Jacob Nelson

Dr. Haim Levkowitz

Mobile App Development I

11/29/2021

Mobile App Development I: Term Paper

Over the course of the semester, I have been developing what I have been calling “PollApp” for Android devices. It is based on the simple idea of allowing users to create and answer polls and surveys on a dedicated platform. There exist other applications that accomplish a similar task, but several of the ones I explored prior to developing PollApp were limited in various ways, such as only allowing polls to consist of just one question or displaying the poll results in a web page rather than within the app itself. PollApp is the answer to this problem, as PollApp will allow users to create robust, multi-question polls that they can create (or respond to) and see the results within the app.

This idea came about at the beginning of the course, where we were tasked with surveying people that we knew about apps they’d like to see developed for problems they might need to be solved. Nobody I asked said anything about a poll or survey app, but the process of surveying people itself gave me the idea afterwards when I got to thinking about how it would be much easier to send some sort of online form to the people I interviewed. Naturally, I looked for an app that might have eased this process, but there was nothing that really hit all the notes for what I was imagining. With this in mind, I started to conceive of the idea that would bring about PollApp. I really liked the idea because it seems like a very useful app as well as one that could be reasonably accomplished by myself, a junior computer science student with no prior Android development knowledge, within the course of a semester or two.

I decided to use Android Studio to develop PollApp, wherein I would use XML for the front end and Java for the back end. Java was one of the first languages I’ve ever coded in, and it is one I am quite comfortable with, which is why I chose it instead of another language like Kotlin. Installing and getting Android Studio and the Android emulators to work was the first challenge of developing this app. It took a lot of troubleshooting to get everything running. It also took a very long time to get it running on my own personal phone, because only recently I bought a phone with a working USB driver. It was, however, very satisfying and rewarding to finally see my app’s progress on my own phone. At this point in time, I’m very comfortable with (and thoroughly enjoy using) Android Studio as well as getting it to work with emulators and devices.

The biggest challenge that I have faced over the time spent working on my project is certainly trying to find a balance between when to teach myself topics I need to know, and when to apply what I’ve learned to my app. Alongside that was the challenge of actually understanding *what* I would need to learn, as Android development is obviously a very vast subject, and not all of its concepts would apply to my project. The time I spent working on this project was split between learning and actually developing the app. Going into this course, I knew it would be a lot of self-teaching, but I didn’t realize how hard that would be. At this point in time, I am quite proud of what I’ve learned and what I’ve accomplished thus far, especially since I have done it completely independently, and without a partner to work together on this project with.

At this point in time, PollApp is unfinished. It currently has functioning login and signup modules, and a basic home screen with placeholder Views for what would be the user’s currently active polls. The home screen has a navigation header which allows the user to navigate to the “Create Poll” and “Find Poll” fragments, as well as a logout button. The “Create/Find Poll” fragments are still in a primitive state, but they will be fleshed out soon.

The login and signup features of PollApp are very minimal, which is exactly what they need to be. Login and Signup are their own activities, and the login activity features the name of the app, a small “logo” type image, and EditText fields for e-mail and password. There is also a button to press to sign up if you are not a registered user, which, of course, brings you to the signup activity, which features three EditText fields: one for the user’s email, one for the desired password, and another to confirm the password. There is input validation that takes place here, where if the passwords do not match, a Toast message appears, and the user cannot sign up nor proceed onward into the app.

The user information (their email and password) is stored in a local database using the Room API. Getting this feature to work as well as learning it and taking the time to understand it has probably been the single thing I have spent the most time on thus far over the course of this project. It is an extremely involved feature, but a powerful one. The Room API for Android is an API that allows you to write Java code in order to perform SQLite operations. Using this API was an obvious choice to me because I know extremely little about structured query languages and databases. But with Room, one is able to create structured data for databases, data access objects (DAOs) and entities of a database, all completely as abstract Java classes. Of course, an ideal app would store login credentials and user information on a server or cloud, but at this time my experience with that particular subject is limited, so I made the choice to save the user information locally for simplicity’s sake. When this app has matured and developed in the future, this will change so that the database is stored on a server.

Once a user has logged onto PollApp at this current stage of its development, they are met with the “Home Activity,” where they first see the “My Polls” fragment of the app, which features a scrollable list of CardViews that represent the user’s polls that they have made. The user will be able to click on any one of their polls and see their results or their progress. This list will show the user both the polls that they created in the past that are now no longer running, and polls that they have made that are currently active. At this time, this list is filled with placeholder poll CardViews that are not linked to any actual poll or data, but this will change in the near future as I continue to develop the app.

Developing the “My Polls” fragment of the app taught me what I feel is a very powerful concept in application development: creating elements on the interface programmatically. This is because the list in the “My Polls” section is a RecyclerView, which hold custom CardViews. The custom CardView is a basic .xml layout fragment, which are associated with a ListView adapter, which allows list data to be presented in a RecyclerView. This allows for the RecyclerView to be populated by these custom views programmatically and dynamically. Learning how all of this works is a crucial part in the process of designing my app, as this feature of dynamically changing views and adding elements to an interface during the app’s runtime will be something I need to use again for parts of the app I still need to develop.

I think that the greatest challenge in developing PollApp is one that I am still working on, and that is figuring out a way to store poll data in a database. This is a challenge because the polls a user will be able to create with PollApp will be various in their number of questions and number of answers. With my limited knowledge of database languages like SQL, it is difficult for me to imagine how I might represent a poll (which could have any number of questions) as a database entity? The way I’m currently approaching this problem is by creating a data structure to represent Polls, Questions, and Answers. The Poll class contains an ArrayList of Questions, and Questions have their own ArrayList of Answers. This way, I can hold all of the information about a Poll in a Poll object, and then I can use the Room API (like before with Users) to create the database entity class PollEntity, as well as a database PollDatabase to hold the PollEntities, and a DAO PollDAO which will allow access to PollDatabase. In code, I can then create a Poll object and then store it in the database. I think that this object-oriented approach is a simple and clean way to tackle this problem without having to worry about any potentially really complicated boilerplate database design. Of course, since this would be a Room database, it would be stored locally on the device, but like the User database, the ideal or finished version of PollApp would store that data on a non-local server.

One of the next tasks I must complete for the app is creating the interface for poll creation. This will be a challenge, but one I have the tools and the knowledge to tackle. The idea behind poll creation in this app is to allow users to have robust polls with multiple questions with various numbers of answers. So when the users first starts creating a poll, they are given an EditText for entering the title of the poll at the top of the page. Below, they at first only have an EditText for one question, and below that, there is only one EditText for a single answer to that question. However, there will be a button (likely with a ‘+’ symbol on it) to add additional question EditTexts to the page, as well as another similar button to add additional EditTexts for answers to the questions. This way, a user can design a poll with as many questions as they’d like, with as many responses they would like for each question. Designing this interface can be accomplished with a RecyclerView with adapters for custom views, similar in concept to what takes place in the “My Polls” fragment, as discussed earlier. Once the user is finished filling out all of the EditTexts for the questions and answers, they can click a button on the bottom of the page that will create the poll for them. This will be accomplished by extracting all of the text from each of the EditTexts, and each question will be represented by its own Question class, and all of the responses will be represented as its own Answer class, which will be stored in the ArrayList of its corresponding Question object. All of the Question objects will then be put into the ArrayList of the all-containing Poll object. Once it is all set inside a new Poll object, that Poll object can be stored into the PollDatabase.

Following that, I will then develop the “Find Poll” feature of the app, which allows a user to type in the unique ID of a poll that someone has created. Poll.java has a method to generate a random string, which serves as the ID of the poll. If the user types in the ID for an existing poll in the database, they will then be allowed to answer that poll. For each answer they select, that answer gets a value incremented which keeps track of the number of different people who have voted for that same answer. Designing this part of the app will involve querying the database for the correct poll, retrieving that poll, and creating the layout for the poll, using TextViews for the questions and radio button lists for all of the answers to each question. It will involve RecyclerViews and adapters and custom Views just like previous parts of the app. Once the user is done answering the poll questions, they can press a button that will be at the bottom of the page that says “submit,” and the Poll Entity in the database will be updated so that the answers are incremented appropriately.

At this time, I plan on taking Mobile App Development II at some point in the future. If I do, I fully believe that by the end of that course I would have a fully-working app, unconstrained by my own limitations of time and knowledge that I currently have at this point in my education. If the current semester was only a few weeks longer, I’m almost certain I would have a working PollApp that functions completely, with only the constraint of using local databases to save data. I think, however, that before the end of the semester, I will be quite close to bringing PollApp to that level. It is hard to say exactly how much of that can get done in the remaining two weeks, but I will continue to work hard on it until the final day. A primitive, but still fully functioning PollApp is just over the horizon. I’m extremely proud of what I have learned and accomplished in the field of Android app development. I went from knowing virtually nothing on the subject to feeling quite capable with what I can actually program into an app. Even if circumstances change and I don’t end up taking Mobile App Development II in the future, I still feel it is imperative I keep working on PollApp. I really do think it has the potential to be a useful app, and I would love to continue its growth.