**Mobile App Development Total**

**Final Project**



Class: 2020级计算机班\_\_\_\_\_\_\_\_\_

Id: 20812502013\_\_\_\_\_\_\_\_\_\_\_

Name: Naimur Rahman \_\_\_\_\_\_\_\_

**Your Content:**

**PROJECT DETAILS**

A Banking app made for Android using Android Studio. No real money is involved, it is a project to showcase my knowledge and practical skill in Android development with Java. The Application was developed using a MVC approach, using proper programming conventions, including documentation, error/exception handling, thorough program structure and memory efficiency.

The app starts out with a login screen, in which the user can either log in with an existing profile, or click a button and create a new profile. When signed in, the user will be brought to their dashboard page, which (when first creating a profile), will prompt them to make their first account. Additionally, there is a menu that slides from the left which includes all of the options for the app, including Dashboard, Account Overview (and subsequently Transactions), Deposits, Payments, Transfers, Profile Settings and Logout.

**ANDROID DEVELOPMENT CONCEPTS USED**

**- Multiple Activities:** There are two activities: one which has the fragments for logging in and creating a profile, and the other for hosting all of the features the bank app has, including account overview, payments, transactions, etc. The activites serve as containers for the different fragments throughout the application. Intents are used to pass data from one activity to another. The activities themselves do not display a view, but rather host the navigation code (among other things) to travel between fragments.

**- Multiple Fragments:** What the user actually sees comes from the fragments of the application. These fragments are almost always launched from the activity that wraps them. Bundles are used to pass data from one fragment to another.

**- Well-designed UI Layouts:** Multiple layout files are used, using a well-thought design that keeps the simplicity of the app, while serving optimal functionality. Most layout files are used for the fragments, while some are used for menus in the application, as well as custom layouts for dialogs.

**- Custom Toolbar:** With the application using AppCompat, custom toolbars are a possibility. The toolbar is consistent throughout the app, with the XML code in a styles.xml file for re-use. The toolbar has a title that changes depending on the current fragment in use, and contains options for the user (including an options menu, back navigation, drawer menu).

**- Drawer Layout:** The application has a DrawerLayout, which is essentially a sliding drawer that typically comes from the left slide of the screen (either by swiping near the left edge or by clicking the hamburger button in the top left of the screen). This menu hosts the different features of the application, with each option either navigation to a fragment (corresponding with the feature), or launching a dialog in some cases. The DrawerLayout is in the second Activity, which serves as the master container for most of the application's fragments.

**- SQLite Database:** All Profile, Account, Payee and Transaction information is stored in a database. The DB consists of four tables, each with a proper primary key (composite or standard) and foreign keys when necessary. The database is stored on the user's device.

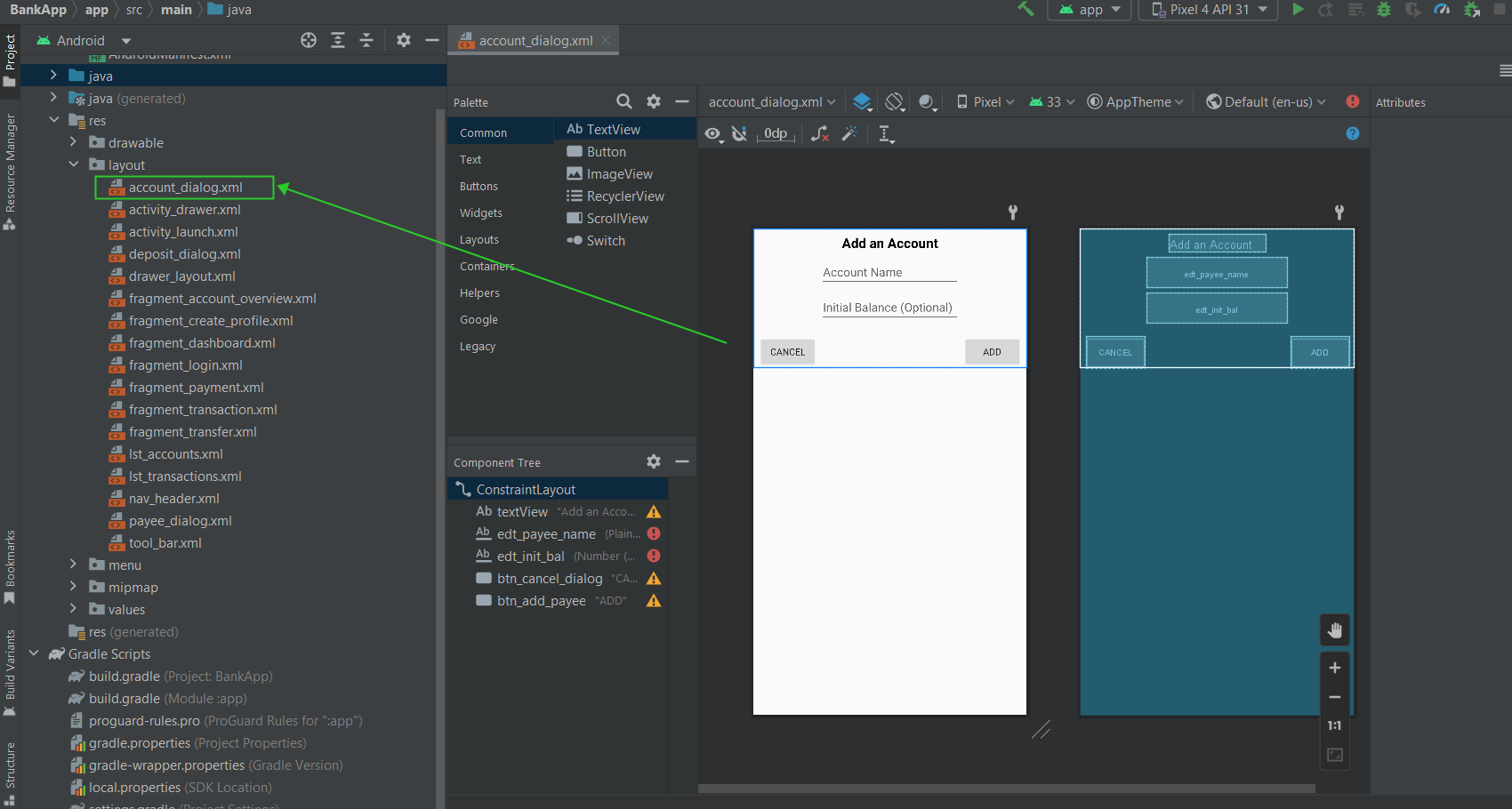
**- Shared Preferences:** Saving the current profile (logged into by the user), and all of its general info, accounts and transactions. When initially logging into a profile, all of the data from that profile is loaded from the database (stores all profile data). This operation is performed once, in which the profile data is stored into Shared Preferences and can be updated and loaded efficiently across the different activities. JSON is involved in the reading and writing of data into Shared Preferences.

**- Array Adapters:** Custom array adapters are used to display information in ListView's and Spinner's. The adapters used are for accounts and all transaction types (deposits, transfers and payments).

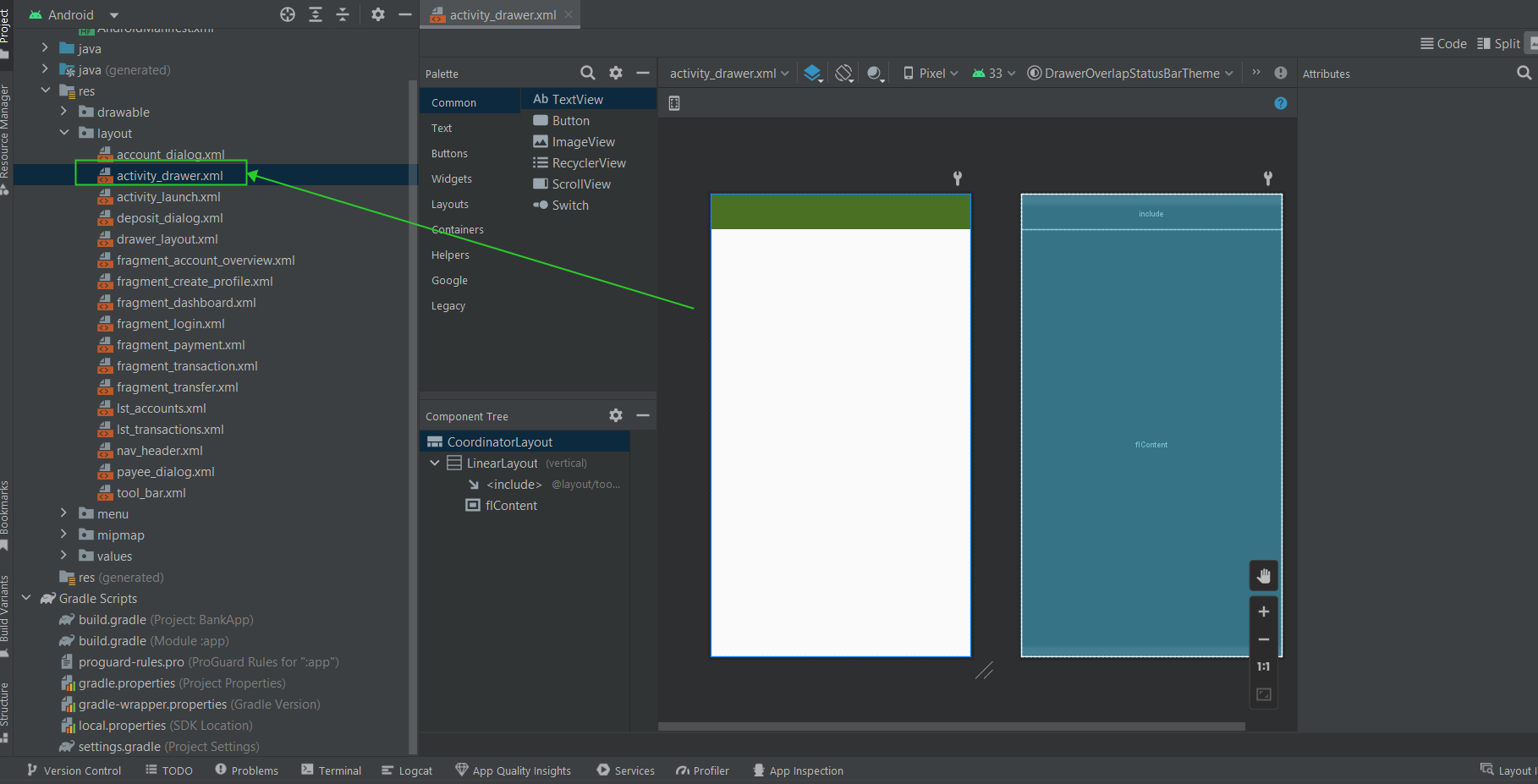
**Functional design logic**

1. **User authentication:** The app should have a secure login system that requires users to enter their username and password or use biometric authentication.
2. **Account management:** Users should be able to view their account balances, transaction history, and manage their account settings.
3. **Transfer funds:** Users should be able to transfer funds between their own accounts or to other bank accounts.
4. **Bill payments:** Users should be able to pay bills directly from the app.
5. **ATM locator:** The app should have a feature that allows users to locate nearby ATMs.
6. **Customer support:** The app should provide users with access to customer support through chat or phone.
7. **Security features:** The app should have security features such as encryption and two-factor authentication to protect user data and transactions.
8. **Notifications:** The app should provide users with notifications for important events such as account activity or payment due dates.
9. **Personalization:** The app should allow users to customize their experience by setting preferences such as language, notification settings, and account alerts.
10. **Integration with other financial apps:** The app may integrate with other financial apps such as budgeting tools or investment platforms for a more comprehensive financial management experience.

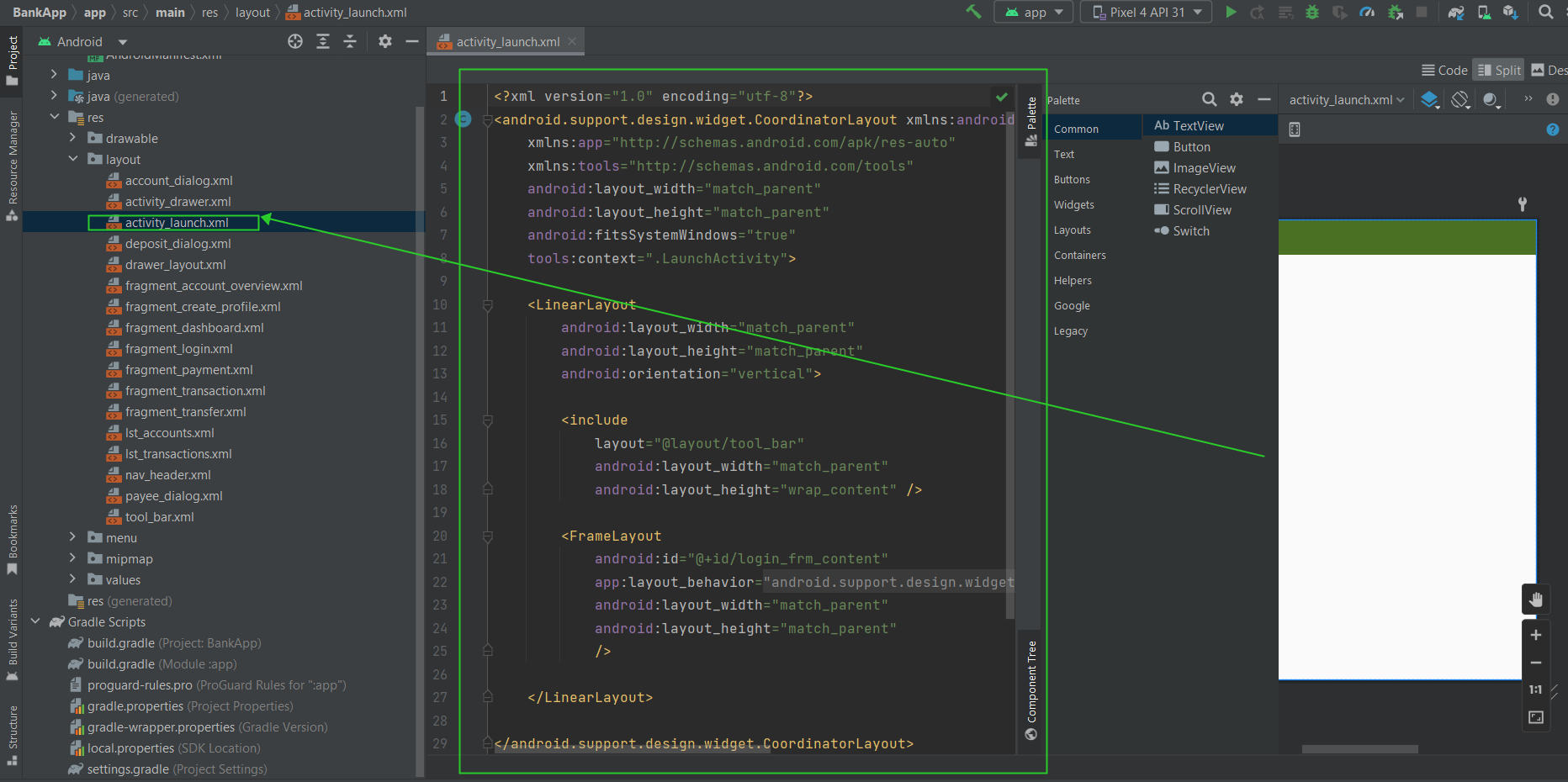
**Code technology implementation**



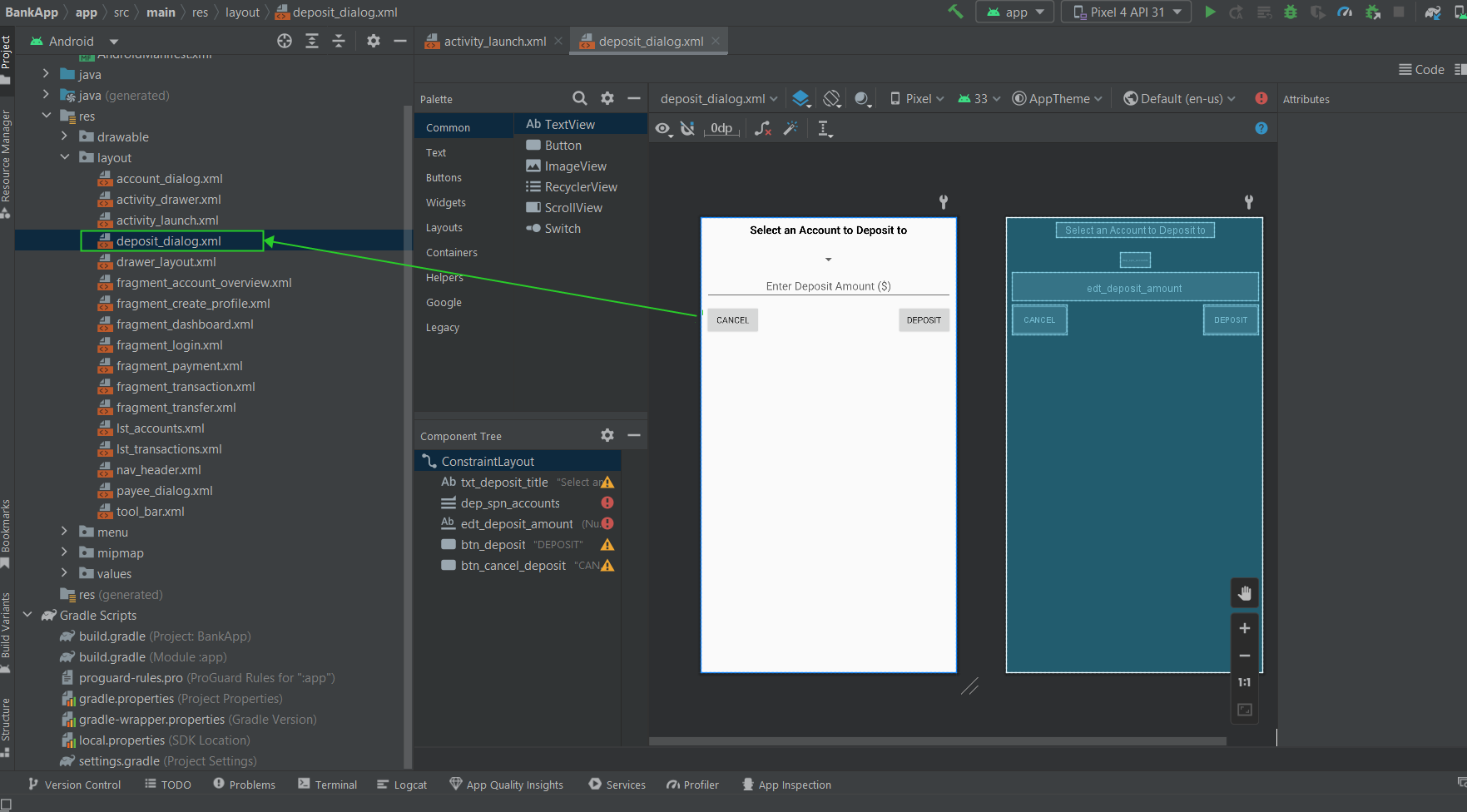
**Figure-01:** account dialog



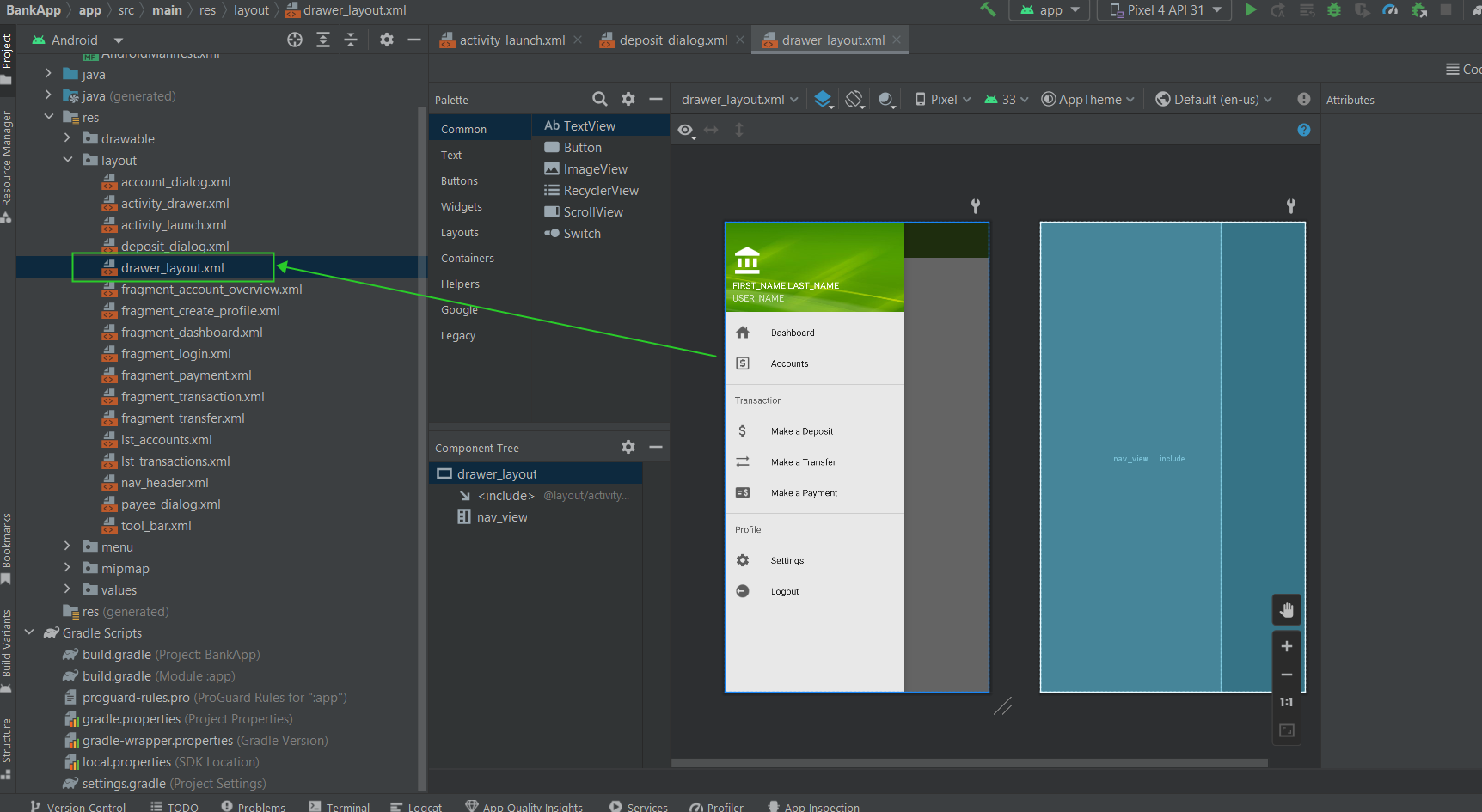
**Figure-02:** activity drawer



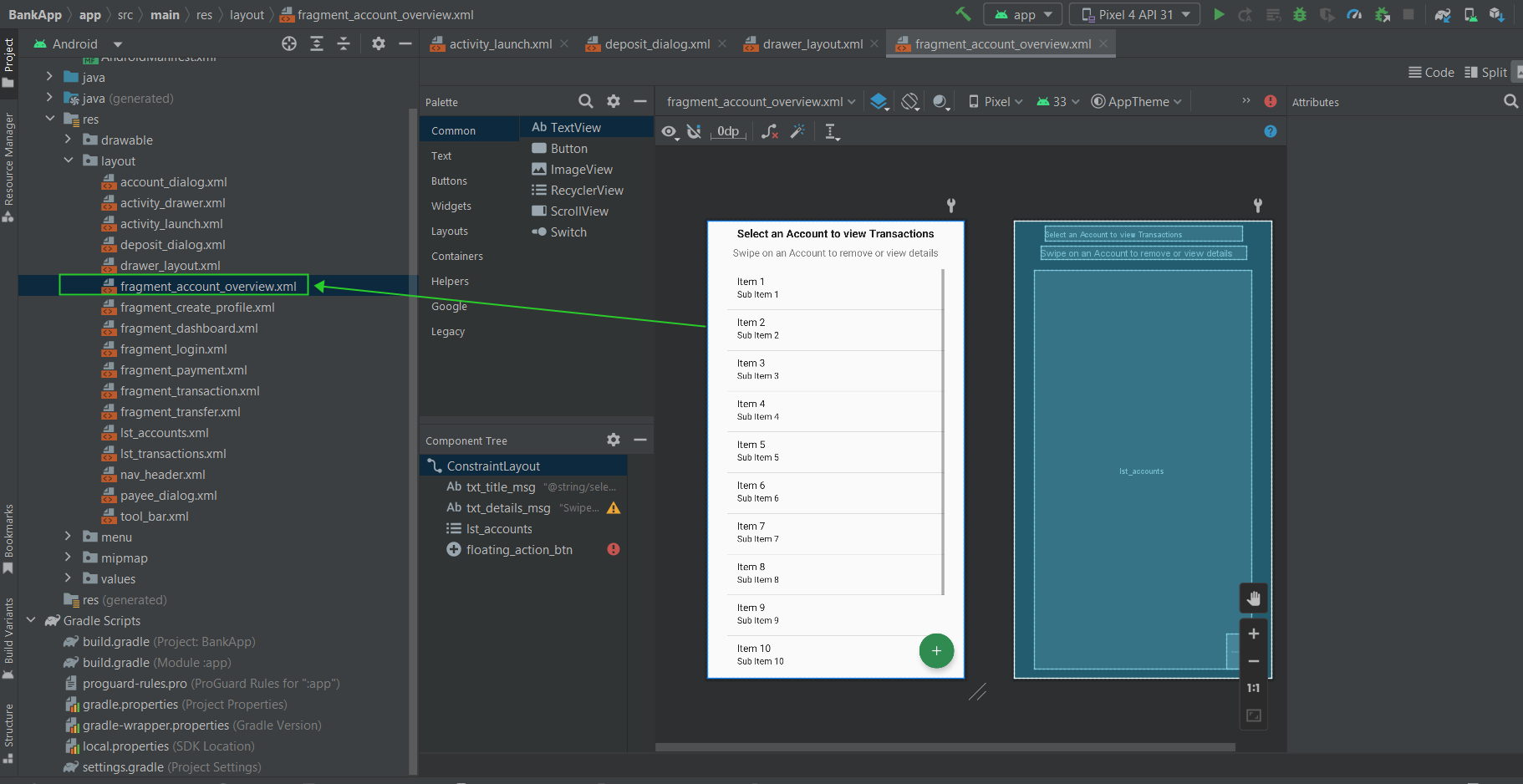
**Figure-03:** activity launch



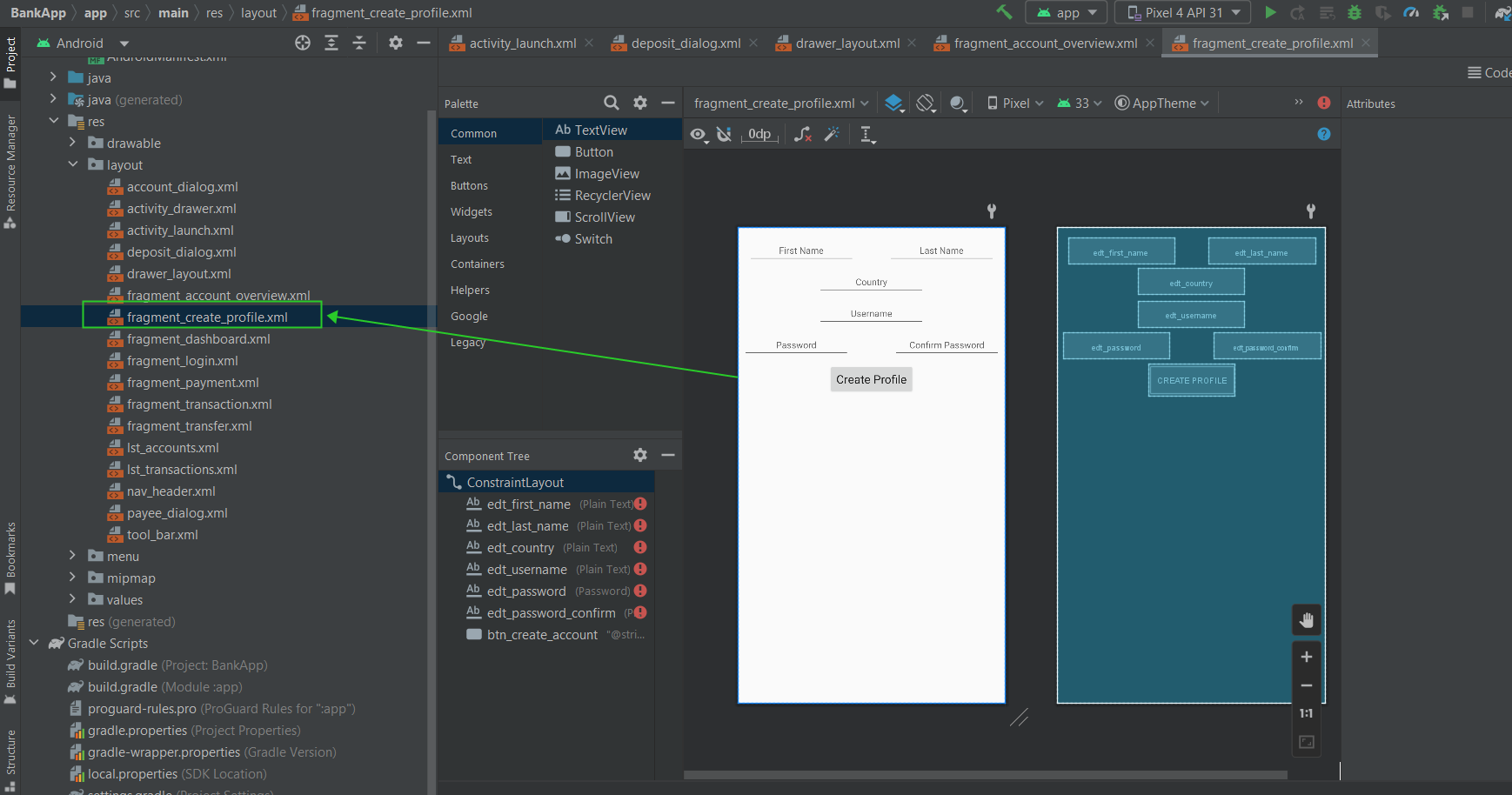
**Figure-04:** deposit dialog



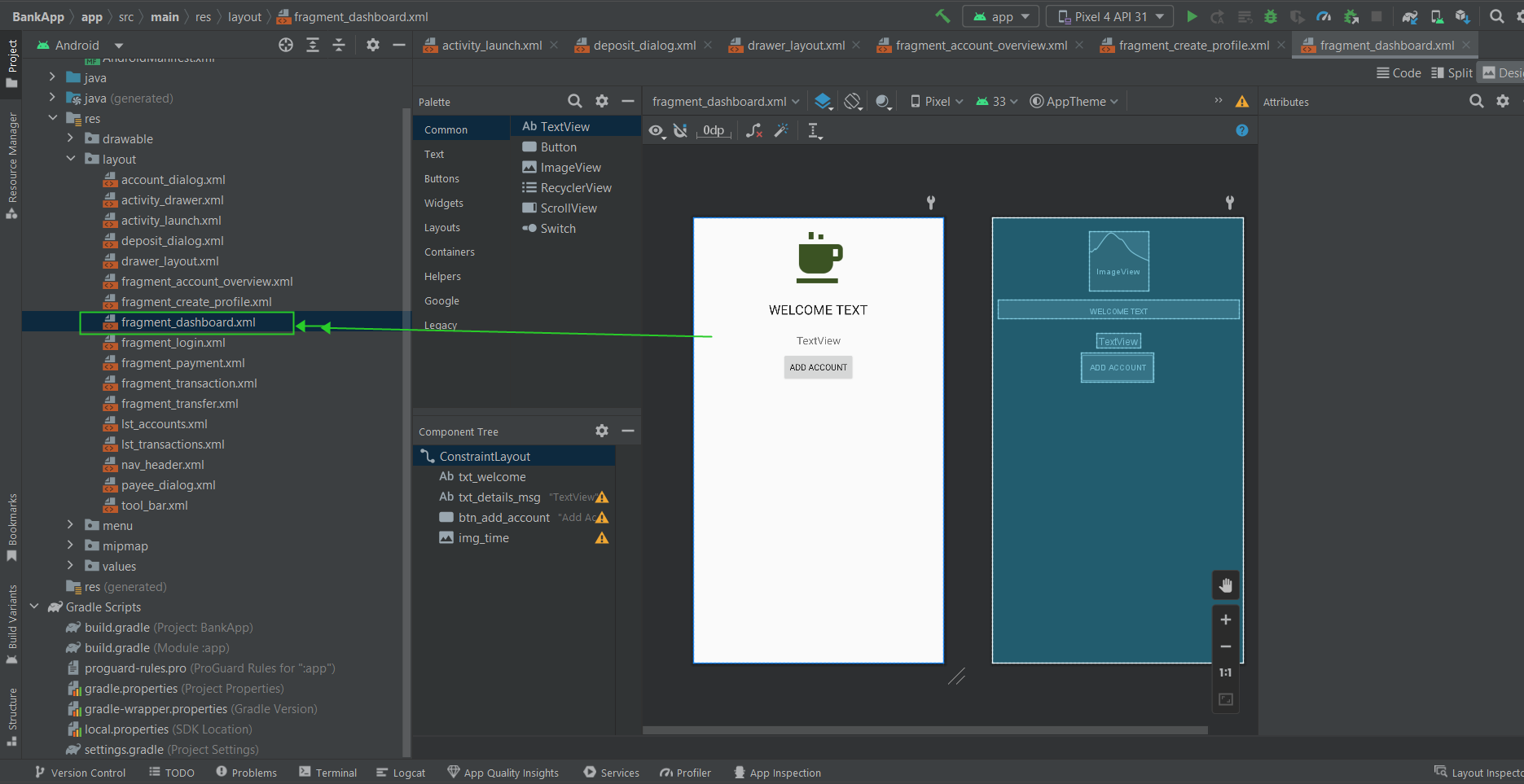
**Figure-05:** drawer layout



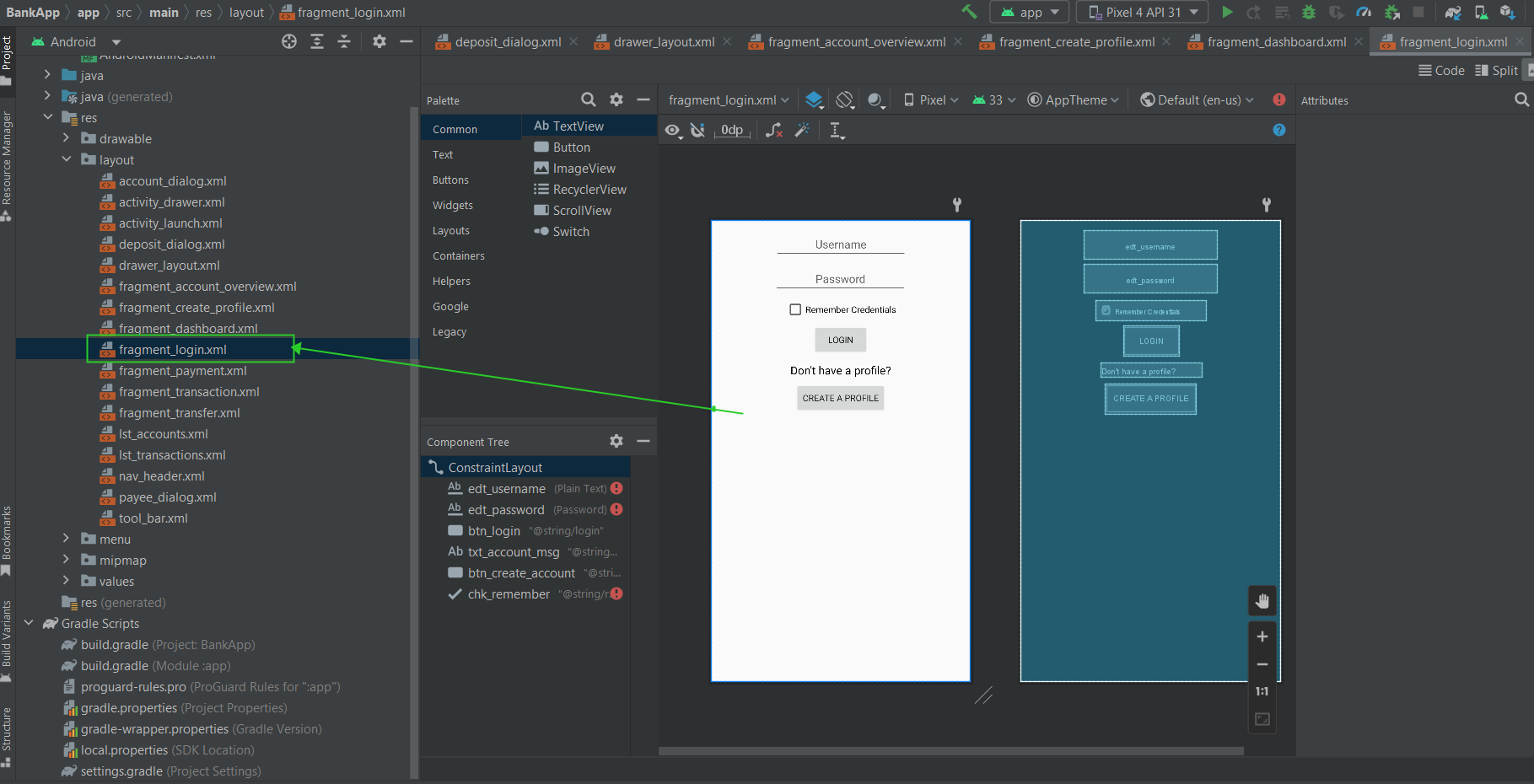
**Figure-06:** fragment\_account\_overview



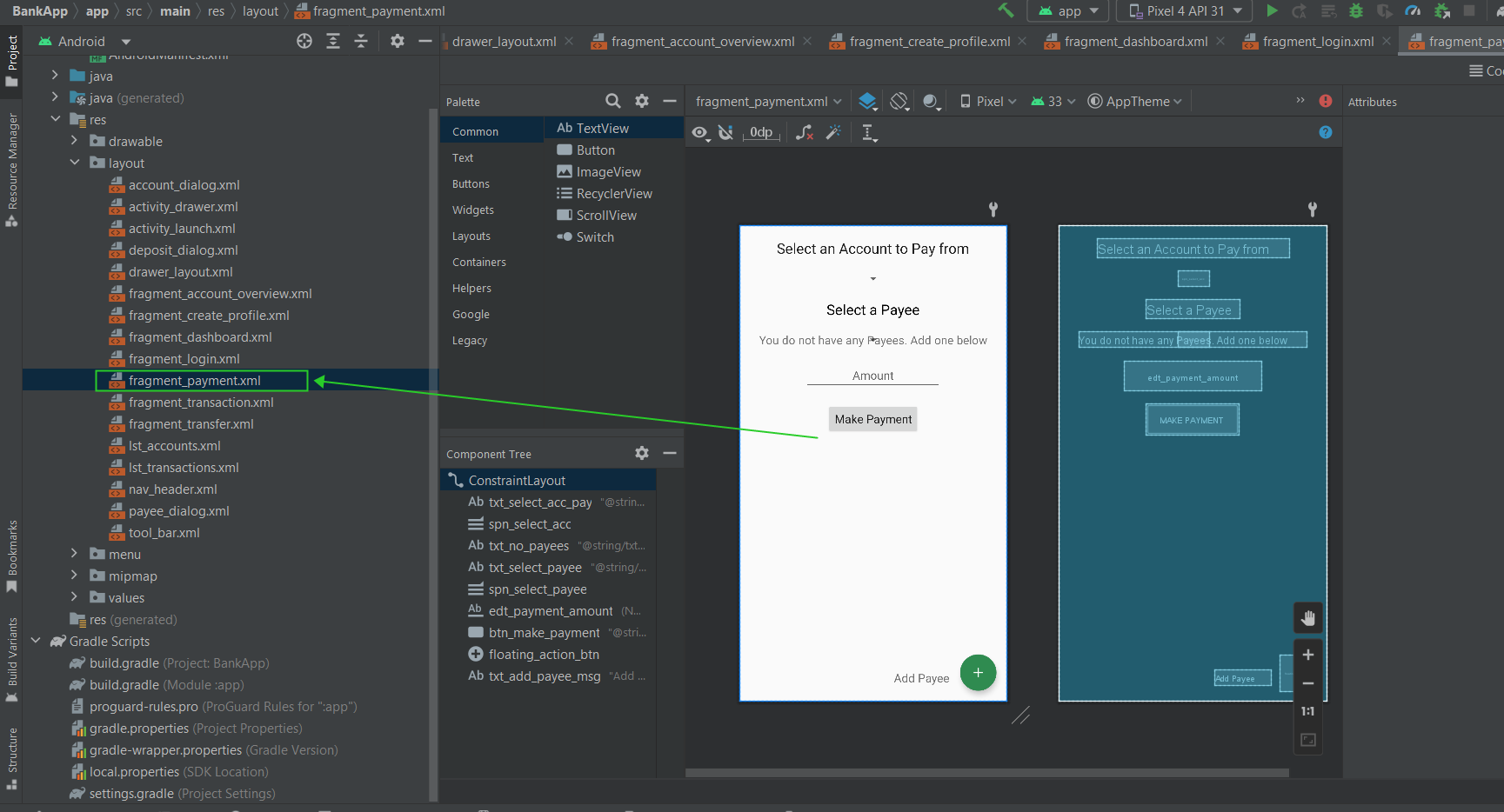
**Figure-07:** fragment\_create\_profile



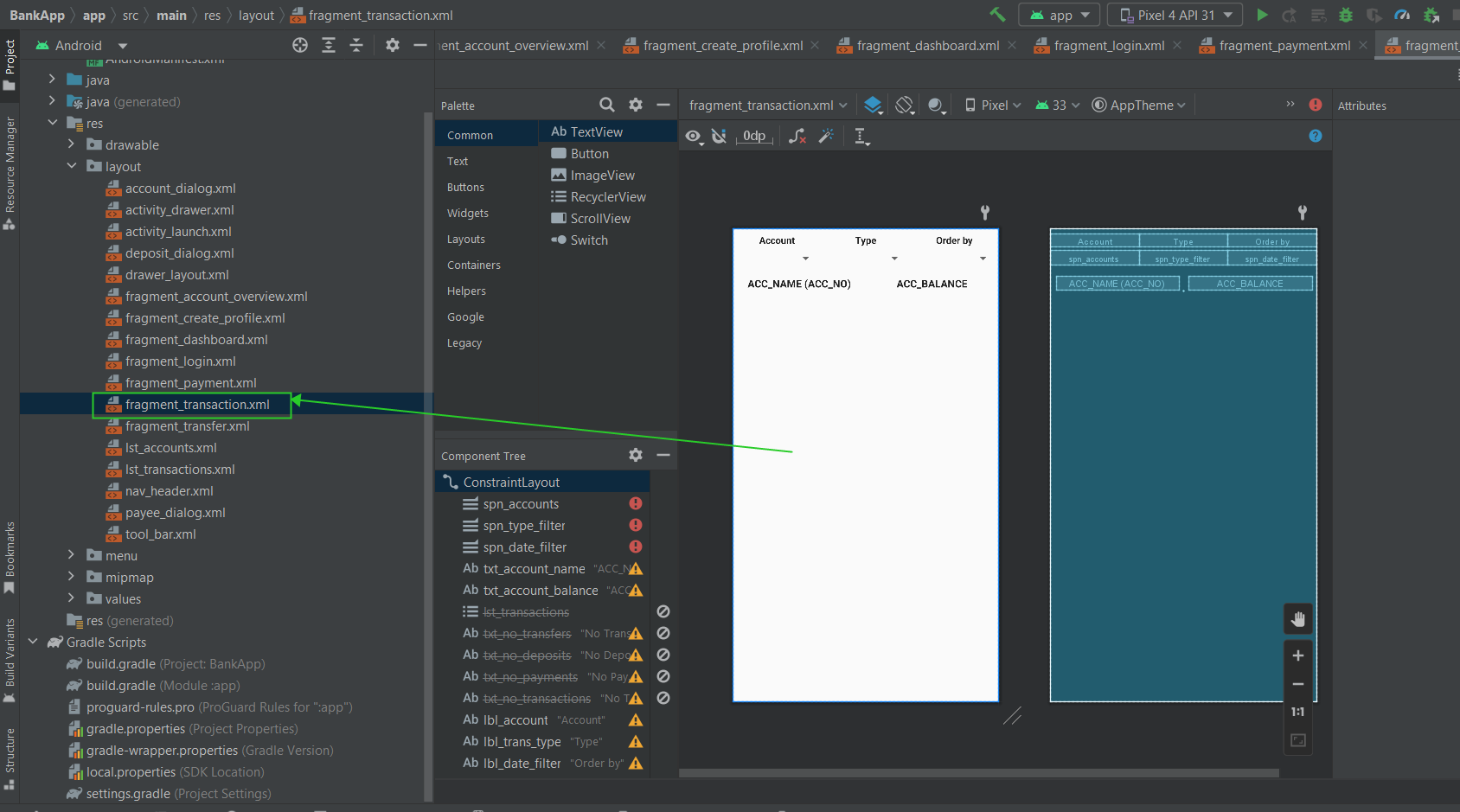
**Figure-08:** fragment\_dashboard



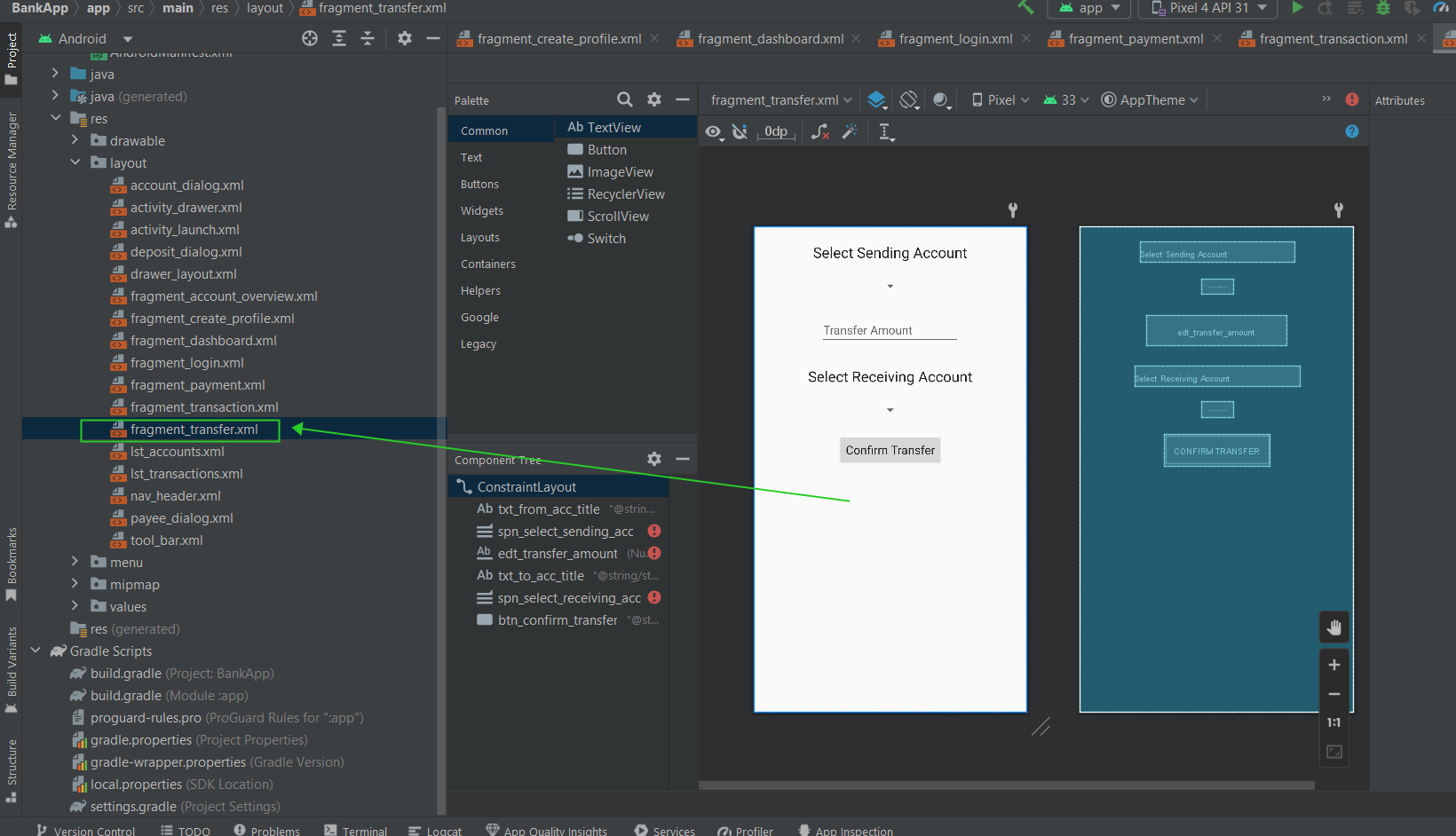
**Figure-09:** fragment\_login



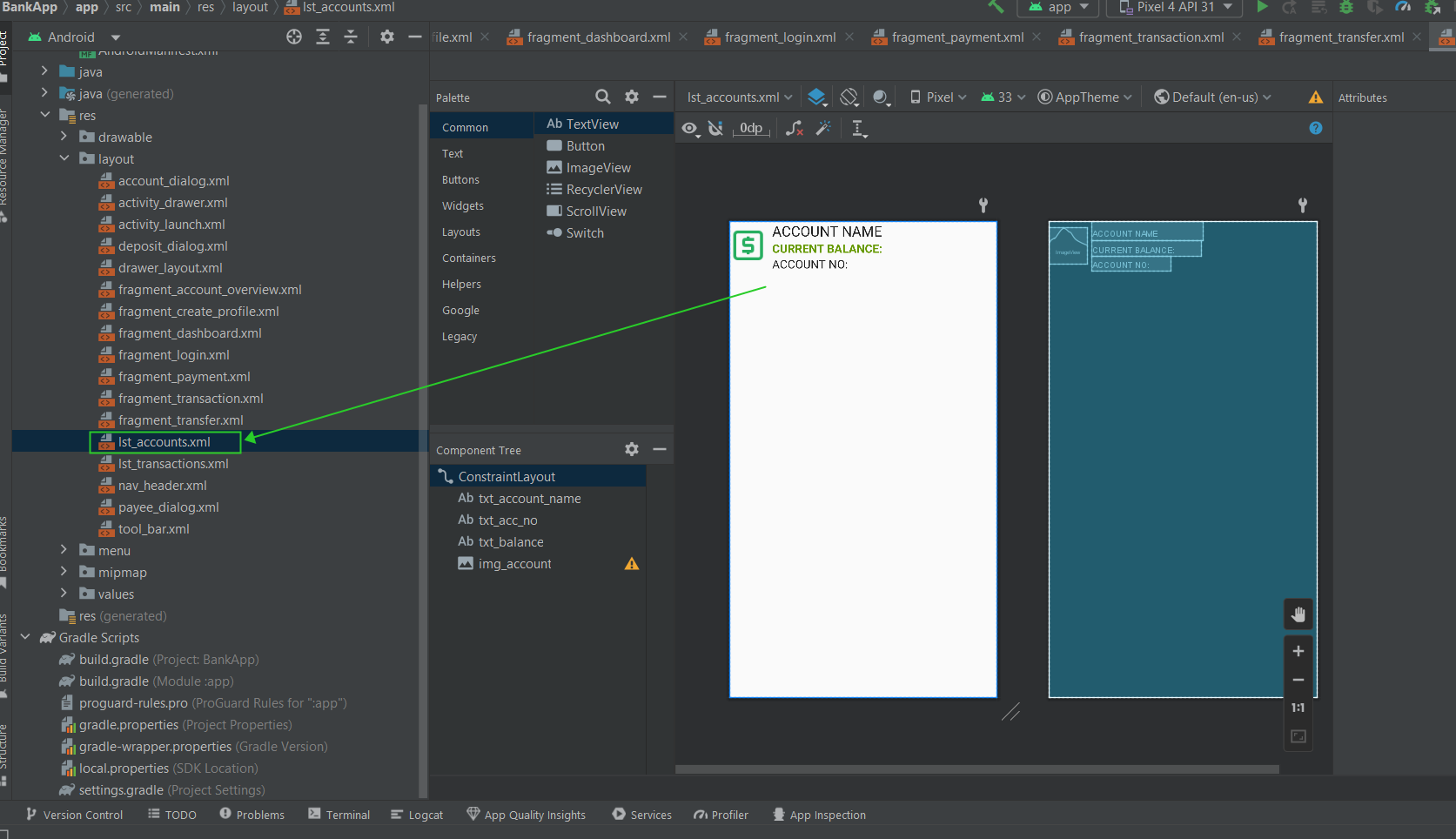
**Figure-10:** fragment\_payment



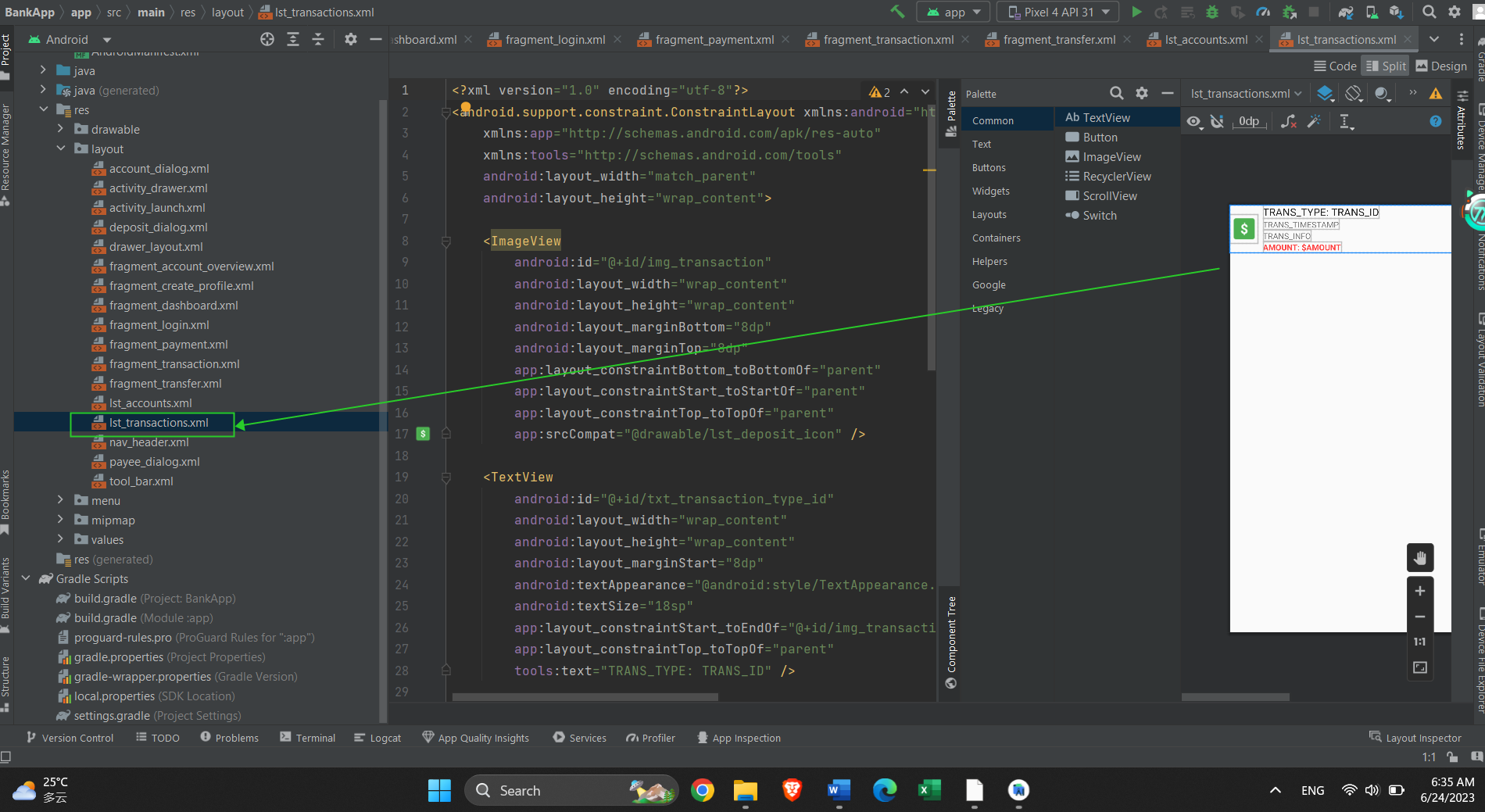
**Figure-11:** fragment\_transaction



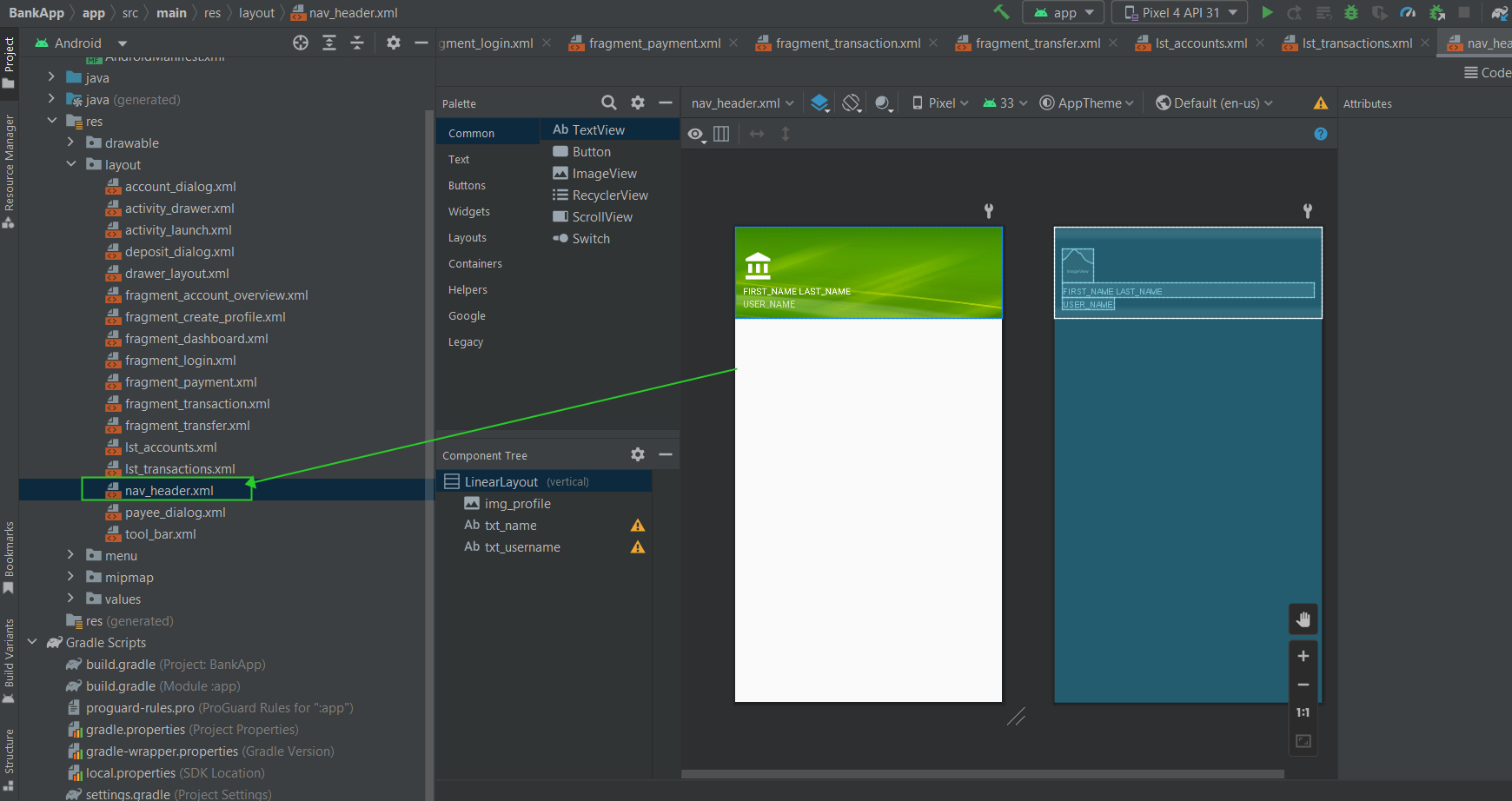
**Figure-12:** fragment\_transfer



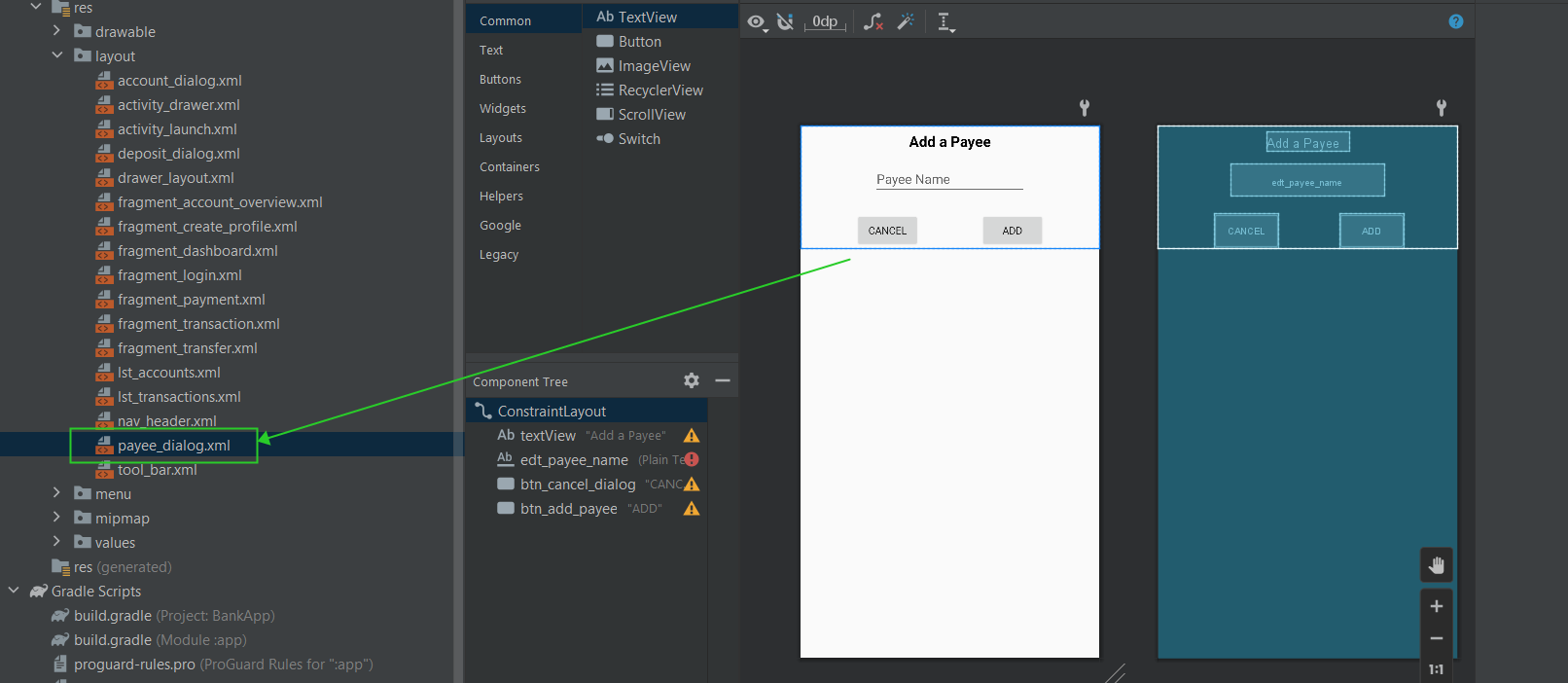
**Figure-13:** lst\_accounts



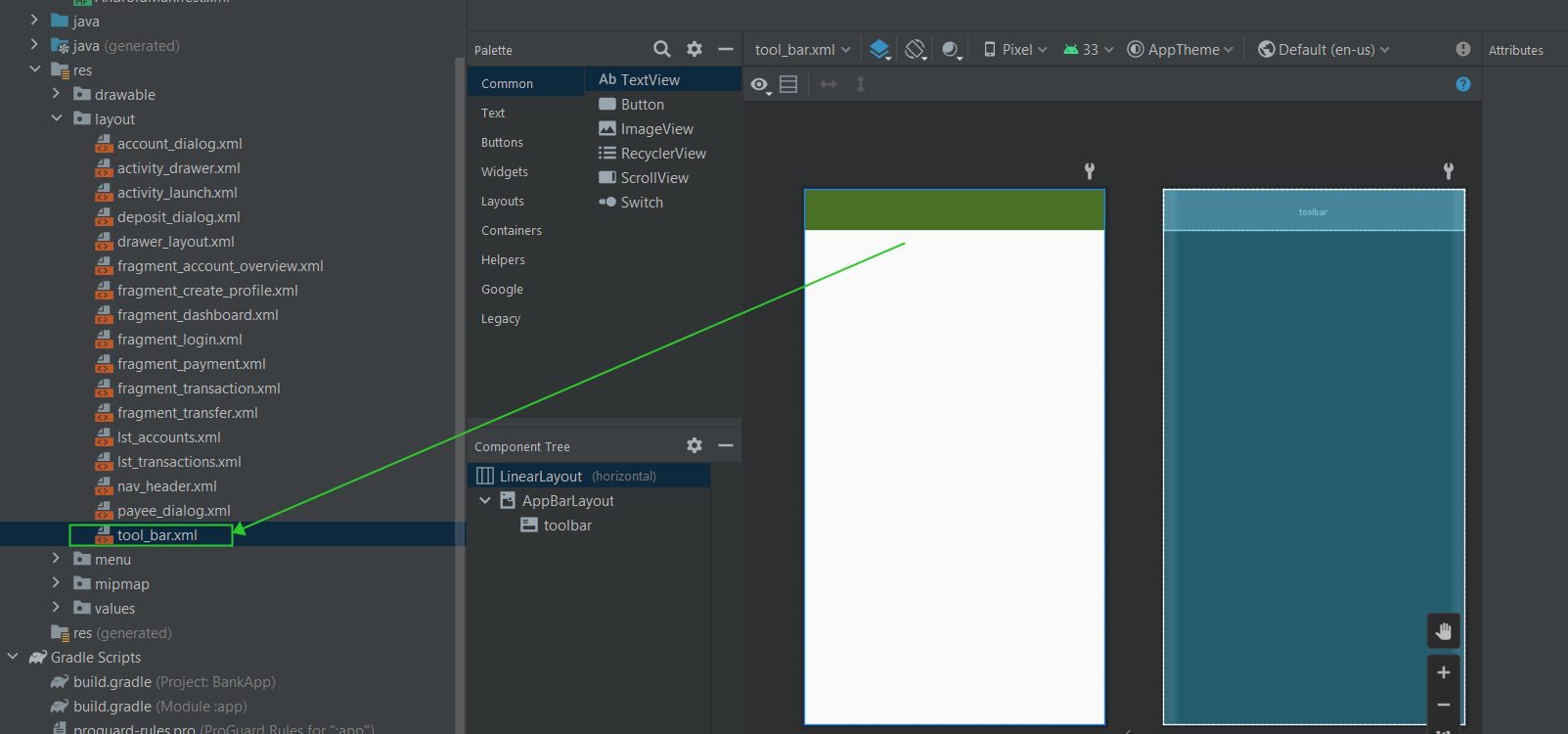
**Figure-14:** lst\_transactions



**Figure-15:** nav\_header



**Figure-16:** payee\_dialog



**Figure-17:** tool\_bar

**Course learning summary**

A Mobile App Development course typically teaches students how to design and develop mobile applications for various platforms such as iOS and Android.

In this course we cover topics such as user interface design, programming languages such as Java, database management, and app deployment.

By doing this course we learn how to use software development tools such as Android Studio to create functional mobile applications. They will also learn about the different types of mobile apps such as native apps, hybrid apps, and web apps.

The course also covers topics related to app monetization strategies, marketing techniques, and user engagement. We gain hands-on experience by developing their own mobile applications throughout the course.

Overall, a Mobile App Development course provides students with the skills and knowledge necessary to create innovative and functional mobile applications that meet the needs of users in today's digital age.