

```
1 package com.example.signinwithgoogle
2
3 import android.os.Bundle
4 import android.widget.Toast
5 import androidx.activity.ComponentActivity
6 import androidx.activity.compose.rememberLauncherForActivityResult
7 import androidx.activity.compose.setContent
8 import androidx.activity.result.IntentSenderRequest
9 import androidx.activity.result.contract.ActivityResultContracts
10 import androidx.compose.foundation.layout.fillMaxSize
11 import androidx.compose.material.MaterialTheme
12 import androidx.compose.material.Surface
13 import androidx.compose.runtime.LaunchedEffect
14 import androidx.compose.runtime.getValue
15 import androidx.compose.ui.Modifier
16 import androidx.lifecycle.compose.collectAsStateWithLifecycle
17 import androidx.lifecycle.LifecycleScope
18 import androidx.lifecycle.viewmodel.compose.viewModel
19 import androidx.navigation.compose.NavHost
20 import androidx.navigation.compose.composable
21 import androidx.navigation.compose.rememberNavController
22 import com.google.android.gms.auth.api.identity.Identity
23 import com.example.signinwithgoogle.presentation.profile.ProfileScreen
24 import com.example.signinwithgoogle.presentation.sign_in.GoogleAuthUiClient
25 import com.example.signinwithgoogle.presentation.sign_in.SignInScreen
26 import com.example.signinwithgoogle.presentation.sign_in.SignInViewModel
27 import com.example.signinwithgoogle.theme.ComposeGoogleSignInCleanArchitectureTheme
```

```
28 import kotlinx.coroutines.launch
```

```
29
30 class MainActivity : ComponentActivity() {
```

```
31
32     private val googleAuthUiClient by lazy {
```

```
33         GoogleAuthUiClient(
```

```
34             context = applicationContext,
```

```
35             oneTapClient = Identity.getSignInClient(applicationContext)
```

```
36         )
```

```
37     }
```

```
38
39     override fun onCreate(savedInstanceState: Bundle?) {
```

```
40         super.onCreate(savedInstanceState)
```

```
41         setContent {
```

```
42             ComposeGoogleSignInCleanArchitectureTheme {
```

```
43                 // A surface container using the 'background' color from the theme
44                 Surface(
```

```
45                     modifier = Modifier.fillMaxSize(),
```

```
46                     color = MaterialTheme.colors.background
```

```
47                 ) {
```

```
48                     val navController = rememberNavController()
```

```
49                     NavHost(navController = navController, startDestination = "
```

```
sign_in") {
```

```
50                         composable("sign_in") {
```

```
51                             val viewModel = viewModel<SignInViewModel>()
```

```
52                             val state by viewModel.state.collectAsStateWithLifecycle
```

```
                    ()
```

```
53
54
55     LaunchedEffect(key1 = Unit) {
56         if(googleAuthUiClient.getSignedInUser() != null) {
57             navController.navigate("profile")
58         }
59     }
60
61     val launcher = rememberLauncherForActivityResult(
62         contract = ActivityResultContracts.
63             StartIntentSenderForResult(),
64             onStartResult = { result ->
65                 if(result.resultCode == RESULT_OK) {
66                     lifecycleScope.launch {
67                         val signInResult = googleAuthUiClient.
68                             signInWithIntent(
69                                 intent = result.data ?: return@
70                                     launch
71                                     )
72                         viewModel.onSignInResult(signInResult)
73                     }
74                 }
75             }
76     )
77
78     LaunchedEffect(key1 = state.isSignInSuccessful) {
79         if(state.isSignInSuccessful) {
80             Toast.makeText(
```

```
77         applicationContext,
78         "Sign in successful",
79         Toast.LENGTH_LONG
80     ).show()
81
82     navController.navigate("profile")
83     viewModel.resetState()
84
85 }
86
87 SignInScreen(
88     state = state,
89     onSignInClick = {
90         lifecycleScope.launch {
91             val signInIntentSender = googleAuthUiClient
92                 .signIn()
93                 .launcher.launch(
94                     IntentSenderRequest.Builder(
95                         signInIntentSender ?: return@launch
96                             ).build()
97                     )
98                 }
99         )
100     }
101     composable("profile") {
102         ProfileScreen(
```

```
1103     userData = googleAuthUiClient.getSignedInUser(),
1104     onSignOut = {
1105         lifecycleScope.launch {
1106             googleAuthUiClient.signOut()
1107             Toast.makeText(
1108                 applicationContext,
1109                 "Signed out",
1110                 Toast.LENGTH_LONG
1111             ).show()
1112         }
1113         navController.popBackStack()
1114     }
1115 }
1116 )
1117 }
1118 }
1119 }
1120 }
1121 }
1122 }
1123 }
```

```
1 package com.example.signinwithgoogle.theme
2
3 import androidx.compose.material.Typography
4 import androidx.compose.ui.text.TextStyle
5 import androidx.compose.ui.text.font.FontFamily
6 import androidx.compose.ui.text.font.FontWeight
7 import androidx.compose.ui.unit.sp
8
9 // Set of Material typography styles to start with
10 val Typography = Typography(
11     body1 = TextStyle(
12         fontFamily = FontFamily.Default,
13         fontWeight = FontWeight.Normal,
14         fontSize = 16.sp
15     )
16     /* Other default text styles to override
17     button = TextStyle(
18         fontFamily = FontFamily.Default,
19         fontWeight = FontWeight.W500,
20         fontSize = 14.sp
21     ),
22     caption = TextStyle(
23         fontFamily = FontFamily.Default,
24         fontWeight = FontWeight.Normal,
25         fontSize = 12.sp
26     )
27 */
```



```
1 package com.example.signinwithgoogle.theme
2
3 import androidx.compose.ui.graphics.Color
4
5 val Purple200 = Color(0xFFBB86FC)
6 val Purple500 = Color(0xFF6200EE)
7 val Purple700 = Color(0xFF3700B3)
8 val Teal200 = Color(0xFF03DAC5)
```



```
1 package com.example.signinwithgoogle.theme
2
3 import androidx.compose.foundation.shape.RoundedCornerShape
4 import androidx.compose.material.Shapes
5 import androidx.compose.ui.unit.dp
6
7 val Shapes = Shapes(
8     small = RoundedCornerShape(4.dp),
9     medium = RoundedCornerShape(4.dp),
10    large = RoundedCornerShape(0.dp)
11 )
```

```
1 package com.example.signinwithgoogle.theme
2
3 import androidx.compose.foundation.isSystemInDarkTheme
4 import androidx.compose.material.MaterialTheme
5 import androidx.compose.material.darkColors
6 import androidx.compose.material.lightColors
7 import androidx.compose.runtime.Composable
8
9 private val DarkColorPalette = darkColors(
10     primary = Purple200,
11     primaryVariant = Purple700,
12     secondary = Teal200
13 )
14
15 private val LightColorPalette = lightColors(
16     primary = Purple500,
17     primaryVariant = Purple700,
18     secondary = Teal200
19
20     /* Other default colors to override
21     background = Color.White,
22     surface = Color.White,
23     onPrimary = Color.White,
24     onSecondary = Color.Black,
25     onBackground = Color.Black,
26     onSurface = Color.Black,
27     */
28 )
```

```
28 )
29
30 @Composable
31 fun ComposeGoogleSignInCleanArchitectureTheme(
32     darkTheme: Boolean = isSystemInDarkTheme(),
33     content: @Composable () -> Unit
34 ) {
35     val colors = if (darkTheme) {
36         DarkColorPalette
37     } else {
38         LightColorPalette
39     }
40
41     MaterialTheme(
42         colors = colors,
43         typography = Typography,
44         shapes = Shapes,
45         content = content
46     )
47 }
```

```
1 package com.example.signinwithgoogle.presentation.profile
2
3 import androidx.compose.foundation.layout.*
4 import androidx.compose.foundation.shape.CircleShape
5 import androidx.compose.material.Button
6 import androidx.compose.material.Text
7 import androidx.compose.runtime.Composable
8 import androidx.compose.ui.Alignment
9 import androidx.compose.ui.Modifier
10 import androidx.compose.ui.draw.clip
11 import androidx.compose.ui.layout.ContentScale
12 import androidx.compose.ui.text.font.FontWeight
13 import androidx.compose.ui.text.style.TextAlign
14 import androidx.compose.ui.unit.dp
15 import androidx.compose.ui.unit.sp
16 import coil.compose.AsyncImage
17 import com.example.signinwithgoogle.presentation.sign_in.UserData
18
19 @Composable
20 fun ProfileScreen(
21     userData: UserData?,
22     onSignOut: () -> Unit
23 ) {
24     Column(
25         modifier = Modifier.fillMaxSize(),
26         verticalArrangement = Arrangement.Center,
27         horizontalAlignment = Alignment.CenterHorizontally
28     ) {
29         AsyncImage(
30             model = userData?.profilePicture,
31             contentDescription = "Profile picture of the user",
32             placeholder = androidx.compose.ui.graphics.Color.Gray,
33             modifier = Modifier
34                 .fillMaxWidth()
35                 .height(100.dp)
36                 .clip(CircleShape)
37         )
38     }
39 }
```

```
28 ) {
29     if(userData?.profilePictureUrl != null) {
30         AsyncImage(
31             model = userData.profilePictureUrl,
32             contentDescription = "Profile picture",
33             modifier = Modifier
34                 .size(150.dp)
35                 .clip(CircleShape),
36             contentScale = ContentScale.Crop
37         )
38         Spacer(modifier = Modifier.height(16.dp))
39     }
40     if(userData?.username != null) {
41         Text(
42             text = userData.username,
43             textAlign = TextAlign.Center,
44             fontSize = 36.sp,
45             fontWeight = FontWeight.SemiBold
46         )
47         Spacer(modifier = Modifier.height(16.dp))
48     }
49     Button(onClick = onSignOut) {
50         Text(text = "Sign out")
51     }
52 }
53 }
```

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

```
1 package com.example.signinwithgoogle.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.example.signinwithgoogle.presentation.sign_in
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")
40         }
41     }
42 }
```

```
1 package com.example.signinwithgoogle.presentation.sign_in
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.example.signinwithgoogle.presentation.sign_in
2
3 import android.content.Context
4 import android.content.Intent
5 import android.content.IntentSender
6 import com.example.signinwithgoogle.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, private val oneTapClient: SignInClient
18 ) {
19     private val auth = Firebase.auth
20
21     suspend fun signIn(): IntentSender? {
22         val result = try {
23             oneTapClient.beginSignIn(
24                 buildSignInRequest()
25             ).await()
26         } catch (e: Exception) {
```

```
27         e.printStackTrace()
28         if (e is CancellationException) throw e
29         null
30     }
31     return result?.pendingIntent?.intentSender
32 }
33
34 suspend fun signInWithIntent(intent: Intent): SignInResult {
35     val credential = oneTapClient.getSignInCredentialFromIntent(intent)
36     val googleIdToken = credential.googleIdToken
37     val googleCredentials = GoogleAuthProvider.getCredential(googleIdToken, null
38 )
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                 ), errorMessage = null
48             )
49     } catch (e: Exception) {
50         e.printStackTrace()
51         if (e is CancellationException) throw e
52         SignInResult(
```

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

```
78 ( )
79     ).setAutoSelectEnabled(true).build()
80 }
81 }
```