```kotlin
package com.giri.composegooglesignincleanarchitecture.presentation.sign_in

data class SignInState(
    val isSignInSuccessful: Boolean = false,
    val signInError: String? = null
)

```

```kotlin
package com.giri.composegooglesignincleanarchitecture.presentation.sign_in

data class SignInResult(
    val data: UserData?,
    val errorMessage: String?
)

data class UserData(
    val userId: String,
    val username: String?,
    val profilePictureUrl: String?
)

```

```kotlin
package com.giri.composegooglesignincleanarchitecture.presentation.sign_in

import android.widget.Toast
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material.Button
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp

@Composable
fun SignInScreen(
    state: SignInState,
    onSignInClick: () -> Unit
) {
    val context = LocalContext.current
    LaunchedEffect(key1 = state.signInError) {
        state.signInError?.let { error ->
            Toast.makeText(
                context,
                error,
                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")
40            }
41        }
42    }
```

```kotlin
package com.giri.composegooglesignincleanarchitecture.presentation.sign_in

import androidx.lifecycle.ViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update

class SignInViewModel: ViewModel() {

    private val _state = MutableStateFlow(SignInState())
    val state = _state.asStateFlow()

    fun onSignInResult(result: SignInResult) {
        _state.update { it.copy(
            isSignInSuccessful = result.data != null,
            signInError = result.errorMessage
        ) }
    }

    fun resetState() {
        _state.update { SignInState() }
    }
}
```

```
 1 package com.giri.composegooglesignincleanarchitecture.presentation.sign_in
 2
 3 import android.content.Context
 4 import android.content.Intent
 5 import android.content.IntentSender
 6 import com.google.android.gms.auth.api.identity.BeginSignInRequest
 7 import com.google.android.gms.auth.api.identity.BeginSignInRequest.
   GoogleIdTokenRequestOptions
 8 import com.google.android.gms.auth.api.identity.SignInClient
 9 import com.google.firebase.auth.GoogleAuthProvider
10 import com.google.firebase.auth.ktx.auth
11 import com.google.firebase.ktx.Firebase
12 import com.plcoding.composegooglesignincleanarchitecture.R
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          return try {
40              val user = auth.signInWithCredential(googleCredentials).await().user
41              SignInResult(
42                  data = user?.run {
43                      UserData(
44                          userId = uid,
45                          username = displayName,
46                          profilePictureUrl = photoUrl?.toString()
47                      )
48                  },
49                  errorMessage = null
50              )
51          } catch(e: Exception) {
52              e.printStackTrace()
```

```
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  }
```