

```
1 package com.bst.google_signin.presentation.sign_in
2
3 data class SignInState(
4     val isSuccessfull: Boolean = false,
5     val signInError: String? = null
6 )
7
```

```
1 package com.bst.google_signin.presentation.sign_in
```

```
2
```

```
3 data class SignInResult(  
4     val data: UserData?,  
5     val errorMessage: String?  
6 )  
7
```

```
8 data class UserData(  
9     val userId: String,  
10    val username: String?,  
11    val profilePictureUrl: String?  
12 )  
13
```

```
1 package com.bst.google_signin.presentation.signin
2
3 import android.widget.Toast
4 import androidx.compose.foundation.layout.Box
5 import androidx.compose.foundation.layout.fillMaxSize
6 import androidx.compose.foundation.layout.padding
7 import androidx.compose.material.Button
8 import androidx.compose.material.Text
9 import androidx.compose.runtime.Composable
10 import androidx.compose.runtime.LaunchedEffect
11 import androidx.compose.ui.Alignment
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.platform.LocalContext
14 import androidx.compose.ui.unit.dp
15
16 @Composable
17 fun SignInScreen(
18     state: SignInState,
19     onSignInClick: () -> Unit
20 ) {
21     val context = LocalContext.current
22     LaunchedEffect(key1 = state.signInError) {
23         state.signInError?.let { error ->
24             Toast.makeText(
25                 context,
26                 error,
27                 Toast.LENGTH_LONG
```

```
28         ).show()
29     }
30 }
31
32 Box(
33     modifier = Modifier
34         .fillMaxSize()
35         .padding(16.dp),
36     contentAlignment = Alignment.Center
37 ) {
38     Button(onClick = onSignInClick) {
39         Text(text = "Sign in with Google")
40     }
41 }
42 }
```

```
1 package com.bst.google_signin.presentation.signin
2
3 import androidx.lifecycle.ViewModel
4 import kotlinx.coroutines.flow.MutableStateFlow
5 import kotlinx.coroutines.flow.asStateFlow
6 import kotlinx.coroutines.flow.update
7
8 class SignInViewModel: ViewModel() {
9
10     private val _state = MutableStateFlow(SignInState())
11     val state = _state.asStateFlow()
12
13     fun onSignInResult(result: SignInResult) {
14         _state.update { it.copy(
15             isSignInSuccessful = result.data != null,
16             signInError = result.errorMessage
17         ) }
18     }
19
20     fun resetState() {
21         _state.update { SignInState() }
22     }
23 }
```

```
1 package com.bst.google_signin.presentation.signin_in
2
3 import android.content.Context
4 import android.content.Intent
5 import android.content.IntentSender
6 import com.bst.google_signin.R
7 import com.google.android.gms.auth.api.identity.BeginSignInRequest
8 import com.google.android.gms.auth.api.identity.BeginSignInRequest.
    GoogleIdTokenRequestOptions
9 import com.google.android.gms.auth.api.identity.SignInClient
10 import com.google.firebase.auth.GoogleAuthProvider
11 import com.google.firebase.auth.ktx.auth
12 import com.google.firebase.ktx.Firebase
13 import kotlinx.coroutines.CancellationException
14 import kotlinx.coroutines.tasks.await
15
16 class GoogleAuthUiClient(
17     private val context: Context,
18     private val oneTapClient: SignInClient
19 ) {
20     private val auth = Firebase.auth
21
22     suspend fun signIn(): IntentSender? {
23         val result = try {
24             oneTapClient.beginSignIn(
25                 buildSignInRequest()
26             ).await()
```

```
27     } catch (e: Exception) {
28         e.printStackTrace()
29         if (e is CancellationException) throw e
30         null
31     }
32     return result?.pendingIntent?.intentSender
33 }
34
35 suspend fun signInWithIntent(intent: Intent): SignInResult {
36     val credential = oneTapClient.getSignInCredentialFromIntent(intent)
37     val googleIdToken = credential.googleIdToken
38     val googleCredentials = GoogleAuthProvider.getCredential(googleIdToken, null
39 )
40     return try {
41         val user = auth.signInWithCredential(googleCredentials).await().user
42         SignInResult(
43             data = user?.run {
44                 UserData(
45                     userId = uid,
46                     username = displayName,
47                     profilePictureUrl = photoUrl?.toString()
48                 )
49             },
50             errorMessage = null
51         )
52     } catch (e: Exception) {
53         e.printStackTrace()
54     }
```

```
53     if (e is CancellationException) throw e
54     SignInResult(
55         data = null,
56         errorMessage = e.message
57     )
58     }
59 }
60
61 suspend fun signOut() {
62     try {
63         oneTapClient.signOut().await()
64         auth.signOut()
65     } catch (e: Exception) {
66         e.printStackTrace()
67         if (e is CancellationException) throw e
68     }
69 }
70
71 fun getSignedInUser(): UserData? = auth.currentUser?.run {
72     UserData(
73         userId = uid,
74         username = displayName,
75         profilePictureUrl = photoUrl?.toString()
76     )
77 }
78
79 private fun buildSignInRequest(): BeginSignInRequest {
```



```
80     return BeginSignInRequest.Builder()
81         .setGoogleIdTokenRequestOptions(
82             GoogleIdTokenRequestOptions.builder()
83                 .setSupported(true)
84                 .setFilterByAuthorizedAccounts(false)
85                 .setServerClientId(context.getString(R.string.web_client_id))
86                 .build()
87         )
88         .setAutoSelectEnabled(true)
89         .build()
90     }
91 }
```