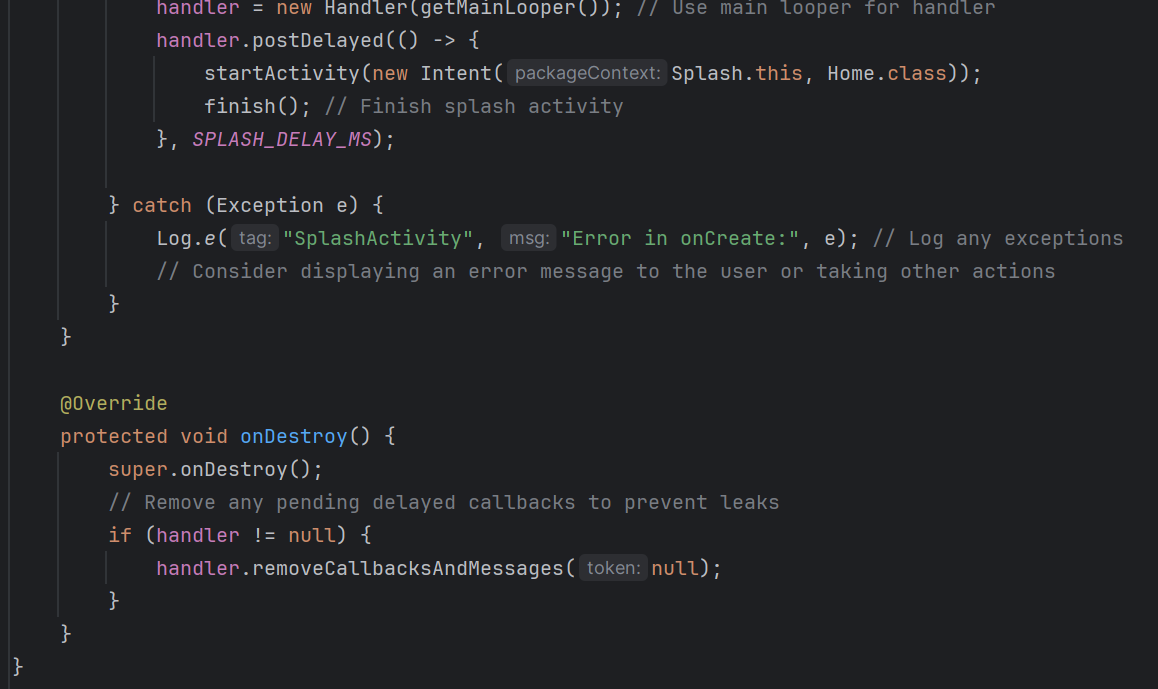
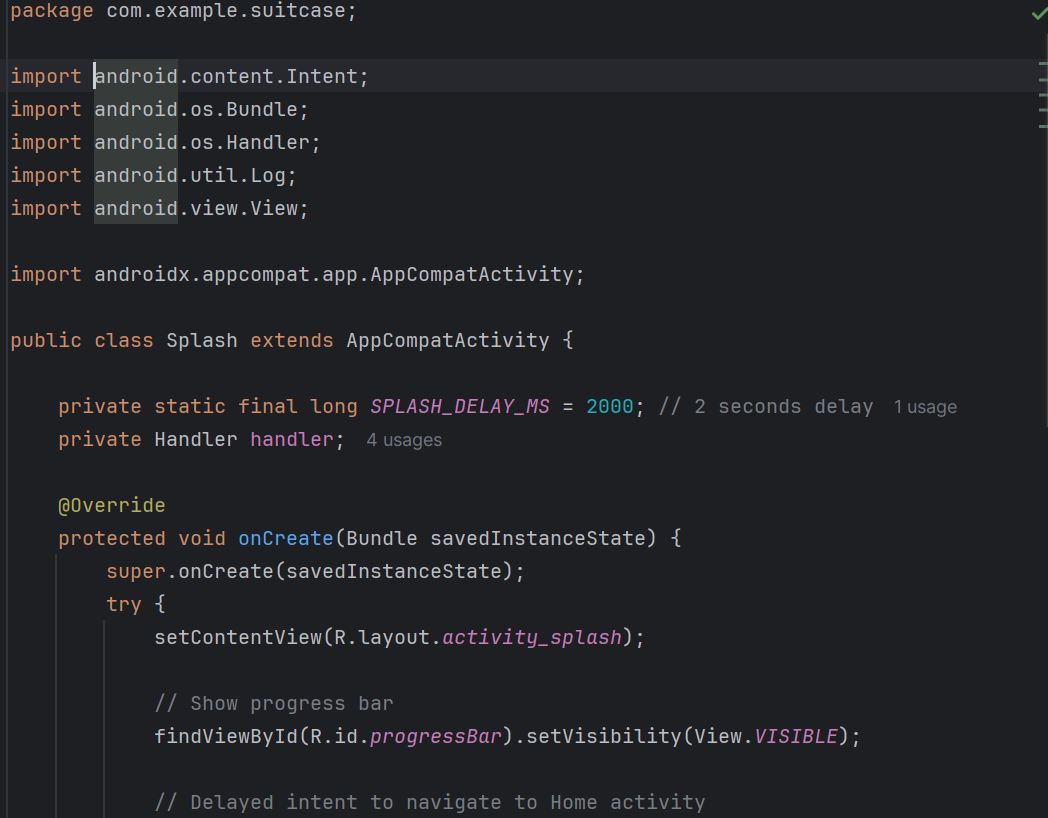
# Readme

I have used java as the main programming language to develop my app ‘Suitcase’. Firebase is used by the Suitcase app, which was developed in Java in Android Studio, for authentication, cloud storage, and database management. This paper breaks down the app's primary features and provides a thorough description of the implementation, storage requirements, and external APIs used, in addition to rationales for design and implementation choices.

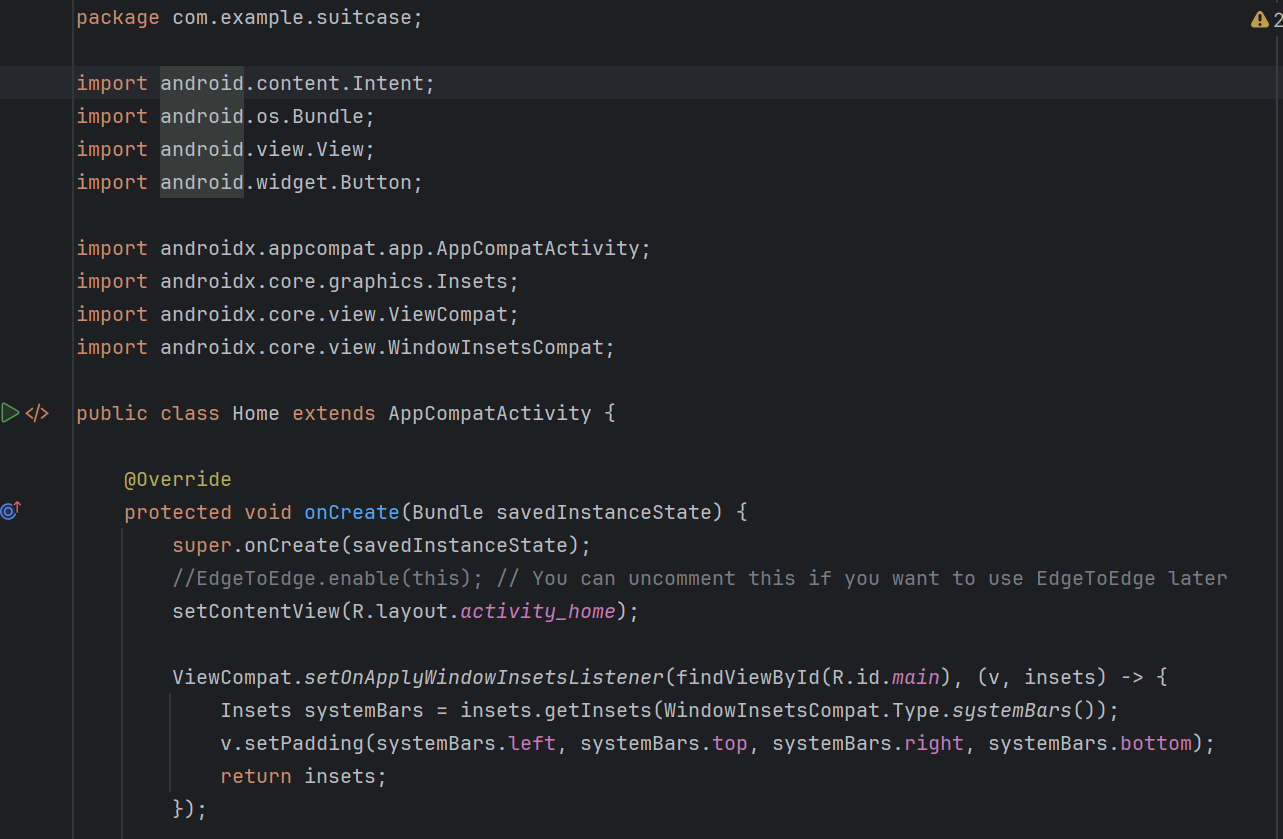
## Splash Screen

The splash screen displays the graphical element of Suitcase app, featuring my unique content with a one-second delay before navigating to the home page.



Home page

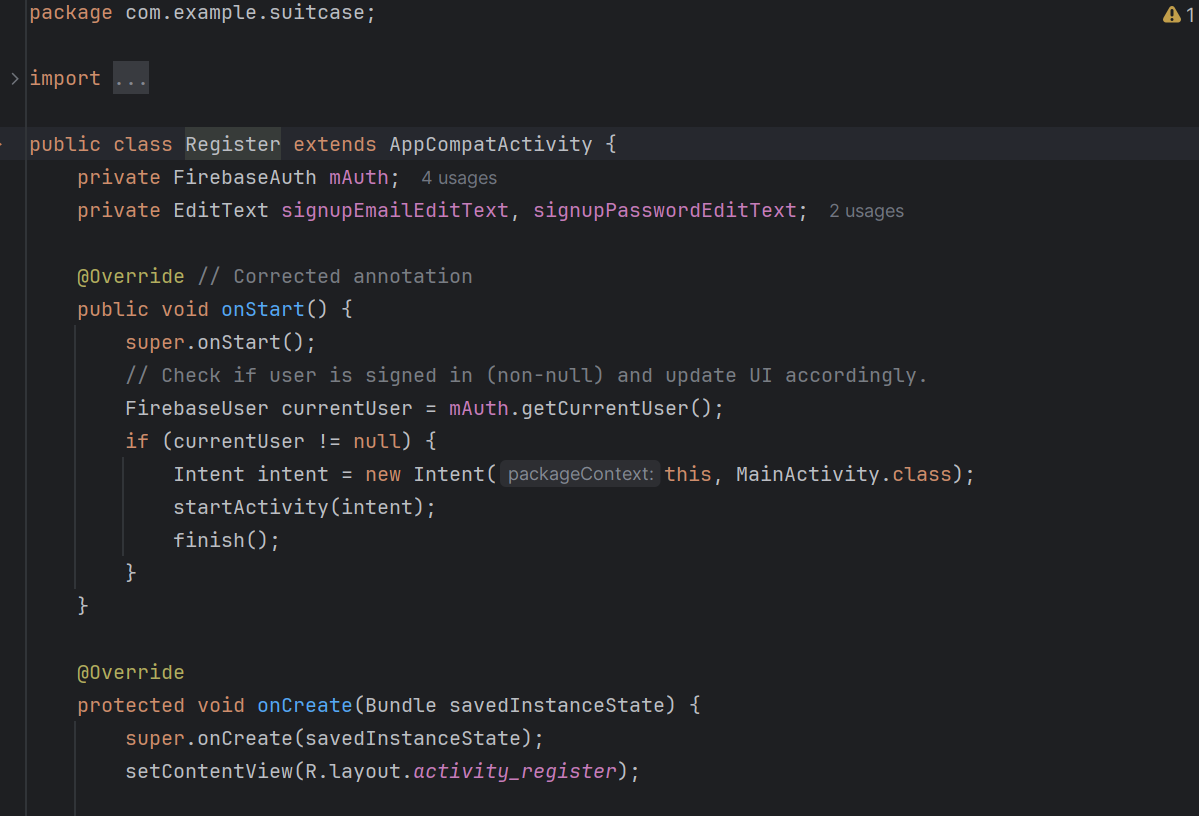
This is my home page which we can see the app name and the image icon of the app. Here users can login or register.

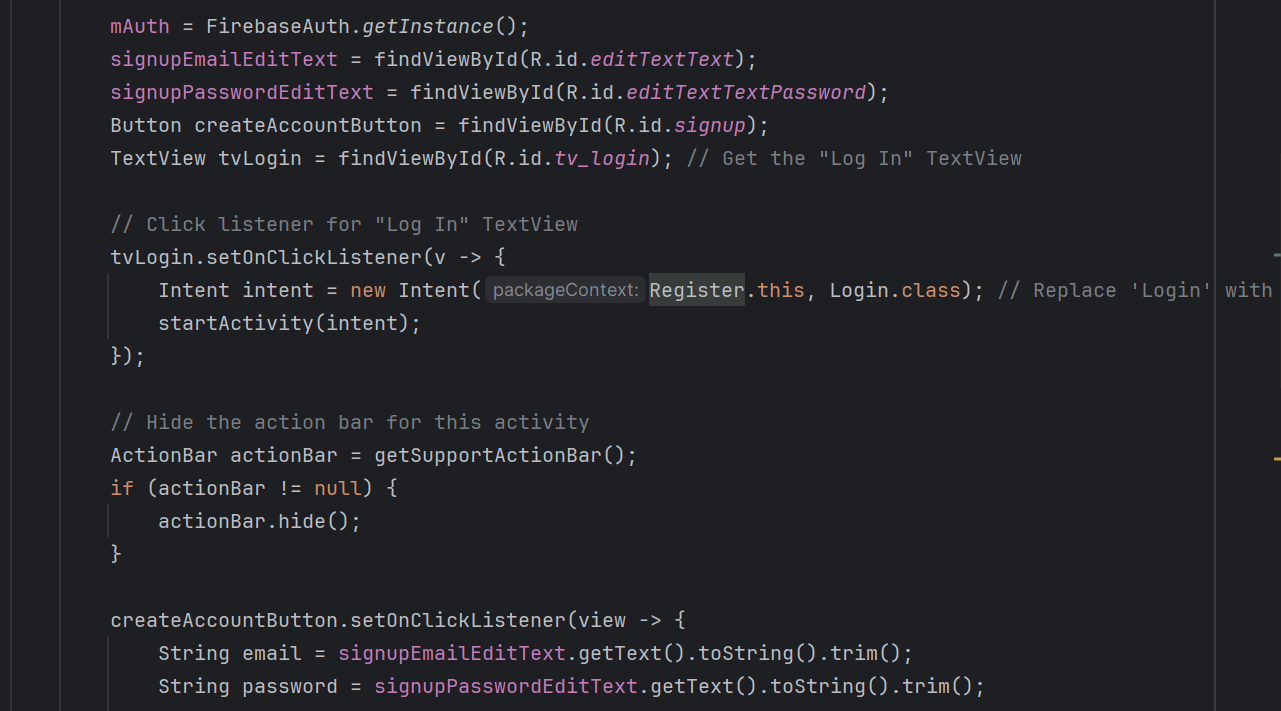


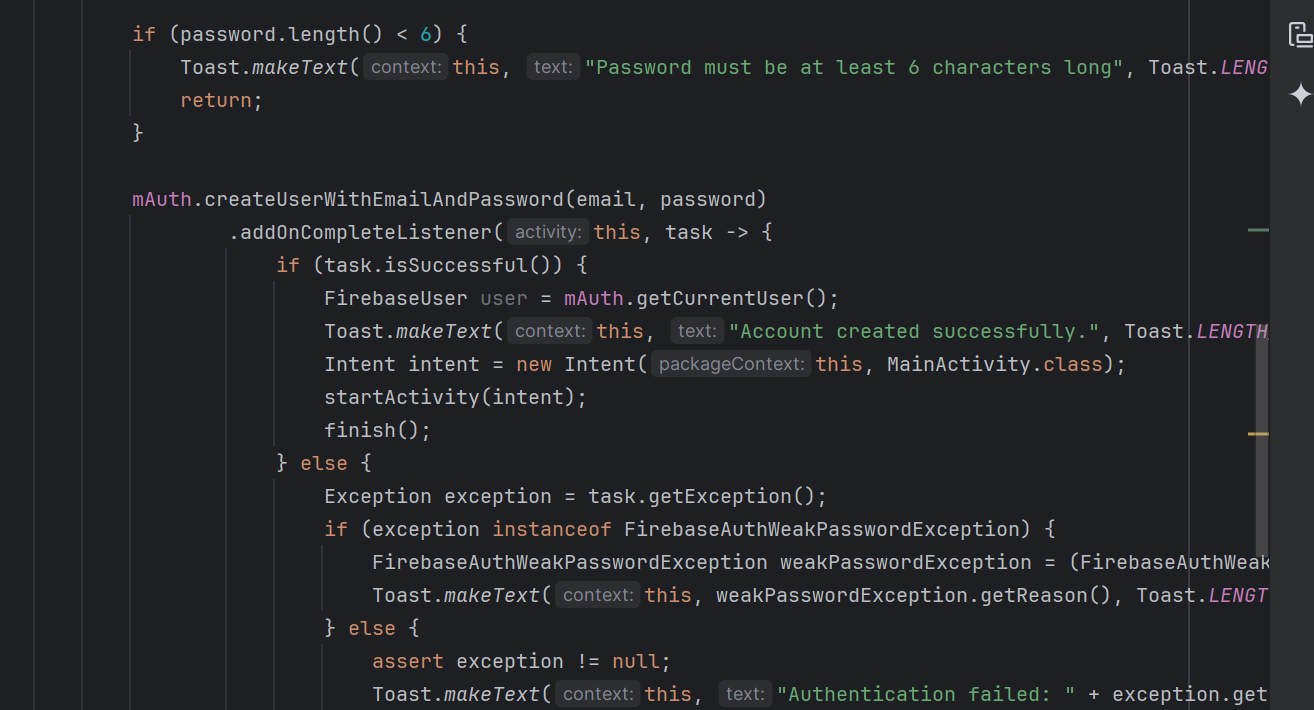


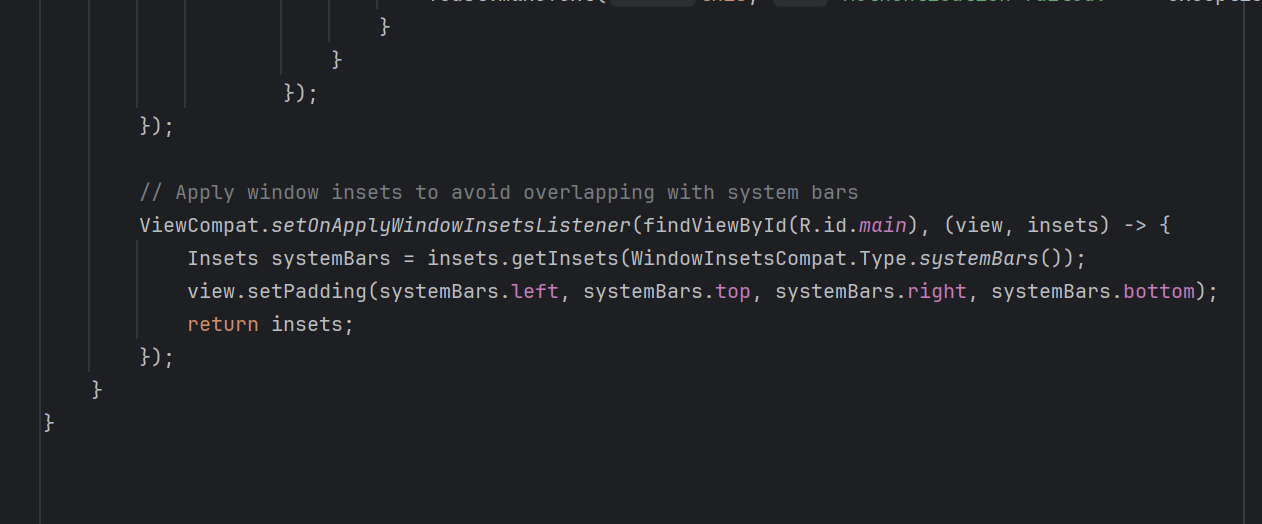
## Registration

The Suitcase application offers a seamless registration process for users to join the application database. To begin, users need to navigate to the registration page by selecting the 'Register' option on the home screen. It is essential to complete all required fields on the registration page to successfully register.



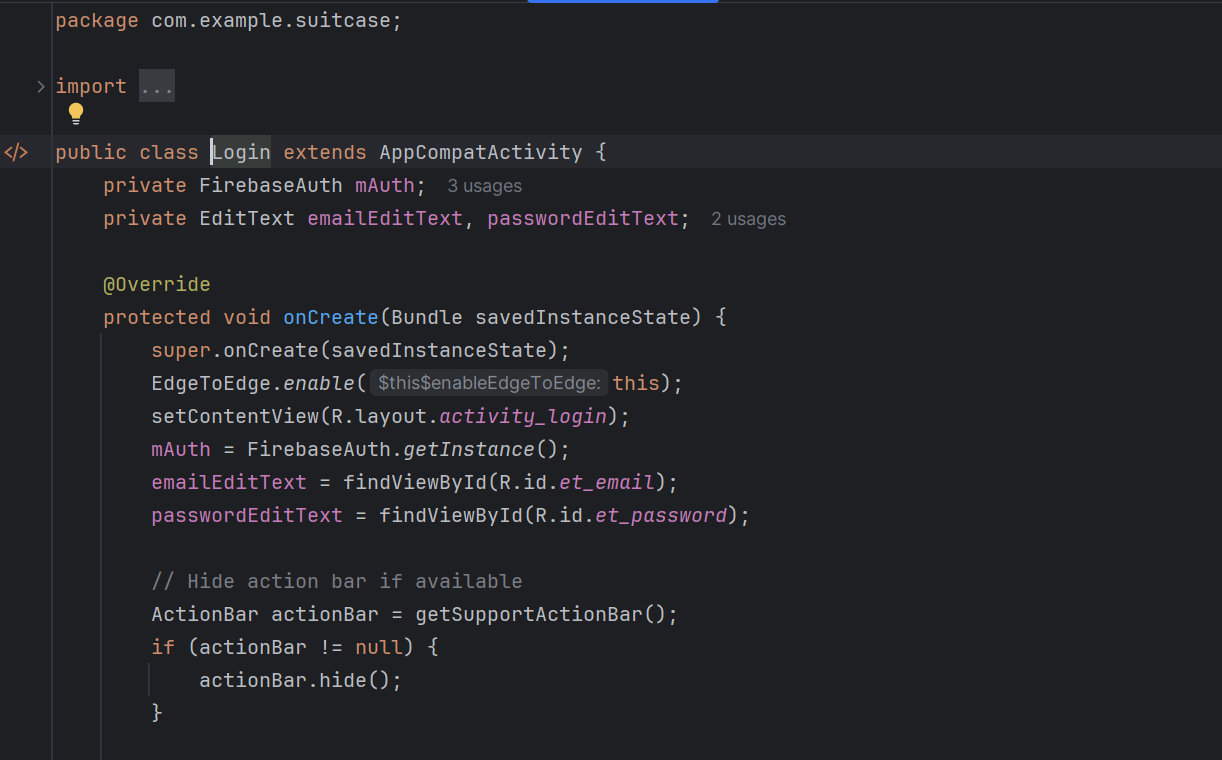






## Login

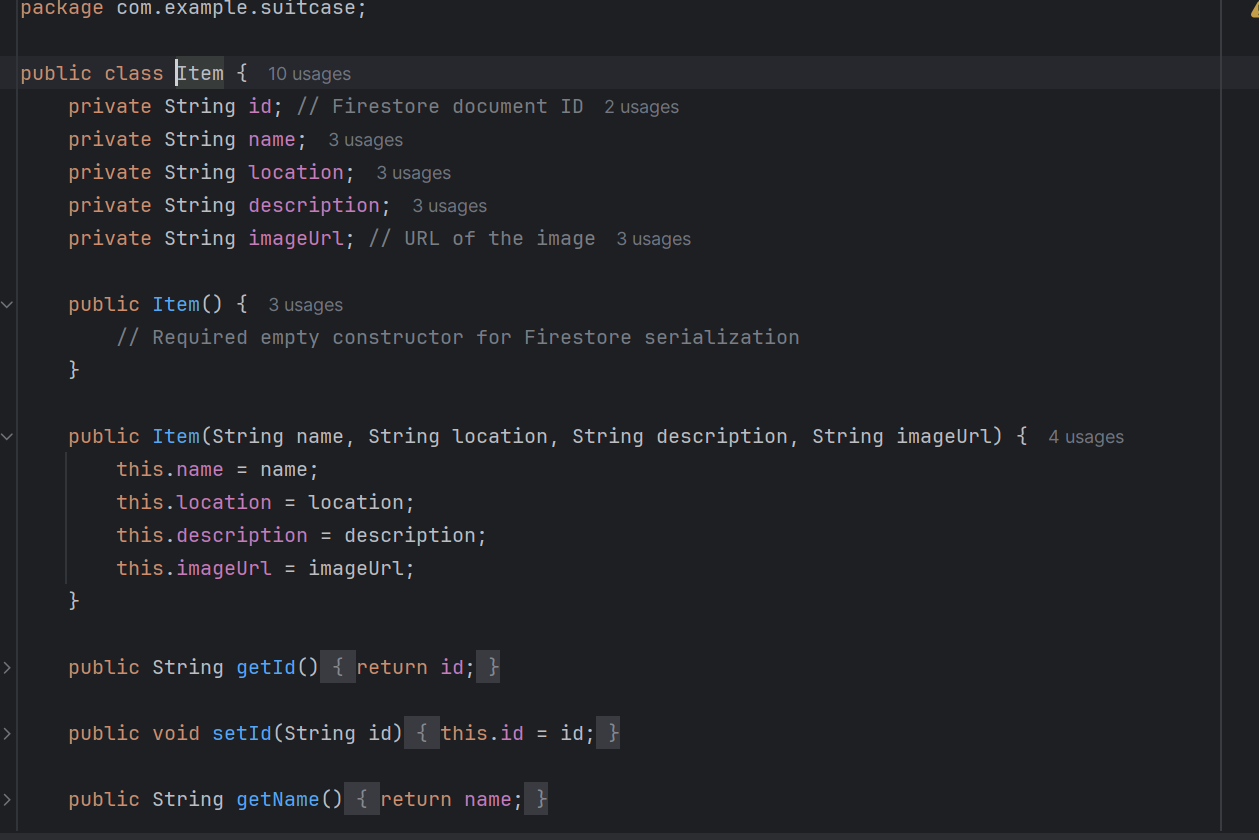
Registered users can log in to the system, gaining access to all features of the application. User credentials are authenticated and authorized, ensuring that only registered users can access their respective dashboards. Users must provide their registered email and password, which will be validated against the data in Firebase for authentication and authorization.

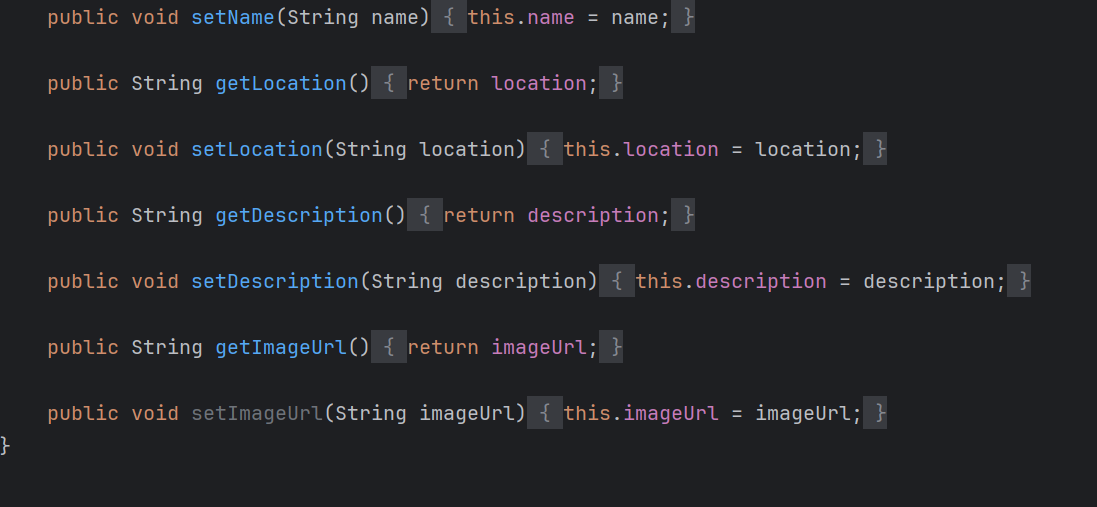




## Add item activity

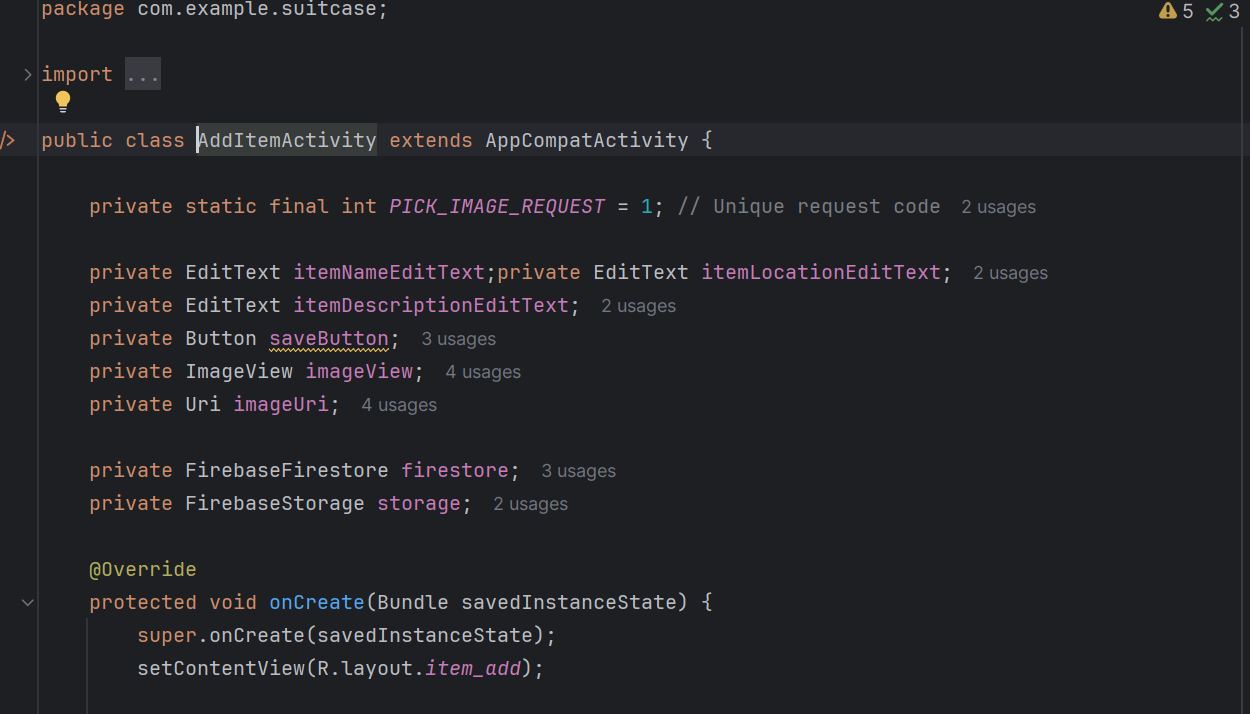
The "Add Item activity" class is responsible for adding new items to your application. It includes several key elements: EditText fields for item name, location, and description, buttons for saving, showing, and selecting images, and an ImageView to display the selected image. Firebase integration is also a crucial component, utilizing FirebaseFirestore for database operations and FirebaseStorage for storing images. During initialization in the onCreate method, the UI is set up with setContentView, and Firebase instances are initialized using FirebaseFirestore.getInstance() and FirebaseStorage.getInstance().

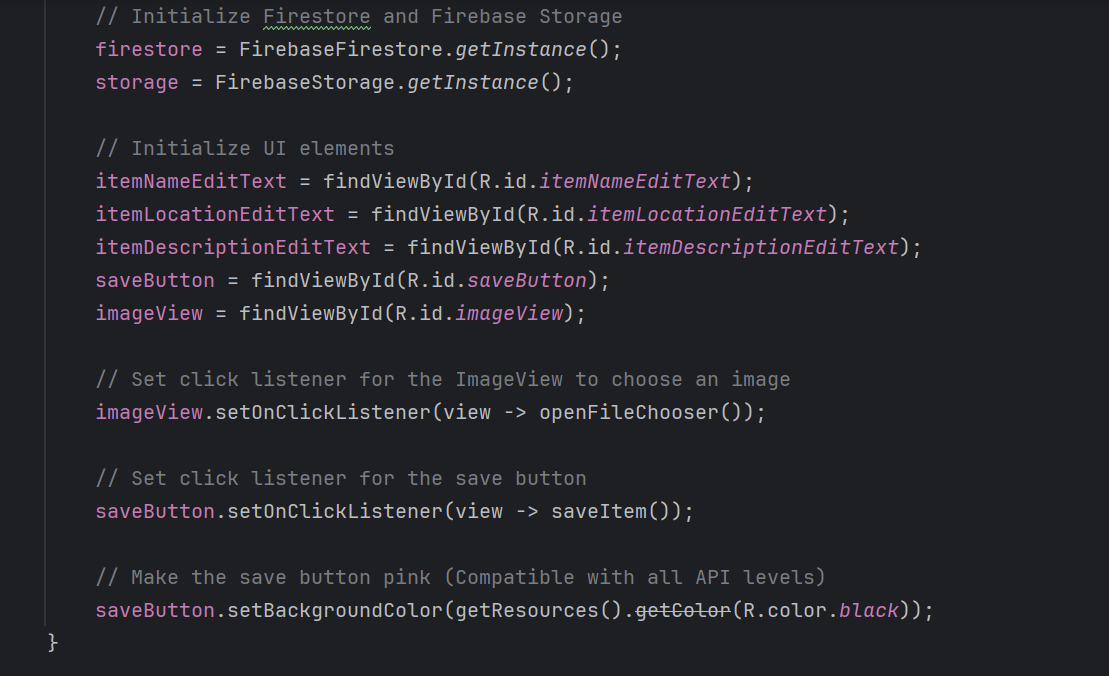


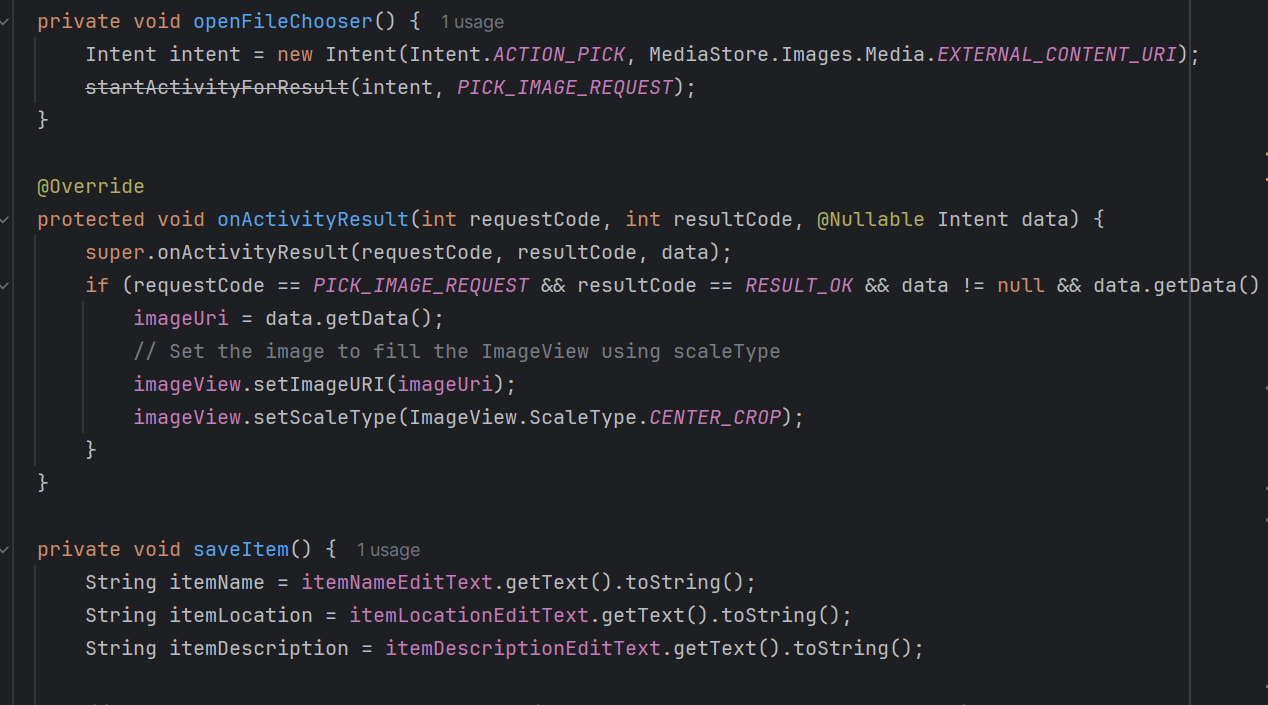


**Item adapter**

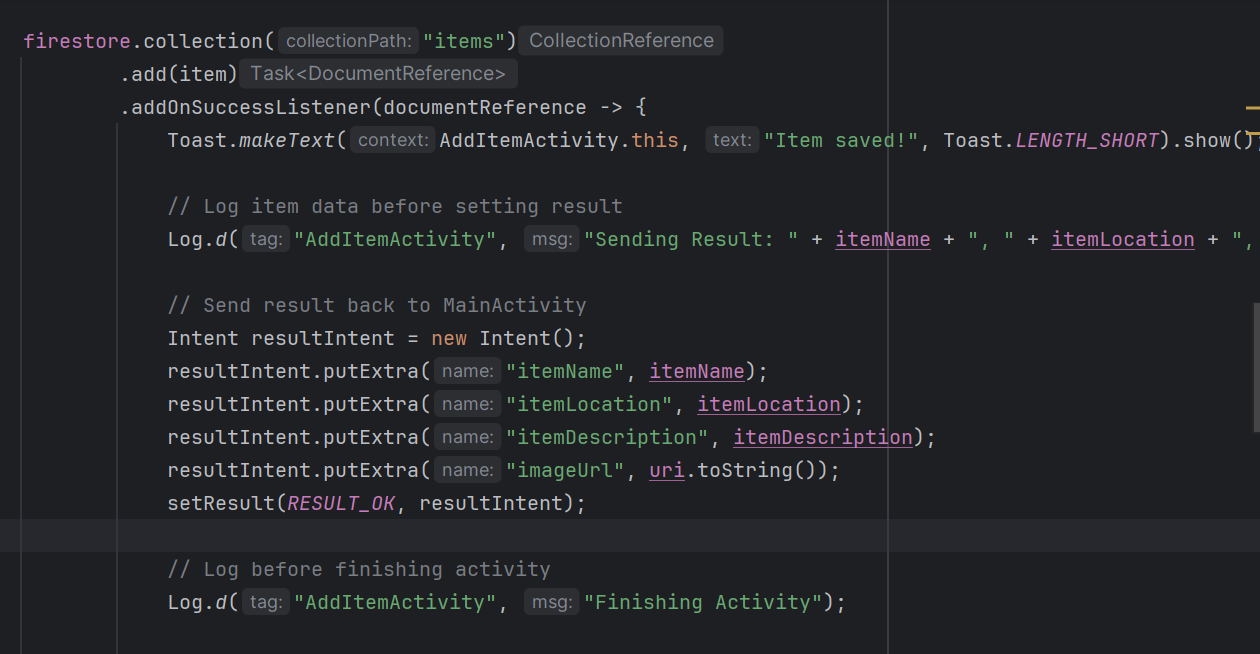
The "ItemAdapter" code snippet is designed to retrieve items from a Firestore collection and update the RecyclerView accordingly. Key elements include Firestore retrieval, where `firestore.collection("items").get()` starts the process, and an addOnCompleteListener is attached to manage success and failure scenarios. The retrieval logic involves clearing the current itemList with `itemList.clear()` upon successful retrieval, iterating through each document in the result to convert it to an Item object, setting the document ID to the Item object, adding the Item object to itemList, and notifying the adapter of the data set change with `itemAdapter.notifyDataSetChanged()`. On retrieval failure, a toast message displays the error: "Failed to fetch items: " + task.getException().getMessage().

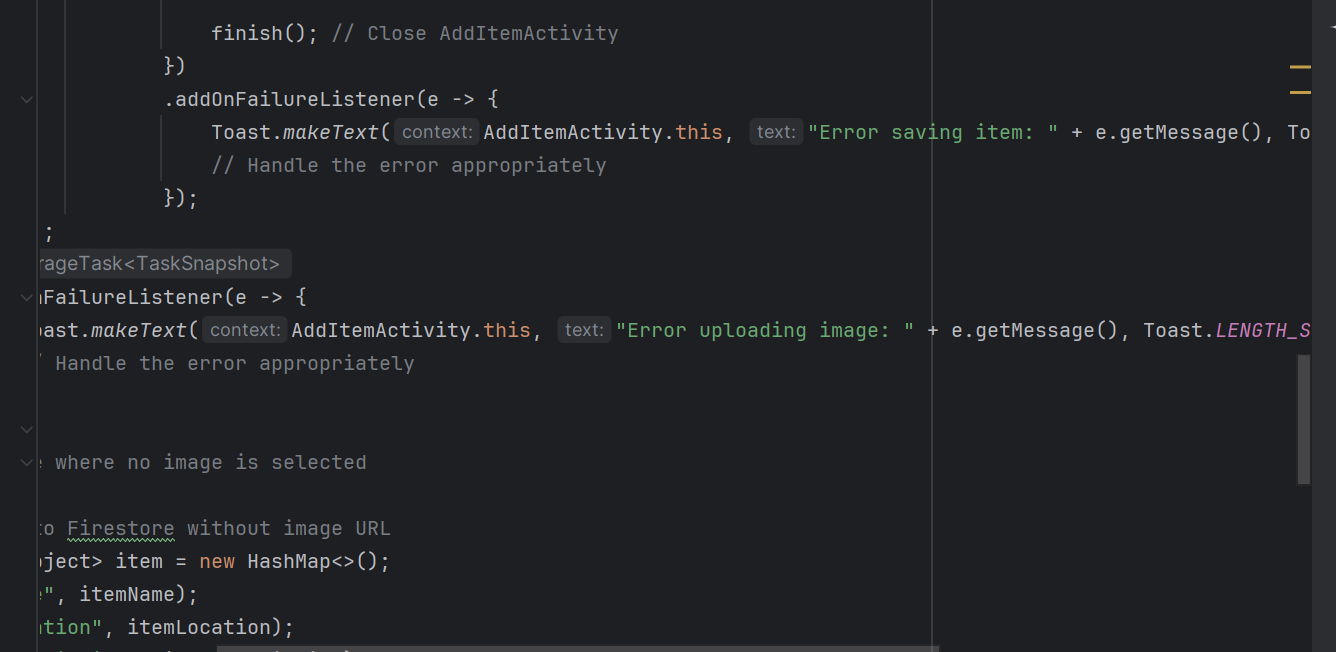


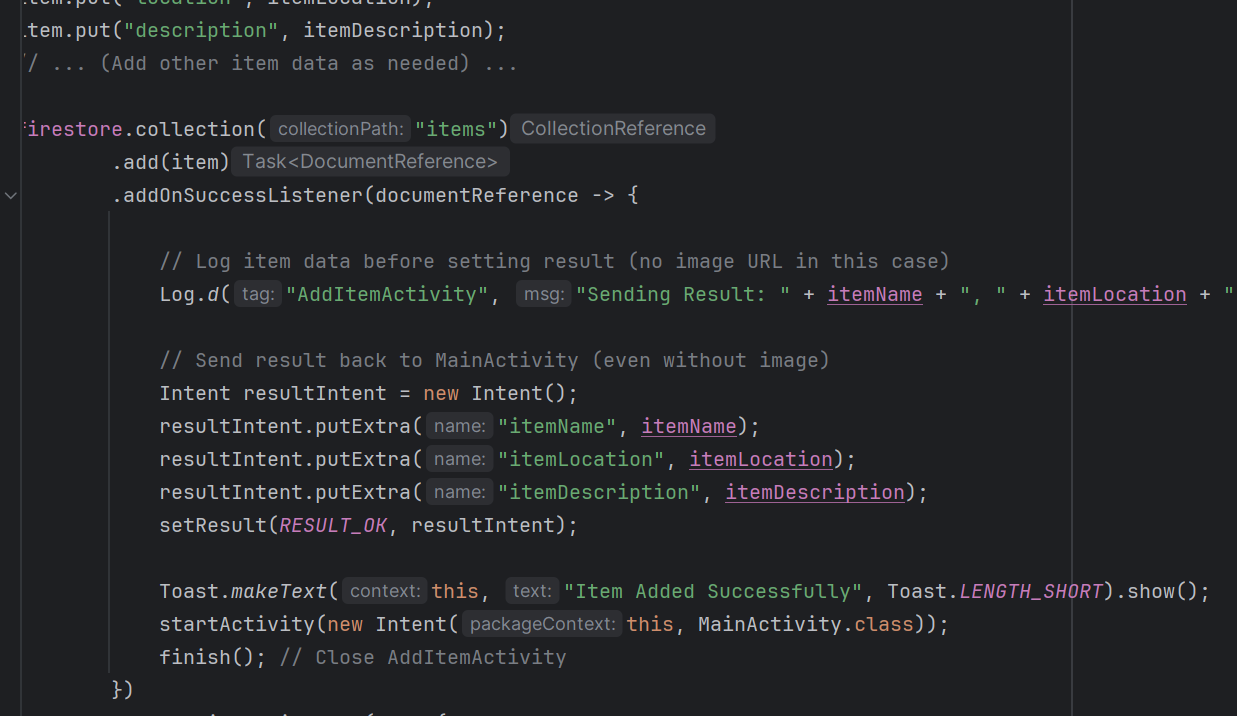








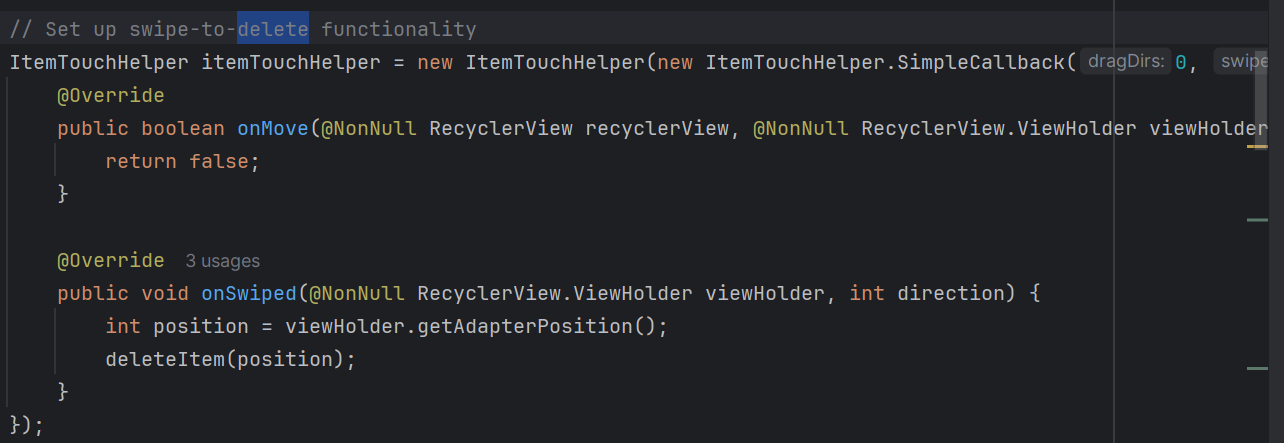






**Swipe to delete**

The "Swipe" code snippet manages the swipe-to-delete gesture in a RecyclerView. Key elements include the swipe gesture, where the onSwiped method of the RecyclerView.ViewHolder is overridden, the position of the swiped item is retrieved with viewHolder.getAdapterPosition(), and the deleteItem method is called with the retrieved position. The swipe logic involves calling the deleteItem(position) method to delete the item at the swiped position.



**Logout**

The "Logout" code snippet details the process for logging out with confirmation. Key elements include using an AlertDialog for logout confirmation. When the logout button is clicked, an alert dialog is initiated with logoutButton.setOnClickListener(v -> new AlertDialog.Builder(this)). The dialog's title is set with .setTitle("Logout Confirmation") and the message is set with .setMessage("Are you sure you want to logout?"). A "No" button dismisses the dialog with .setNegativeButton("No", (dialog, which) -> dialog.dismiss()), and a "Yes" button initiates the logout process with .setPositiveButton("Yes", (dialog, which) -> {...}).

