Android Application Development: Android Physical Therapy

Authors:

Leanna Myers, Felipe Soares, Quintin Warren

Abstract:

Our project had as its goal to create an android app that would be used in the medical field and that would integrate with the VistA medical system. Our assigned nursing student has a lot of experience in the orthopedics floor and suggested that one aspect of provider and patient relationship that could be improved is the communication and home therapy follow up. So the Android Physical Therapy app was developed with the intention of enhancing such information exchange, by providing ready access to information to the patient, and detailed follow up information to the provider, in the form of exercise logs uploaded by the patient to the VistA system database.

Background Information:

The VistA health information system was developed for the Department of Veteran Affairs to be used at their clinics and keep health records of patients. The system is developed under an open source license, so other hospitals and clinics can develop their own systems using the VistA infrastructure. It has at its core a database that is records driver and not relational, much like current NoSQL databases, even though it was developed several years before the NoSQL concept became mainstream. The record drive databases has the advantage of keeping records more private by emphasizing the speed of accessing all information relevant to a single record and increasing security by limiting the ability of searching across all the patients records for single bits of common information, as relational SQL databases are commonly used by businesses.

For our app we used a second record driven database (MongoDB) to store the information, to allow compatibility with the VistA database, while extending functionality not currently available in the VistA system.

On the Android app side, aside from the common development resources, to be described in the methods section, we used a couple of freely available internet access clients to communicate with the vista server.

Communication between the dedicated database and the general is expected to be done by background server side services either on demand or automatically.

The app was deigned to be customized and distributed by specific clinics, because it requires a back-end infrastructure to function, as such the demonstrated version included thematically relevant information, but ultimately used only as place holders for content provided by the clinic that eventually implements the system.

Methods:

For the User Interface part of the AndroidPhysicalTherapy app we coded in Android Studio using Java. The package name is edu.fau.group10.AndroidPysicalTherapy. There are 15 java class files

- FragmentActivity
- FragmentActivityFragment
- FragmentActivityFragment2
- FragmentActivityFragment3
- FragmentActivityFragment4
- FragmentActivityFragment5
- FragmentActivityFragment6
- FragmentActivityFragment7
- FragmentActivityFragment8
- FragmentActivityFragment9
- FragmentActivityFragment10
- FragmentActivityFragment11
- MainActivity
- MyDBHandler
- Users

The MainActivity .java file is the first file that is loaded on app startup and is also the largest file. It contains the code that controls the users login/signup as well as the code that controls the user interface, including exercises assigned and a video from Heartland Orthopedics. The MyDBHandler and Users .java files are temporary solutions for user login and signup. All the various FragmentActivityFragments contain the code for loading each unique fragment.

For the .xml files for our app we have 16 files.

- activity_fragment
- activity_login
- activity_main
- content_fragment
- content_main
- fragment_fragment
- fragment_fragment2
- fragment_fragment3
- fragment_fragment4
- fragment_fragment5
- fragment_fragment6
- fragment_fragment7
- fragment_fragment8
- fragment_fragment9
- fragment_fragment10
- fragment_fragment11

The content_main.xml file contains the layout for the main page of the app. It includes the Exercise names in the top left, the fragment container on the right, a timer and corresponding buttons on the bottom left and a webview at the very bottom. The timer was made using a modified Countdown timer and the webview was made by loading a javascript enabled iframe that makes a call to youtube. The activity_login.xml contains the EditTexts and Buttons necessary for login/signup. All the various fragment_fragments contain the name and description for the various exercises as provided by group 10's medical student Arielle Labiner.

There are 4 resource files for the app:

- logo1.png
- reset1.jpg
- start2.jpg
- stop1.jpg

The logo1.png contains the logo for the app as seen on the icon. The start2, reset1, and stop1 files contain buttons for the app's built in timer. All resource files were created by Vinicius Sandins.

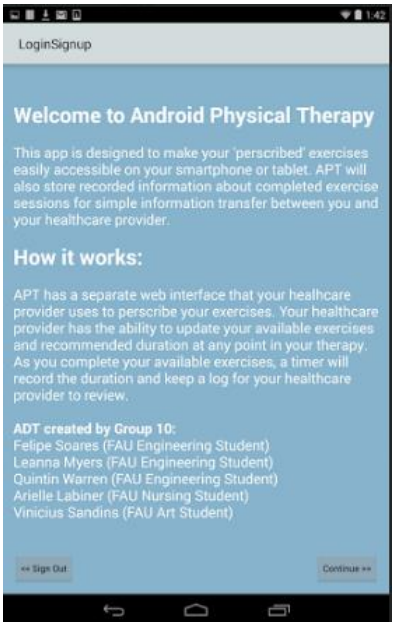
Results:

We were able to build a functional application using SQLite for storing recursive information. The SignUp function stores user information in the database. The username, password and VistA identification is stored for use later in the application. The login function checks the username and password with the values stored in the SQLite database so that the application is only available for registered users.

After registration, the user will be directed to the about section where more information about the application is provided. We have this about section prepared and uploaded to Github but unfortunately we did not have time to add it to the Demo version of the app that was demonstrated in our presentation.

Figure 1: Login / Registration Screen

Figure 2: About Activity



After clicking continue on the about screen, the user would then be taken directly to the main screen for the application. We kept the application simple because our primary target for our physical therapy application is older patients. The main screen shows a list of available exercises on the left hand side that the patient can navigate through. The exercises are hardcoded to contain every exercise available right now, but that can be adjusted so that the exercises available are set by the healthcare professional and connected through the patients VistA identification number. Each exercise has a description showing the proper way of completing the exercise and a video showing the patient visually how the exercise should be done. Each exercise has a timer that the patient will use to keep track of how long the exercise was done for. That information will be stored and ideally will be available for the healthcare provider to view on the provider's web interface, which will help with patient and healthcare provider communication. The main activity where the exercises are listed is a fragment activity where the patient can navigate through each exercise in any order.

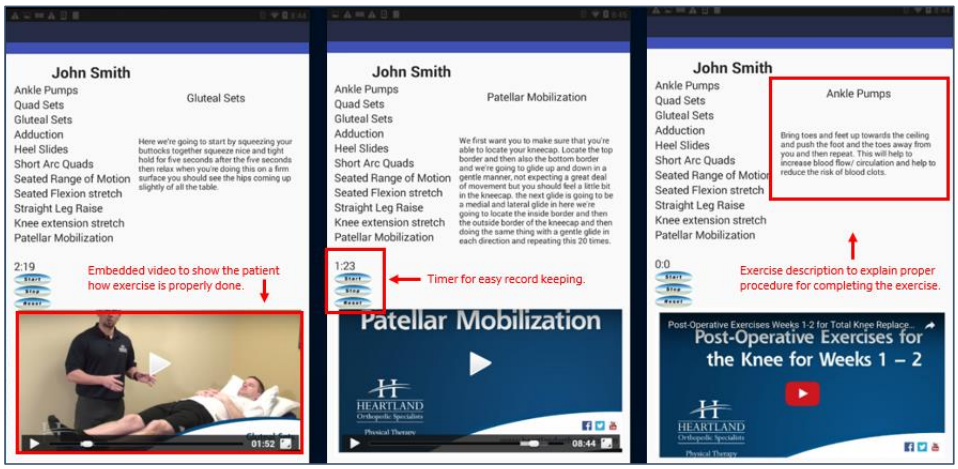
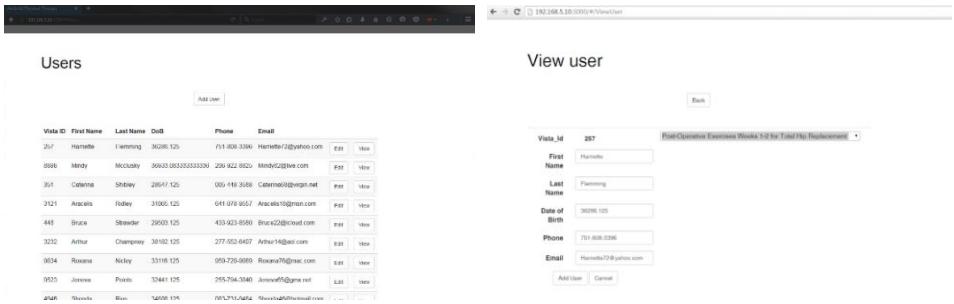


Figure 3: Main Activity showing separate fragments, including comments (in red)



Figures 4 and 5: Placeholder Healthcare Provider Interface

Discussion:

The initial purpose of integrating an Android app with the VistA health care provider system proved to have several challenges. While the software itself is open source and readily available, its front end is not, thus we could not browse a functional system to try and find the different parts where the information we would like to insert could fit.

Another challenge we had was the fact that the VistA system is not built to be easily extensible, so creating new tables or extending existing system tables on the system for now were out of the question, even API REST calls for insertion were limited to pre-determined fields that were not relevant to our app's purpose.

The solution we found to the problem was to keep using the same server that was already running the vista system and setup a parallel database using the same server stack using Node.js and a similar NoSQL database (MongoDB[i]) to store the data.

For the presentation, a website was quickly developed to simulate the data to be visible by the doctor inside the VistA system. The integration between both systems is still viable with this setup given some already existing software (EWD-REST[ii] npm module), but for now limited to the VistA data input limitations, which I believe should be resolved in the near future.

The server that was setup to support the app, was functioning and fully responsive to REST call, allowing queries and accepting insert calls. This brings us to the Android App integration, at this current time the Android SDK recently deprecated[iii] the main HTTP call client, and moved to a different method that does not have drop in capabilities with its replacement. After extensive research we found a library that was modern and simple to implement called Retrofit[iv], the implementation of such library proved more challenging than expected, and by the time the app was due to be presented, we still needed 4-5 more days of development and testing. A sample app using this library called REST-Tutorial[v] was modified and successful in querying the server for information and able to display a list of users.

As the group project app was presented, it could have the calls made to SQLite replaced with Retrofit calls to the server, since all the fragments used in the exercise list are dynamic. Some additional work would need to be done in order to upload the exercise log entries to the server, but the amount of work

Commented [f1]:

Commented [f2]: https://github.com/robtweed/ewdrest

Commented [f3]: https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html#behavior-apache-http-client

Commented [f4]: https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html#behavior-apache-http-client

Commented [f5]:

should be minimal to achieve that. The login page also has the correct logic implemented do to properly authenticate the user from the database. For a commercial implementation, steps would need to be taken to properly secure the server and communication between app and server.

Conclusion:

Throughout the semester our group was able to learn a lot about Android Application development. Most of our group had no prior experience in Java programming or developing Android applications so we had the ability to be exposed to a variety of new concepts that tested our problem solving abilities and our capacity for working in groups.

In summation, the improvements highlighted in the discussion section such as VistA integration and Retrofit calls to populate the fragment activities in the application could be achieved if time constraints were less strict and the project could be continued on to create a more integrated application that would make patient and healthcare provider communication simple for everyone involved.

References:

Android App Development for Beginners. (n.d.). Retrieved April 05, 2016, from https://thenewboston.com/videos.php?cat=278

Android Developers. (n.d.). Retrieved April 15, 2016, from http://developer.android.com/

Smyth, N. (n.d.). *Android Studio development essentials*.

Appendix:

https://github.com/HealthCareApps/Group-10-Physical-Activity

[i] https://www.mongodb.org/
[ii] https://github.com/robtweed/ewdrest
[iii] https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html#behavior-apache-http-client
[iv] http://square.github.io/retrofit/
[v] androidadvance.com/downloads/RESTTutorial.zip