Android Log-in Application

Ben Puhalski
Computer Science
Lakehead University
Thunder Bay, Ontario P7B 5E1
Email: bcpuhals@lakeheadu.ca

***Abstract*—This report will go through the features, shortcomings and bugs-fixes associated with this Android Login application.**

I. Introduction

Here I will go over the implementation and the journey that I went through in solving the problem described in section 2.

II. Problem

In this application, I would like to implement a solution to this given problem. The problem is to have some sort of method of saving data for multiple users in some application that multiple people may use. Each person should be able to see their saved data without other people being able to access that data. There should also be some way to "sign in" and "sign out" in order to let other users access their data.

III. Method

In this implementation I will create an app in android studio using the Kotlin programming language and create an app that will give a solution to the problem described in section 2.

A. Layouts

This application contains 2 layouts in each build. One for the main log-in screen and another for when the user logs in to see their data.

1) Login Screen: There are multiple activities in this implementation and therefore are multiple layouts. The first layout is the layout the user sees when they launch the application. The login in screen. This consists of 2 EditText fields and a 'login' button. These 3 Views are constrained inside of a linear layout. And this linear layout is again constrained inside a constraint view. The linear layout is also coloured a brighter shade of green to conform to the "material design" philosophy that many android developers conform to. The linear layout is constrained vertically to meet in the center of the user's screen. It also contains margins of 24dp horizontally so that the linear layout is not meeting with the edge of the screen and there is instead a consistent border around the linear layout. It also has a padding of 16dp so that the modules inside also have that border around them. The EditTexts are vertically layed out on top of each other as well as the button. Their widths match each other. Also, the EditText designed for passwords is a password field and will hide text inside. The button is a brighter shade of green and has white text so that the text is clearly visible to the user. There is also hint text inside of the Text Fields that say 'Enter Username' and 'Enter password' both in a darker shade of white so the user does not get the text confused with text that they wrote.

2) User Screen: The second activity holds a text view near the top of the screen, the users data in an EditText, as well as a save button designed to save the users data and return to the Login screen. The layout uses instead, 2 linear layouts to differentiate the username displayed at the top of the screen and the user's data. The username linear layout is constrained to be at 90% of the height of the screen and the user data layout is constrained vertically to the bottom of the username as well as 64dp from the bottom of the screen to allow room for the save button. This is so the user data box shrinks instead of the username as the screen shrinks or visa versa. The user data field also has hint text that says 'enter text data here' in the English version. There is also a button that is constrained to the bottom right of the screen that says 'save'. The button is colored similarly to the login button in our login screen activity.

B. Interaction

1) Class: User: I created a 'User' class that exists to send forward and backward user info through the activities as well as keep track of all of them simpler. The class contains 3 strings; a username, password and user Data string. The class contains gets and sets for the 3 strings as well as a parcel able attribute that allows this class to be sent through activities. This will be useful when we want to send our user through to our second activity.

2) Main_Activity.kt: Essentially, the login screen will just create a ArrayList of users that it will use to keep track of all of its users. When the user wants to create a user that does not exist in the application, the app will check through all of its users in the ArrayList and see if the inputted username matches any of the existing usernames in the program already. If one does not already exist, the program will create a new instance of the User class and send it to the second User Screen in which has default text in its user data member string. If a user already exists it will then check if the passwords match. If they don't, the app will display a toast message that says "incorrect password". If they match then the app will get that user and send it to the second user activity.

3) User_Instance.kt: in this activity, the app will get some user. the user's username will be displayed to the top of the screen. The user's data will appear in the edit text. If the user is a new user, a default constructor has constructed the User and will show the text ”hello new user” to show that the user has created a new user and they have not logged in to some other User activity. When the user hits the save button, the string from the EditText that holds the user’s data will be taken and used to change the User Data member. That User instance will then be sent back to the first activity, since the first activity was launched using StartActivityForResult().

4) Main_Activity_ForResult.kt: When the user sends their result back, they will get the user. The main activity will then have to check its arrayList to see what user it is. Then will change that user’s data to the changed string.

C. Builds

There are 2 builds; the full version and demo version. The full version consists of every attribute described in the above sections. The demo does not allow the app to display user data or support saving any data. But still supports creating users and logging in to them.

D. Translation

The app supports a french translation

IV. Bugs

Many bugs were encountered. Most were overcome when the professor told me about the startActivityForResult() function that allowed me to better execute what I was doing in the way I had intended. subsec

V. Results

The result is an example of a login activity application

A. Pros

This app can save and create users freely.

B. Cons

There is not implementation for writing to internal storage and will delete all the user once the app is exited.

VI. Future Implementation

In the future I could either

VII. Conclusion

In conclusion this project was difficult with many obstacles along the way.

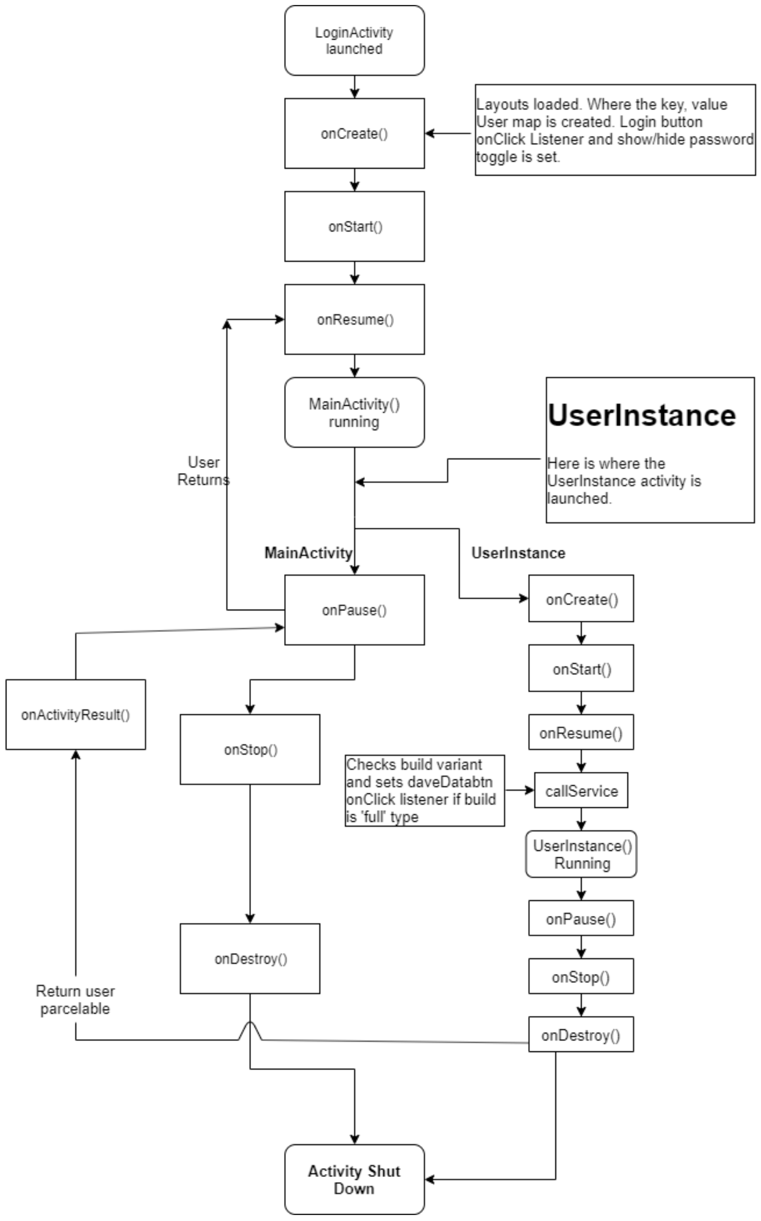

Fig. 1. Login Activity life cycle

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.