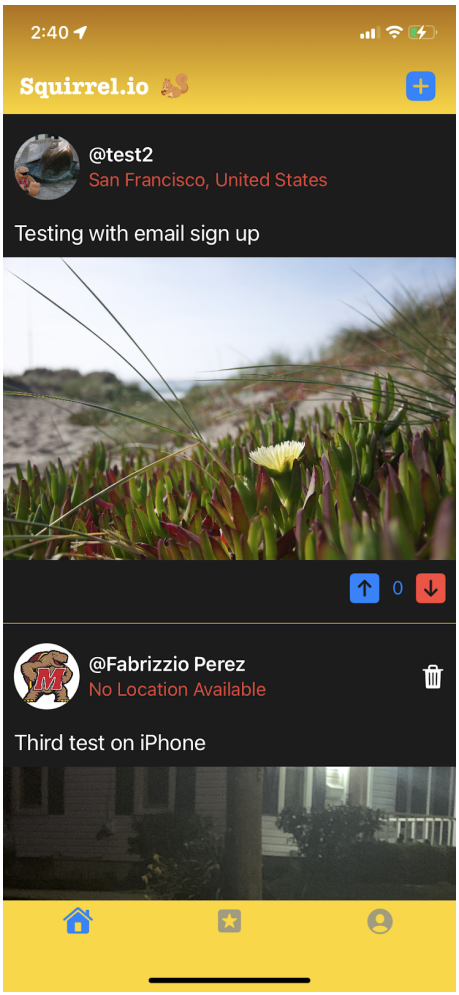


Figure 2: location permission

After selecting your location preference, you'll land on the first tab. You'll know which tab you're on because it'll be highlighted. In the home page, you can scroll infinitely to see posts around you of squirrels. Above each photo, the post should contain the user that posted, the location, and a caption. Below the photo, you'd be able to upvote and downvote depending on user preference.



Figures 3: home page

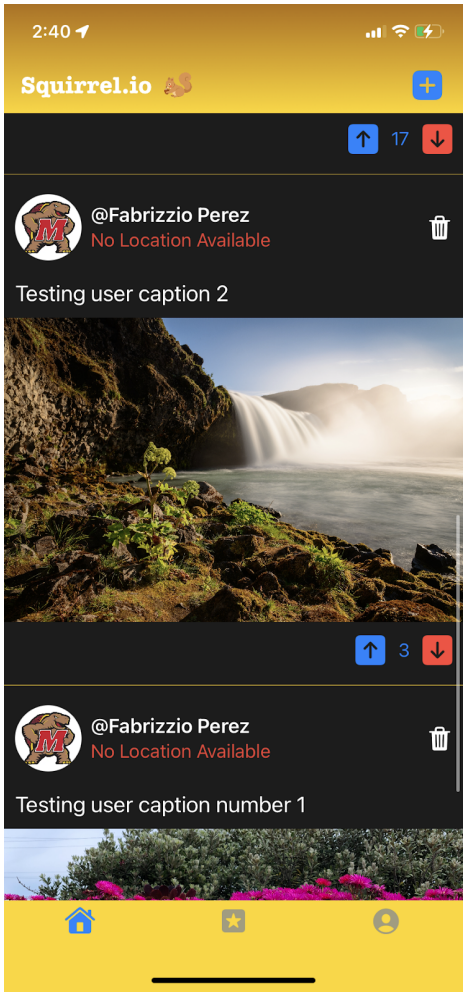


Figure 4: home page scrolled

On the top right hand corner of the home page, we have a plus button. This button will allow you to create new posts on your account. After this is clicked, you are granted a new blank page for your post. At the bottom of the page, you're given the option to take a new photo, choose a photo in your camera roll, and add your location.

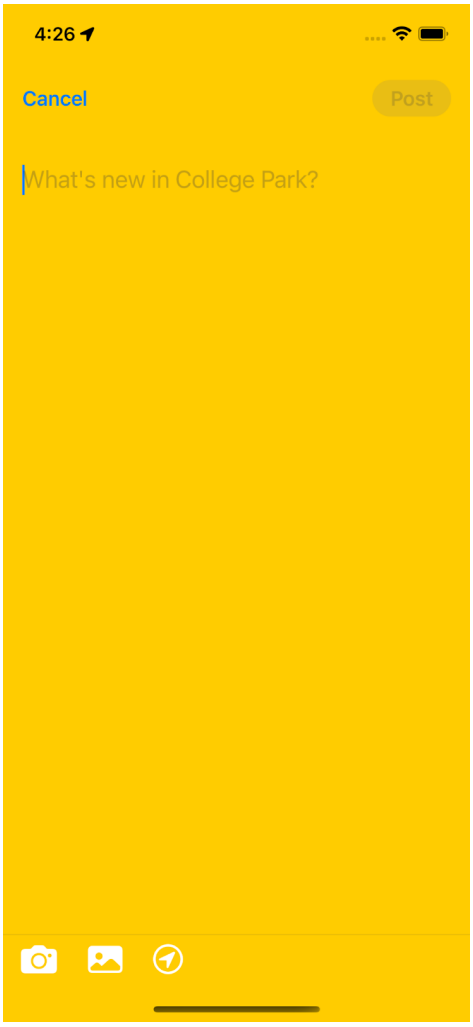


Figure 5: new post page

Figure 6: take a new photo

After choosing the photo you want uploaded, feel free to add a caption. Once satisfied with your post, you can post it by clicking the “Post” button in the top right.

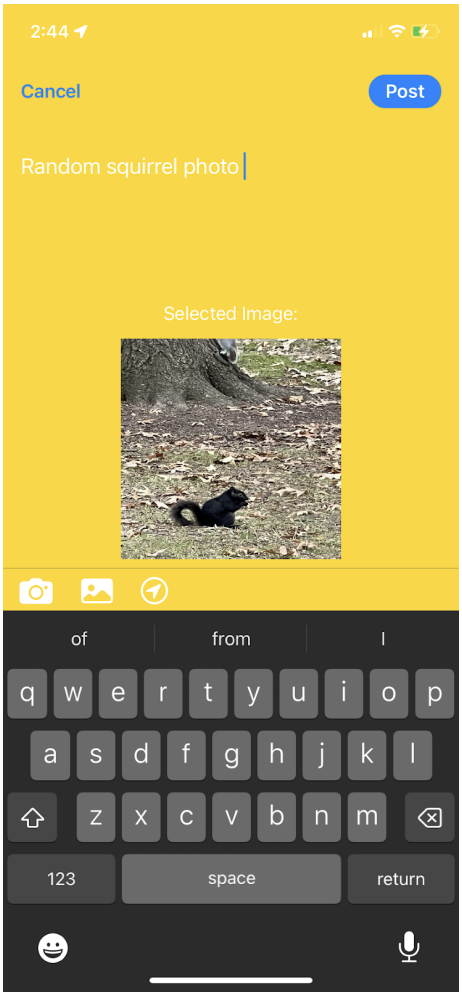


Figure 7: drafted post

Once your photo has been posted, a banner will appear towards the top of your screen to notify you that your post has successfully been posted.

Figure 8: successful post

As you can see in Figure 9, this is your new post and there are 0 upvotes and downvotes. In addition, as the original poster of this photo, there is a little trashcan next to your account in the event you want to delete your post.

Figure 9: new post on feed

After posting, users have the ability to upvote and downvote. For example, in figure 10, the post has been upvoted 6 times.

Figure 10: upvote example

In our second tab, we feature a “Squirrel of the Day!” This page features a squirrel posted with the potential of being from any user. Having this tab boosts user interactions and gives an incentive to users to try to get their post featured. You can see what this tab looks like in Figure 11.

Figure 11: squirrel of the day!

Because our own photo was featured as the squirrel of the day, we're able to delete it if we want. However, that is not the case if the post is from a different user.

In our third tab, we present the current user's profile. Here, we're viewing Fabrizzio's profile. There are multiple options we can perform on this page.

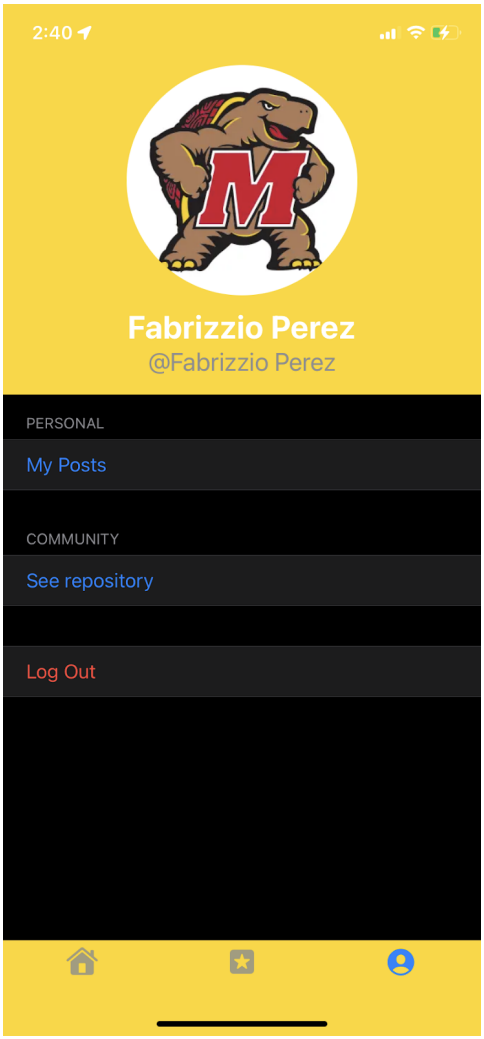


Figure 12: profile tab

If we click on “My Posts”, you can view all of your posts. In this case, we’re viewing all of Fabrizzio’s posts. Here, we can choose to delete posts or see how well the posts have been doing in terms of upvotes and downvotes.

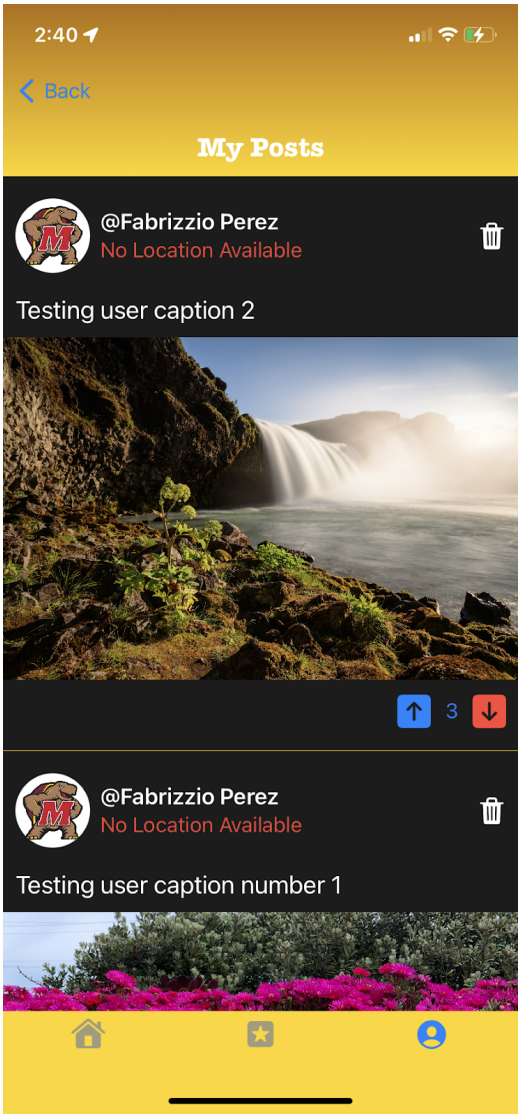


Figure 13: my posts

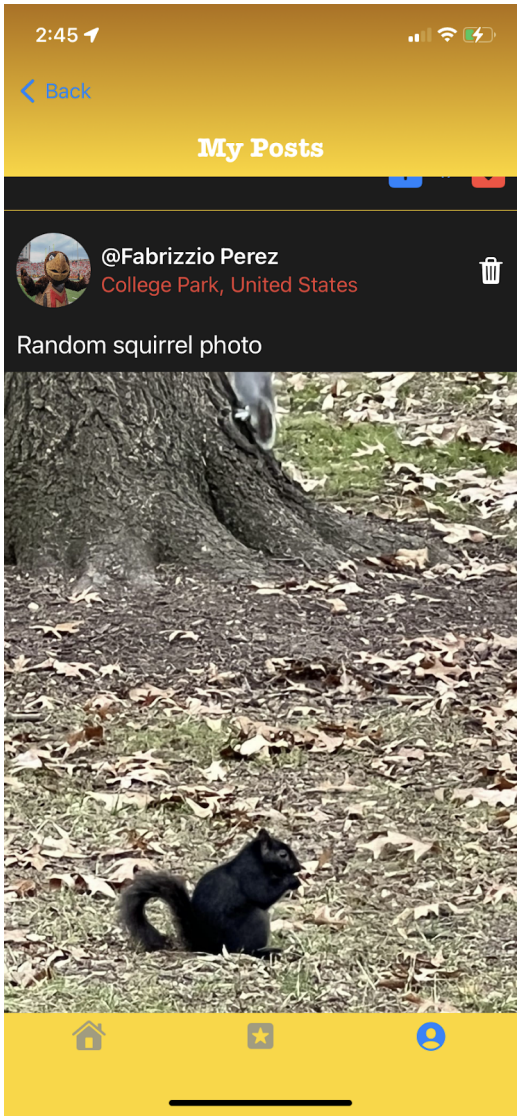


Figure 14: my posts scrolled

Posts are filtered to be shown from least recent at the top to most recent at the bottom. You can go through your own posts by scrolling down.

Finally, users will be able to log out of their account by clicking the “Log Out” button in Figure 12. Upon clicking the button, you will be prompted with an alert to confirm that you want to be logged out.

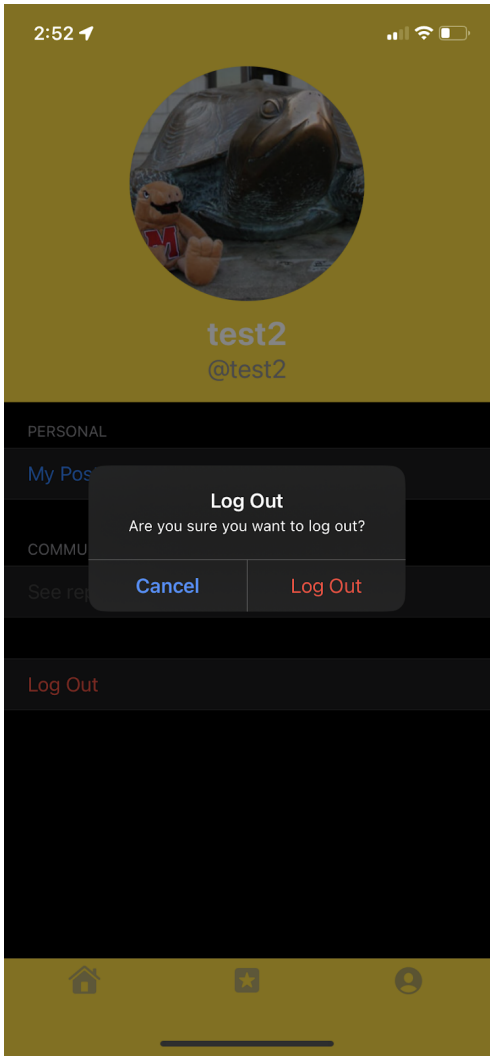


Figure 15: Log out

Development Process

Our development process was modeled after the agile methodology used in the software engineering industry. We felt that this approach would emphasize flexibility and collaboration among team members throughout the project lifecycle. To manage our workflow, we created a Kanban board using Notion, which allowed us to propose, assign, and track tasks for team members. This project management tool helped us to stay organized and ensured that everyone was aware of the status of various tasks while working asynchronously. All team members created branches off of the main branch, and for every push, a pull request was created for any other team member to review and merge into the main branch. This approach allowed us to collaborate on changes and ensure that all code changes were properly reviewed before being merged into the main codebase.

One of the key principles of the agile methodology is to work in an iterative, incremental manner, rather than attempting to complete the entire project sequentially or at once. This approach allows the team to continually refine and improve the project as it progresses, and allows for a faster overall development time, which is appropriate given our time constraints. As such, we tried our best to adhere to this principle and feature work was really go-with-the-flow. We worked on what felt appropriate at the time and added tasks to the Kanban board as we thought of them.

Naturally, there were challenges in our development process, but we did an overall great job at scoping out and defining attainable goals in our milestones. Therefore, we didn't run into any major roadblocks in feature development, but instead from the development process itself. As we mainly worked on this project during our free time, the team operated on varying schedules, which sometimes made it challenging to stay synchronized. Avoiding merge conflicts and ensuring that all team members could build the project locally by setting up file dependencies were some of the issues that we faced. Despite these obstacles, we made a concerted effort to communicate regularly and proactively address any issues that arose, allowing us to work collaboratively towards our shared objectives.

There were a few features that we implemented in our app that were not covered in this class. These features largely had to do with photos and implementing camera and photo library functionality into our app. We were surprised to learn that there was no native camera and photo library support in SwiftUI. Instead, we needed to rely on UIKit and modeling that in SwiftUI. Luckily, Apple thought about this and gave us the UIViewControllerRepresentable. Basically, this allowed us to use UIKit features, like accessing the camera and photo library, in SwiftUI. There were some challenges here as it was not entirely easy to do so. You need a model that handles this integration with SwiftUI. However, it worked in the end and switching with the camera roll and the camera itself ended up being a change to an enum. It helped us understand how we could implement other UIKit features in SwiftUI.

Nixxed Features

One of the features we did not implement is Apple Sign In. Usually, signing in with Apple is required if you use other sign in methods like Google, Facebook, etc and if you want the app to be on the app store. Unfortunately, we ran into numerous obscure errors that we could not solve by troubleshooting. We looked at numerous online resources and it seemed that there was something missing that we needed to implement. In the end, we removed the "sign in with Apple" button. It would have been nice for users to sign in with Apple as an option, but the constant issues meant we had to exclude it. Another feature we wanted to add was comments. The main reason we didn't do this was due to time constraints. We would have had another collection that linked each comment with a post. Querying wouldn't have been too hard either. However, there was also the logistical error of if people would have the ability to comment on other comments, how far would these trains go, etc. It would have had to have been implemented in some obscure way. In the end, the time constraints prevented us from implementing it.

Future Directions

While the focus of our app is niche to college campuses, it is definitely a platform that many students can relate to and choose to share on. From what we've completed, students that are using this app are able to share photos of different squirrels on their campus and connect with new students. In addition to this, students can boost posts to make the platform more interactive.

Though we have finished out benchmarks for the app and included some stretch goals, there are certainly many more improvements and possible future paths for Squirrel.io. Despite having set quite a few stretch goals, we were unable to achieve all of them due to time limitations. One such goal that we were interested in implementing was push notifications, which is a fundamental feature in any social media platform–it is an essential way to have users remember that our app is still active and to check in and post a squirrel picture! The inability to deliver this feature was a bit disappointing, as push notifications are an important aspect of user engagement, keeping them informed and up-to-date with the latest activities on the platform. Due to the nature of push notifications, which require a service provider (usually a web server) to pass through the Apple Push Notification (APN) servers and target multiple devices simultaneously, we encountered some unexpected complexities during the setup process. This included working with the Mac OS Keychain Access app and navigating the Apple Developer Member Center portal, which added an additional layer of intricacy to the process. In addition to notifications, we were hoping to add a filtering posts functionality. The purpose of this would be to ensure that appropriate pictures of animals were being posted. To implement this, we would have needed to create some sort of computer vision or machine learning algorithm to dictate whether a posted image contains a squirrel (or an animal in general). A simpler alternative we thought could work to be similar enough to our filtering functionality would be reliant on the upvoting/downvoting system. If it wasn't a relevant picture of an animal or squirrel, we could ask users to downvote to get rid of it from all users' feeds. And if it is a picture relevant or appreciated, naturally users would upvote. But even using this system, there is a possibility that upvoting and downvoting won't accurately reflect if the image being voted on is an appropriate image of the content. For example, an important external factor that could affect the reliability of this could be user bias: if the users develop hive mind, a couple of downvotes could result in a large amount of people doing the same thing even if the image is valid. Unfortunately, we were unable to fulfill our post filtering functionality, but this is definitely an important feature to update the app with in the future to ensure no spam, inappropriate, or irrelevant content is displayed on our app. With the possibilities of notifications and post filtering (to name a few future ventures), our app could become much more popular and gain thousands of downloads among the students at college campuses.