```kotlin
package com.example.signinwithgoogle

import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.compose.setContent
import androidx.activity.result.IntentSenderRequest
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.getValue
import androidx.compose.ui.Modifier
import androidx.lifecycle.compose.collectAsStateWithLifecycle
import androidx.lifecycle.lifecycleScope
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.google.android.gms.auth.api.identity.Identity
import com.example.signinwithgoogle.presentation.profile.ProfileScreen
import com.example.signinwithgoogle.presentation.sign_in.GoogleAuthUiClient
import com.example.signinwithgoogle.presentation.sign_in.SignInScreen
import com.example.signinwithgoogle.presentation.sign_in.SignInViewModel
import com.example.signinwithgoogle.theme.ComposeGoogleSignInCleanArchitectureTheme
import kotlinx.coroutines.launch

class MainActivity : ComponentActivity() {

    private val googleAuthUiClient by lazy {
        GoogleAuthUiClient(
            context = applicationContext,
            oneTapClient = Identity.getSignInClient(applicationContext)
        )
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ComposeGoogleSignInCleanArchitectureTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    val navController = rememberNavController()
                    NavHost(navController = navController, startDestination = "sign_in") {
                        composable("sign_in") {
                            val viewModel = viewModel<SignInViewModel>()
                            val state by viewModel.state.collectAsStateWithLifecycle()

                            LaunchedEffect(key1 = Unit) {
                                if(googleAuthUiClient.getSignedInUser() != null) {
                                    navController.navigate("profile")
                                }
                            }

                            val launcher = rememberLauncherForActivityResult(
                                contract = ActivityResultContracts.StartIntentSenderForResult(),
                                onResult = { result ->
                                    if(result.resultCode == RESULT_OK) {
                                        lifecycleScope.launch {
                                            val signInResult = googleAuthUiClient.signInWithIntent(
                                                intent = result.data ?: return@launch
                                            )
                                            viewModel.onSignInResult(signInResult)
                                        }
                                    }
                                }
                            )

                            LaunchedEffect(key1 = state.isSignInSuccessful) {
                                if(state.isSignInSuccessful) {
                                    Toast.makeText(
                                        applicationContext,
                                        "Sign in successful",
                                        Toast.LENGTH_LONG
                                    ).show()

                                    navController.navigate("profile")
                                    viewModel.resetState()
                                }
                            }

                            SignInScreen(
                                state = state,
                                onSignInClick = {
                                    lifecycleScope.launch {
                                        val signInIntentSender = googleAuthUiClient.signIn()
                                        launcher.launch(
                                            IntentSenderRequest.Builder(
                                                signInIntentSender ?: return@launch
                                            ).build()
                                        )
                                    }
                                }
                            )
                        }
                        composable("profile") {
                            ProfileScreen(
                                userData = googleAuthUiClient.getSignedInUser(),
                                onSignOut = {
                                    lifecycleScope.launch {
                                        googleAuthUiClient.signOut()
                                        Toast.makeText(
                                            applicationContext,
                                            "Signed out",
                                            Toast.LENGTH_LONG
                                        ).show()

                                        navController.popBackStack()
                                    }
                                }
                            )
                        }
                    }
                }
            }
        }
    }
}
```

```kotlin
package com.example.signinwithgoogle.theme

import androidx.compose.material.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    body1 = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
    /* Other default text styles to override
    button = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.W500,
        fontSize = 14.sp
    ),
    caption = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 12.sp
    )
    */
)
```

```kotlin
package com.example.signinwithgoogle.theme

import androidx.compose.ui.graphics.Color

val Purple200 = Color(0xFFBB86FC)
val Purple500 = Color(0xFF6200EE)
val Purple700 = Color(0xFF3700B3)
val Teal200 = Color(0xFF03DAC5)
```

```kotlin
package com.example.signinwithgoogle.theme

import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.Shapes
import androidx.compose.ui.unit.dp

val Shapes = Shapes(
    small = RoundedCornerShape(4.dp),
    medium = RoundedCornerShape(4.dp),
    large = RoundedCornerShape(0.dp)
)
```

```kotlin
package com.example.signinwithgoogle.theme

import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material.MaterialTheme
import androidx.compose.material.darkColors
import androidx.compose.material.lightColors
import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(
    primary = Purple200,
    primaryVariant = Purple700,
    secondary = Teal200
)

private val LightColorPalette = lightColors(
    primary = Purple500,
    primaryVariant = Purple700,
    secondary = Teal200

    /* Other default colors to override
    background = Color.White,
    surface = Color.White,
    onPrimary = Color.White,
    onSecondary = Color.Black,
    onBackground = Color.Black,
    onSurface = Color.Black,
    */
)

@Composable
fun ComposeGoogleSignInCleanArchitectureTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable () -> Unit
) {
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }

    MaterialTheme(
        colors = colors,
        typography = Typography,
        shapes = Shapes,
        content = content
    )
}
```

```kotlin
package com.example.signinwithgoogle.presentation.profile

import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.material.Button
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import coil.compose.AsyncImage
import com.example.signinwithgoogle.presentation.sign_in.UserData

@Composable
fun ProfileScreen(
    userData: UserData?,
    onSignOut: () -> Unit
) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        if(userData?.profilePictureUrl != null) {
            AsyncImage(
                model = userData.profilePictureUrl,
                contentDescription = "Profile picture",
                modifier = Modifier
                    .size(150.dp)
                    .clip(CircleShape),
                contentScale = ContentScale.Crop
            )
            Spacer(modifier = Modifier.height(16.dp))
        }
        if(userData?.username != null) {
            Text(
                text = userData.username,
                textAlign = TextAlign.Center,
                fontSize = 36.sp,
                fontWeight = FontWeight.SemiBold
            )
            Spacer(modifier = Modifier.height(16.dp))
        }
        Button(onClick = onSignOut) {
            Text(text = "Sign out")
        }
    }
}
```

```kotlin
package com.example.signinwithgoogle.presentation.sign_in

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

```

```kotlin
package com.example.signinwithgoogle.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.example.signinwithgoogle.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
            ).show()
        }
    }

    Box(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        contentAlignment = Alignment.Center
    ) {
        Button(onClick = onSignInClick) {
            Text(text = "Sign in")
        }
    }
}
```

```kotlin
package com.example.signinwithgoogle.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() }
    }
}
```

```kotlin
package com.example.signinwithgoogle.presentation.sign_in

import android.content.Context
import android.content.Intent
import android.content.IntentSender
import com.example.signinwithgoogle.R
import com.google.android.gms.auth.api.identity.BeginSignInRequest
import com.google.android.gms.auth.api.identity.BeginSignInRequest.
GoogleIdTokenRequestOptions
import com.google.android.gms.auth.api.identity.SignInClient
import com.google.firebase.auth.GoogleAuthProvider
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import kotlinx.coroutines.CancellationException
import kotlinx.coroutines.tasks.await

class GoogleAuthUiClient(
    private val context: Context, private val oneTapClient: SignInClient
) {
    private val auth = Firebase.auth

    suspend fun signIn(): IntentSender? {
        val result = try {
            oneTapClient.beginSignIn(
                buildSignInRequest()
            ).await()
        } catch (e: Exception) {
            e.printStackTrace()
            if (e is CancellationException) throw e
            null
        }
        return result?.pendingIntent?.intentSender
    }

    suspend fun signInWithIntent(intent: Intent): SignInResult {
        val credential = oneTapClient.getSignInCredentialFromIntent(intent)
        val googleIdToken = credential.googleIdToken
        val googleCredentials = GoogleAuthProvider.getCredential(googleIdToken, null)
        return try {
            val user = auth.signInWithCredential(googleCredentials).await().user
            SignInResult(
                data = user?.run {
                    UserData(
                        userId = uid,
                        username = displayName,
                        profilePictureUrl = photoUrl?.toString()
                    )
                }, errorMessage = null
            )
        } catch (e: Exception) {
            e.printStackTrace()
            if (e is CancellationException) throw e
            SignInResult(
                data = null, errorMessage = e.message
            )
        }
    }

    suspend fun signOut() {
        try {
            oneTapClient.signOut().await()
            auth.signOut()
        } catch (e: Exception) {
            e.printStackTrace()
            if (e is CancellationException) throw e
        }
    }

    fun getSignedInUser(): UserData? = auth.currentUser?.run {
        UserData(
            userId = uid, username = displayName, profilePictureUrl = photoUrl?.toString()
        )
    }

    private fun buildSignInRequest(): BeginSignInRequest {
        return BeginSignInRequest.Builder().setGoogleIdTokenRequestOptions(
            GoogleIdTokenRequestOptions.builder().setSupported(true)
                .setFilterByAuthorizedAccounts(false)
                .setServerClientId(context.getString(R.string.web_client_id)).build
```

```
()
            ).setAutoSelectEnabled(true).build()
    }
}
```