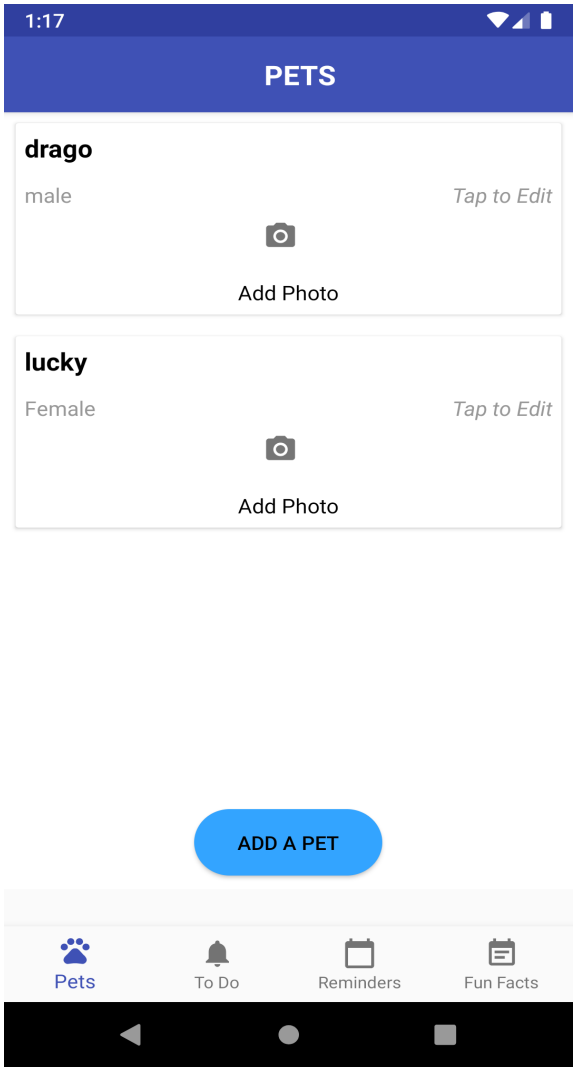
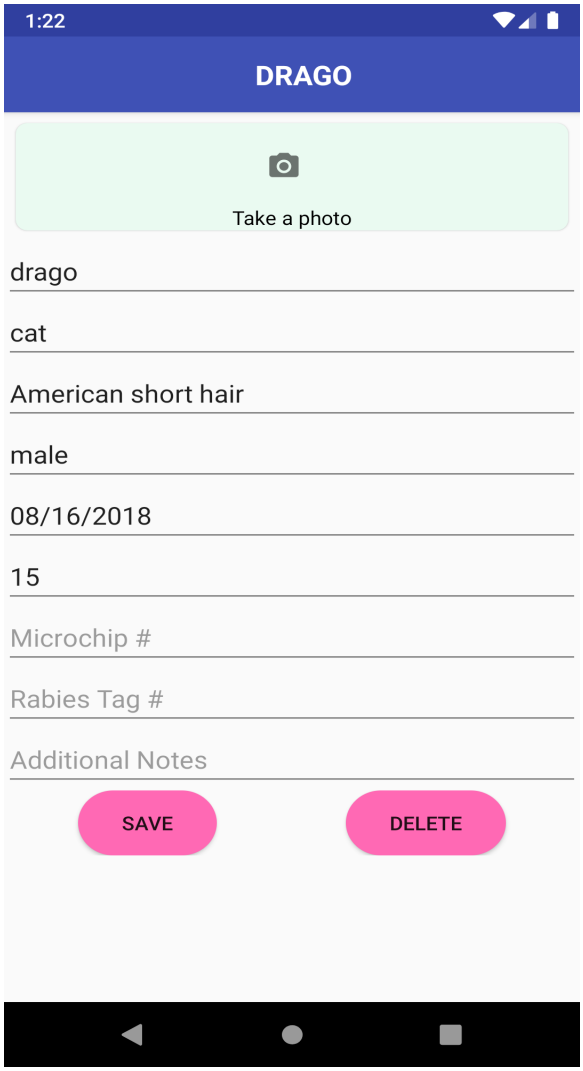
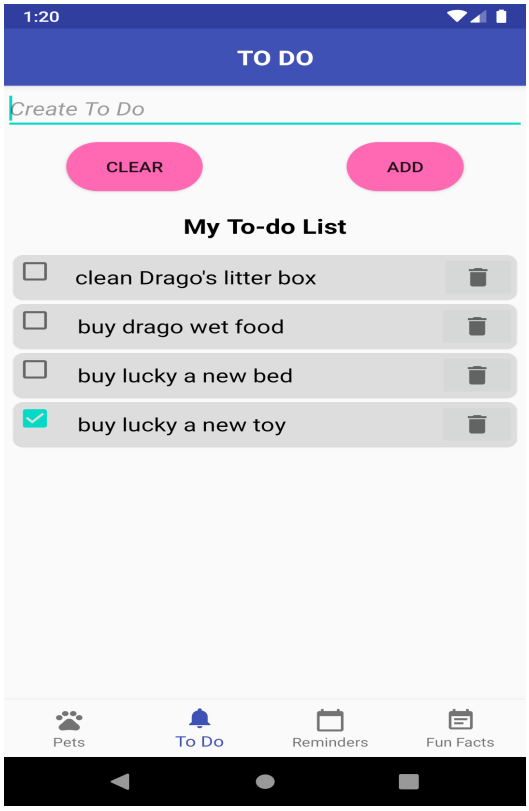
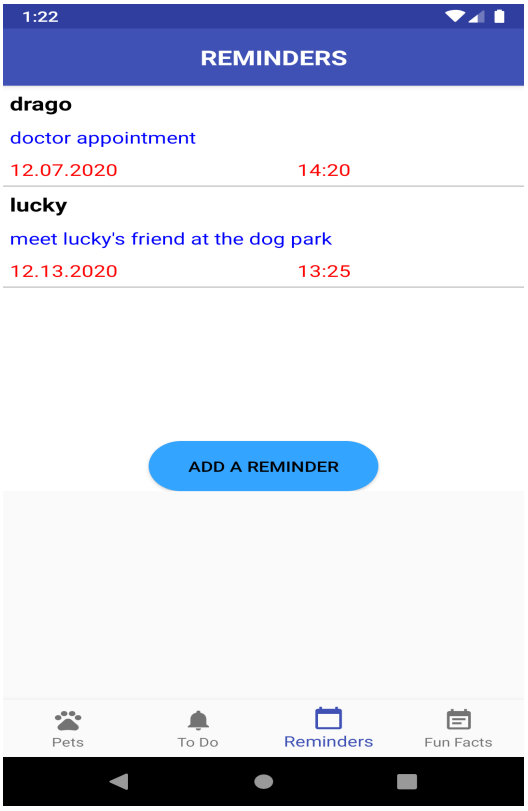
# Project Writeup

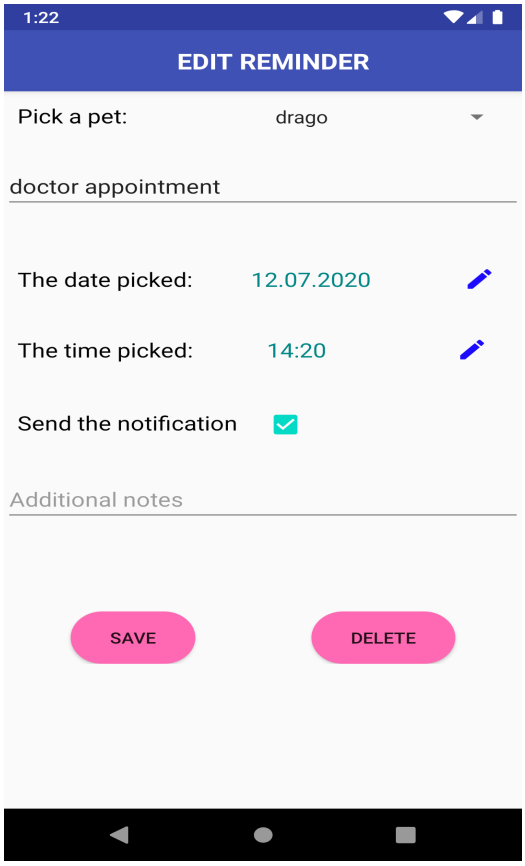
**App Title**: MyPet
**App Description:**
Android Mobile Application helps users take good care of their pets and learn some fun facts about our lovely friends.
**APIs used in the project:**
There are two APIs that I used in this app:
Cat Facts API: https://alexwohlbruck.github.io/cat-facts/docs/
Dog Facts API: https://kinduff.github.io/dog-api/
**Screenshots of the App:**

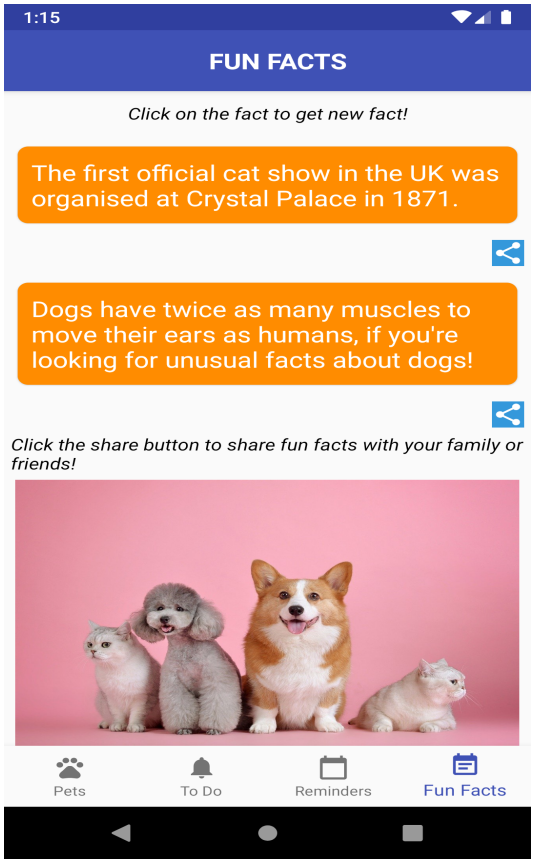



1:20
TO DO
Create To Do
CLEAR
ADD
My To-do List
clean Drago's litter box
buy drago wet food
buy lucky a new bed
buy lucky a new toy
Pets
To Do
Reminders
Fun Facts


1:22
REMINDERS
drago
doctor appointment
12.07.2020
14:20
lucky
meet lucky's friend at the dog park
12.13.2020
13:25
ADD A REMINDER
Pets
To Do
Reminders
Fun Facts


1:22
EDIT REMINDER
Pick a pet:
drago
doctor appointment
The date picked:
12.07.2020
The time picked:
14:20
Send the notification
Additional notes
SAVE
DELETE


1:15
FUN FACTS
Click on the fact to get new fact!
The first official cat show in the UK was organised at Crystal Palace in 1871.
Dogs have twice as many muscles to move their ears as humans, if you're looking for unusual facts about dogs!
Click the share button to share fun facts with your family or friends!
Pets
To Do
Reminders
Fun Facts

**Third Party Libraries:**

(1) Android KTX:

Fragment-ktx: used this library to control the fragment transactions with lambdas.
Lifecycle-livedata-ktx: I used this library to observe the pet profile data, to-do list data, reminder list data, action bar title, and pets fun facts. If those data changes, UI will display the changes.
Lifecycle-viewmodel-ktx: launches coroutine in viewModelScope to fetch the data from the API calls.
We have been using this library a lot this semester, and they are really helpful when it comes to observing the live data, and UI displays the changes immediately. I can't imagine how to accomplish those functions without them.

(2) Retrofit:

I used this library for making RestAPI calls for pets fun facts. I follow the pattern from CatNet, and It's easy to implement it.

(3) OKHttp:

Used it with Retrofit and makes http requests for the APIs.

(4) Glide:

Glide helps me fetch images from the Firebase Storage for this project. Professor already gives us multiple examples of how to use it, and I mostly follow the steps from the FC FireChat.

(5) Navigation KTX:

I used this library to create the bottom navigation view to navigate the four different fragments(pets, to do, reminders, and fun facts). It's a very convenient way to navigate to different views.

(6) Coroutine:

I used coroutine with live data to help me refetch the pets' fun facts from the network calls.

**Third Party Services:**

(1) Google-services:

Used Google-services in this project for the Firebase service.

(2) Firebase

I used Authentication for user login and log out. Besides just using the email login, I also enabled the Google and Facebook Logins for my app. The challenge for me is to enable a Facebook account login. I will explain it in the Facebook section.
I used Cloud FireStore to store the pet profiles, to-do list, and reminders for this app.
I also created a "petImage" folder to store the pet profile pictures in the Storage.
We already implement the firebase several times through the FC practices, so I don't have issues when using them, and they are easy and convenient to use for storing data.

(3) Facebook Login

I used this library with the FirebaseUI-auth for the user login. Setting this up is definitely more complicated than I thought it would be. First, I followed the steps shown in this link https://developers.facebook.com/docs/facebook-login/android/ to set up the things that I need. In the instruction, it tells me to provide the development and release of the key

hashes. I followed the instructions and tried to get it, but for some reason, I kept having some error messages after several tries. So I started searching for my problem on the internet and I found this youtube video very useful.
https://www.youtube.com/watch?v=Jt-Cv-s9AR0 . Instead of following the instructions given by Facebook, I typed the code provided by this video to get the key hashes. The other issue that I had is to make Facebook Login with FirebaseUI work together. After I set up all the codes for the Firebase Authentication, I kept getting the error message telling me something was wrong with my Facebook application id. I tried to make some changes in the String.xml and AndriodManifest.xml for the application id, but it's still not working. Then I found a solution in the stack overflow, and I tried to set the application id in the onCreat() function in the MainActivity.kt and it finally worked this time. I still do not know if this is the right approach to expose the application id in the MainActivity or not.

**UI:**

I used a lot of cardViews in my app, and I really like them. I also implemented the bottom navigation view for the UI. We have been seeing some demo codes provided by the professor that have those implements, but we never use them in the FCs and homework. So this project is a good practice for me to use CardViews and Navigation.

**Back End or Processing Logic:**

In this project, I used six fragments to display the major content, and used Firebase to store data and images. With FCs and homework, we only manipulated three different fragments at most and did not store much data in the Firebase. This time, the scale is larger, it's a good practice for me to make them work well together.

**Major Challenges:**

The major challenge for me in this project was figuring out how to implement the functionalities that we did not cover in the lectures, code demos, or homework.

1. My first major challenge was enabling the Facebook login function and I briefly talked about the problems that I encountered with this above.
2. The second challenge I encountered was with navigation. I know the professor implements navigation in a few Demo projects like RecycleViewFrag. But the problem that I had is how to hide the bottom navigation view when editing the pet profile and reminder. If the bottom navigation view is visible when the user is editing the pet profile/reminder, the user could click one of them and navigate to that view. It does not make that much sense. So I spent some time trying to figure out when to make the bottom navigation gone and visible. After I fixed this problem, another problem came up when I was testing the app. I found that if the photo did not take successfully(it happens when I clicked the back button or cancel photo) when I clicked the existing photo or added a new photo in the pet edit fragment, the bottom navigation view was visible again when finishing the take photo action. So I spent some time figuring out if the picture failed where I should hide the bottom navigation view. During this process, I understood the fragment lifecycle better.
3. Another challenge I had to work through was related to fragments. We studied fragments a lot in lecture, but this time, I encountered an issue where two fragments overlapped together. In the beginning, I used the replace method for the pet fragment then used the

add method to show the pet edit fragment. However, when I did this, two fragments overlapped with each other. I tried to change the background color and some other methods I found online did not solve this issue. Then I tried to use the add method for the pet fragment and the add method for the pet edit fragment. But they still overlapped together. The FireNote has similar functionality and the professor used the replace and add methods for interacting with two fragments. I don't know why it did not work on my project. So I changed to the replace and replace methods instead, and they worked.

4. The last challenge for me was to add notifications to the app. I spent some time finding some examples online about how to use the DatePickerDialog and TimePickerDialog, and how to format the date and time the user picked. Those are not difficult to implement. The hard part is how to use the AlarmManager to create an alarm to send a notification. The logic behind it is complicated too like, first, you need to create a notification channel, determine the date and time, create a pendingIntent, schedule it with the AlarmManager, register a BroadcastReceiver to listen for the alarm, and create a notification and issue it. I spent a few hours doing some learning online and saw how to put them together. After that, I was struggling with how to put the code together with the rest of my functions. It took me a few hours to trigger the first notification, but all in all, the learning experience is worth it.

**What I Learned:**

This project is a good opportunity for me to practice what we learned in the class and how to implement them in our project without following any instructions, code, or hints that the professor provided for us. Unlike the FCs and homework, where we can see other students' questions posted on Piazza and help each other, I had to solve my own issues. This project helped me improve my problem solving skills.

In this project, I used several functions that were not shown in the lecture and homework. It's a good chance for me to learn how to do some research online and try to implement them on my app. It's a good learning experience for me and I enjoy it a lot.

**How to Build and Run This Project:**

Unzip the project folder and open the project with Android studio(update Android studio if needed). The emulator that I used for this project is Pixel 3 API 28 Android 9.0. Then, click the run button to build the app.

For simplicity, this app will show the users all the data that has been stored in the firebase. It's very similar to the FireChat that we worked on. You could create your own email address, Gmail, or Facebook account to log in or you can use the information that I provided below. If you click the menu button on the action bar, you could find the sign out button to log out of the app.

Login information:

Email: fake@example.com

Password: 123456

Gmail(made only for this project): vivian3454592@gmail.com, password: 19920622Lyx+

**Lines of Code:**

I added a plugin in Android studio:https://plugins.jetbrains.com/plugin/4509-statistic

Kt files: 2349 lines, 1925 lines CODE

XML:  1526 lines, 1459 lines CODE

## Database Schema(Firebase):