FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

Department of Computer Engineering

Class	**T.E. Computer A**
Subject Name	**Mobile Computing**

Practical No.	7
Title	To develop an android application that creates an alert
Date of Performance	21/04/2025
Date of Submission	25/04/2025
Roll No.	9914
Name	Vivian

Evaluation:

Sr. No	Rubric	Grade
1	**Timeline(2)**	
2	**Output(3)**	
3	**Code Optimization(3)**	
4	**Knowledge of the topic(2)**	
5	**Total(10)**	

Signature of the teacher:

CODE:

```
//MainActivity.kt
```

```
package com.example.notifier

import android.Manifest
import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import android.content.pm.PackageManager
import android.os.Build
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.contract.ActivityResultContracts
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.getValue
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat
import androidx.core.content.ContextCompat
import kotlin.random.Random

class MainActivity : ComponentActivity() {
    private val CHANNEL_ID = "notification_channel"

    // List of quotes to display in notifications
    private val quotes = listOf(
        "The only way to do great work is to love what you do. - Steve Jobs",
        "Life is what happens when you're busy making other plans. - John
Lennon",
        "The future belongs to those who believe in the beauty of their
dreams. - Eleanor Roosevelt",
        "Success is not final, failure is not fatal: It is the courage to
continue that counts. - Winston Churchill",
        "The best time to plant a tree was 20 years ago. The second best time
is now. - Chinese Proverb",
        "Your time is limited, don't waste it living someone else's life. -
Steve Jobs",
        "If you set your goals ridiculously high and it's a failure, you will
fail above everyone else's success. - James Cameron",
        "It does not matter how slowly you go as long as you do not stop. -
Confucius",
        "Everything you've ever wanted is on the other side of fear. - George
Addair",
        "Success usually comes to those who are too busy to be looking for it.
- Henry David Thoreau"
    )

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Create notification channel (required for Android 8.0 and above)
        createNotificationChannel()

        setContent {
            MaterialTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    NotificationScreen(CHANNEL_ID, quotes)
                }
            }
        }
    }

    private fun createNotificationChannel() {
        // Create the NotificationChannel, but only on API 26+ (Android 8.0)
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val name = "Quotes Channel"
            val descriptionText = "A channel for motivational quotes"
            val importance = NotificationManager.IMPORTANCE_DEFAULT
            val channel = NotificationChannel(CHANNEL_ID, name,
importance).apply {
                description = descriptionText
            }
            // Register the channel with the system
            val notificationManager: NotificationManager =
                getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager
            notificationManager.createNotificationChannel(channel)
        }
    }
}

@Composable
```

```
fun NotificationScreen(channelId: String, quotes: List<String>) {
    val context = LocalContext.current
    var hasNotificationPermission by remember {
        mutableStateOf(
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
                ContextCompat.checkSelfPermission(
                    context,
                    Manifest.permission.POST_NOTIFICATIONS
                ) == PackageManager.PERMISSION_GRANTED
            } else {
                true // Permission not required for Android < 13
            }
        )
    }

    // Request permission launcher
    val permissionLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.RequestPermission()
    ) { isGranted ->
        hasNotificationPermission = isGranted
        if (isGranted) {
            showQuoteNotification(context, channelId, quotes)
        } else {
            Toast.makeText(
                context,
                "Notification permission denied",
                Toast.LENGTH_SHORT
            ).show()
        }
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Button(
            onClick = {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
                    if (hasNotificationPermission) {
                        showQuoteNotification(context, channelId, quotes)
                    } else {

permissionLauncher.launch(Manifest.permission.POST_NOTIFICATIONS)
                    }
                } else {
                    showQuoteNotification(context, channelId, quotes)
                }
            }
        ) {
```

```kotlin
                Text("Send Quote Notification")
            }
        }
    }

private fun showQuoteNotification(context: Context, channelId: String, quotes:
List<String>) {
    // Select a random quote from the list
    val randomQuote = quotes[Random.nextInt(quotes.size)]

    // Split the quote into the quote text and author
    val parts = randomQuote.split(" - ")
    val quoteText = parts[0]
    val author = if (parts.size > 1) parts[1] else "Unknown"

    // Build the notification
    val builder = NotificationCompat.Builder(context, channelId)
        .setSmallIcon(android.R.drawable.ic_dialog_info)
        .setContentTitle("Quote of the Moment")
        .setContentText(quoteText)
        .setStyle(NotificationCompat.BigTextStyle()
            .bigText(quoteText)
            .setSummaryText("- $author"))
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)

    // Show the notification with a unique ID so multiple quotes can be
displayed
    val notificationId = Random.nextInt(1000)
    with(NotificationManagerCompat.from(context)) {
        try {
            // Post notification
            notify(notificationId, builder.build())
        } catch (e: SecurityException) {
            // This should not happen if we've properly checked permissions
            Toast.makeText(
                context,
                "Failed to show notification: ${e.message}",
                Toast.LENGTH_SHORT
            ).show()
            e.printStackTrace()
        }
    }
}
```

```
//libs.versions.toml
```

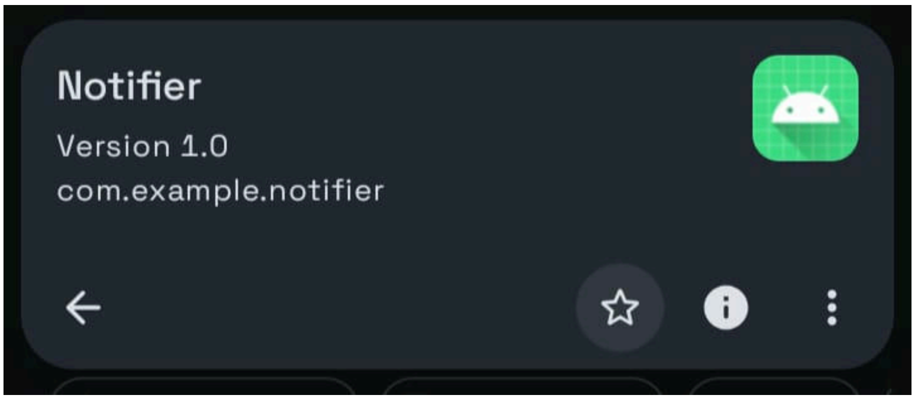
```toml
[versions]
agp = "8.9.1"
kotlin = "2.0.21"
coreKtx = "1.10.1"
junit = "4.13.2"
junitVersion = "1.1.5"
```

```
espressoCore = "3.5.1"
lifecycleRuntimeKtx = "2.6.1"
activityCompose = "1.8.0"
composeBom = "2024.09.00"

[libraries]
androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref
= "coreKtx" }
junit = { group = "junit", name = "junit", version.ref = "junit" }
androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref =
"junitVersion" }
androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso-
core", version.ref = "espressoCore" }
androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name =
"lifecycle-runtime-ktx", version.ref = "lifecycleRuntimeKtx" }
androidx-activity-compose = { group = "androidx.activity", name = "activity-
compose", version.ref = "activityCompose" }
androidx-compose-bom = { group = "androidx.compose", name = "compose-bom",
version.ref = "composeBom" }
androidx-ui = { group = "androidx.compose.ui", name = "ui" }
androidx-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" }
androidx-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" }
androidx-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui-
tooling-preview" }
androidx-ui-test-manifest = { group = "androidx.compose.ui", name = "ui-test-
manifest" }
androidx-ui-test-junit4 = { group = "androidx.compose.ui", name = "ui-test-
junit4" }
androidx-material3 = { group = "androidx.compose.material3", name =
"material3" }

[plugins]
android-application = { id = "com.android.application", version.ref = "agp" }
kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref =
"kotlin" }
kotlin-compose = { id = "org.jetbrains.kotlin.plugin.compose", version.ref =
"kotlin" }
```

```
//build.gradle.kts
```

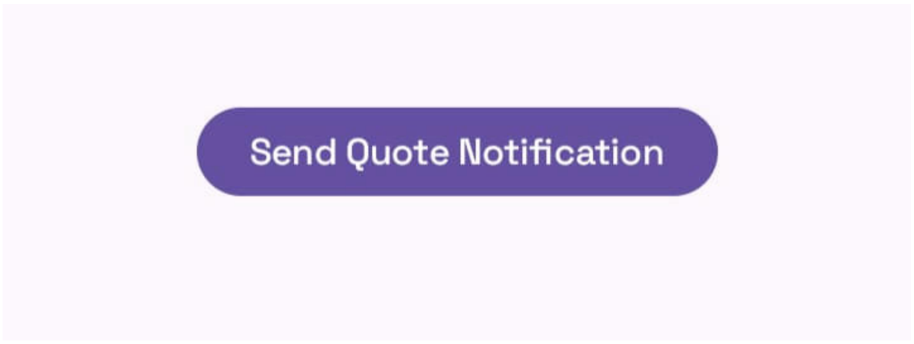
```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose)
}

android {
    namespace = "com.example.notifier"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.notifier"
        minSdk = 24
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = "11"
    }
    buildFeatures {
        compose = true
    }
}

dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform(libs.androidx.compose.bom))
    implementation(libs.androidx.ui)
    implementation(libs.androidx.ui.graphics)
    implementation(libs.androidx.ui.tooling.preview)
    implementation(libs.androidx.material3)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)
}
```

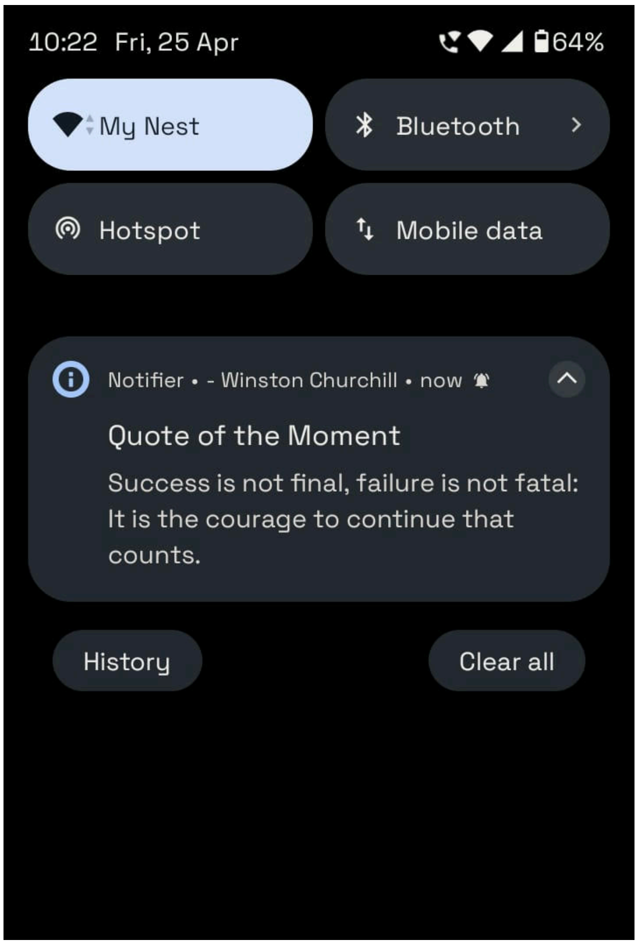
OUTPUT:


Notifier

Version 1.0

com.example.notifier

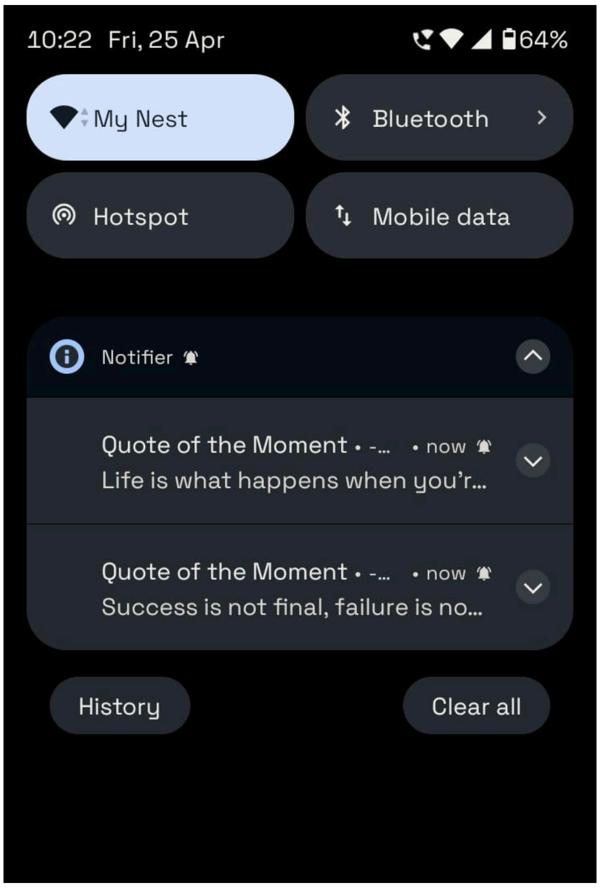



10:22 Fri, 25 Apr 64%

Notifier • - Winston Churchill • now

Quote of the Moment

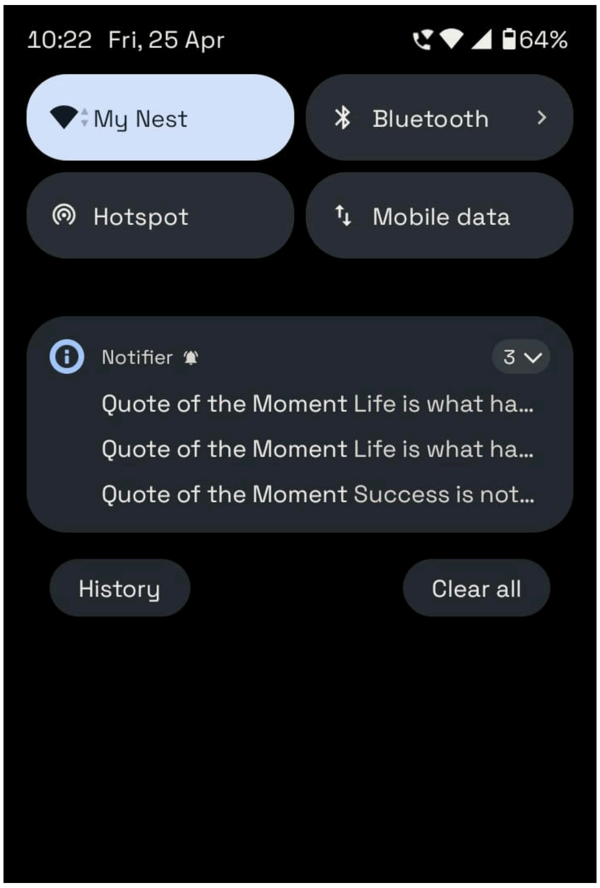
Success is not final, failure is not fatal: It is the courage to continue that counts.


10:22 Fri, 25 Apr 64%

Notifier

Quote of the Moment • -... • now
Life is what happens when you'r...

Quote of the Moment • -... • now
Success is not final, failure is no...


10:22 Fri, 25 Apr 64%

Notifier 3

Quote of the Moment Life is what ha...

Quote of the Moment Life is what ha...

Quote of the Moment Success is not...