WEEKLY EXERCISES 1

Exercise 1 – Application Development in General

1. What is an application?

What is a software application?
A software application is a computer program designed to help users perform specific tasks or activities in an easy and efficient way.

Give two examples of applications you use regularly.

- Spotify
- Google Maps

What problem does each application solve?

- Spotify allows users to stream music and podcasts anytime without downloading individual files.
- Google Maps helps users find locations, plan routes, and navigate using GPS.

2. Web App vs. Mobile App

Aspect	Web Application	Mobile Application
Where does it run?	In a web browser	On a mobile operating system
Installation needed?	No	Yes, via app store
Typical devices	Desktop computers, laptops	Smartphones, tablets
Access to device features	Limited access	Full access (GPS, camera, sensors)
User experience (UX)	Mouse/keyboard based	Touch-optimized, smoother UX

3. Web App vs. Mobile App – Use Cases

1. A restaurant menu that customers scan with a QR code
   Web app
   Users can instantly open the menu in a browser without installing anything.

2. A fitness tracking app that counts steps and uses GPS
   Mobile app
   A mobile app is required to access sensors like GPS and step counters.

3. An internal company tool used only on desktop computers
   Web app
   A web app is easy to deploy and maintain across company computers.

4. A chat application with push notifications
   Both (Web + Mobile)
   Mobile apps provide push notifications, while a web version allows desktop usage.

4. Reflection

Biggest advantage of mobile apps compared to web apps

Mobile apps can use device hardware such as GPS, camera, and notifications, giving a better user experience.

Biggest limitation of mobile apps

They must be installed and usually require separate development for different platforms.

Exercise 2 – Overview of Mobile App Development Frameworks

Technology and Framework Alternatives

When developing apps for **Android and iOS**, the main options are:

- **Native development**
- **Cross-platform development**

Native Development

Native development means building apps specifically for one platform using its official tools.

- **Android:** Kotlin or Java using Android Studio
- **iOS:** Swift using Xcode

Pros: Best performance, full access to device features
Cons: Separate codebases for Android and iOS

Cross-Platform Development

Cross-platform development allows one codebase to run on multiple platforms.

Popular options in 2026:

- **Flutter** (Dart)
- **React Native** (JavaScript / TypeScript)
- **.NET MAUI** (C#)

Pros: Faster development, shared code
Cons: Slightly less native performance in some cases

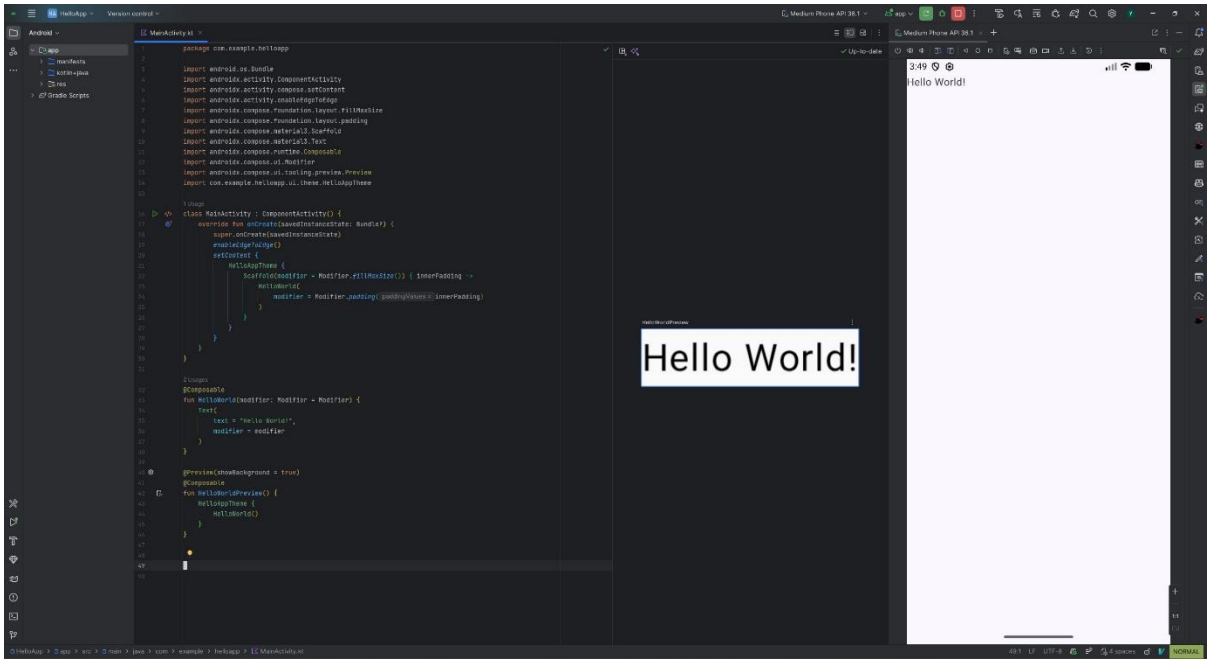
Programming Languages, Tools & Popularity

- **Kotlin**: Very popular and official for Android
- **Swift**: Official and dominant for iOS

- **JavaScript / TypeScript**: Very popular due to React Native
- **Dart**: Popular because of Flutter
  Overall, cross-platform frameworks are growing, but native development is still important for high-performance apps.

Exercise 3 – Android Jetpack Compose: Setup & Hello World

Screenshot



Code

package com.example.helloapp

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.activity.enableEdgeToEdge

import androidx.compose.foundation.layout.fillMaxSize

import androidx.compose.foundation.layout.padding

import androidx.compose.material3.Scaffold

import androidx.compose.material3.Text

import androidx.compose.runtime.Composable

import androidx.compose.ui.Modifier

import androidx.compose.ui.tooling.preview.Preview

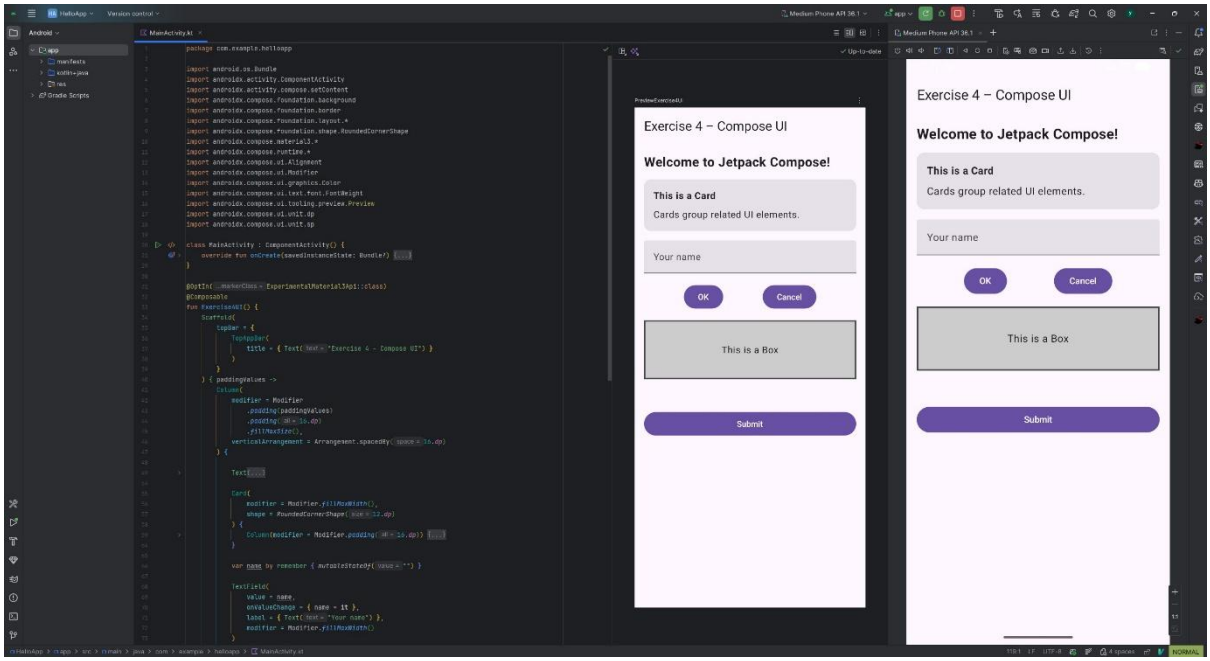
```
import com.example.helloapp.ui.theme.HelloAppTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            HelloAppTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    HelloWorld(
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
}

@Composable
fun HelloWorld(modifier: Modifier = Modifier) {
    Text(
        text = "Hello World!",
        modifier = modifier
    )
}
@Preview(showBackground = true)
@Composable
fun HelloWorldPreview() {
    HelloAppTheme {
        HelloWorld()
    }
}
```

Key Terms

- **Activity:** Entry point of the app that hosts the UI
- **Composable:** A function that describes part of the UI
- **Modifier:** Used to adjust layout, spacing, size, and appearance

Exercise 4 – Basic Composables & UI Experimentation

Screenshot



Code

```
package com.example.helloapp


import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.*
```

```
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MaterialTheme {
                Exercise4UI()
            }
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun Exercise4UI() {
    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Exercise 4 – Compose UI") }
            )
        }
    ) { paddingValues ->
        Column(
            modifier = Modifier
                .padding(paddingValues)
```

```
        .padding(16.dp)
        .fillMaxSize(),
    verticalArrangement = Arrangement.spacedBy(16.dp)
) {

    Text(
        text = "Welcome to Jetpack Compose!",
        fontSize = 22.sp,
        fontWeight = FontWeight.Bold
    )

    Card(
        modifier = Modifier.fillMaxWidth(),
        shape = RoundedCornerShape(12.dp)
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Text("This is a Card", fontWeight = FontWeight.Bold)
            Spacer(modifier = Modifier.height(8.dp))
            Text("Cards group related UI elements.")
        }
    }

    var name by remember { mutableStateOf("") }

    TextField(
        value = name,
        onValueChange = { name = it },
        label = { Text("Your name") },
        modifier = Modifier.fillMaxWidth()
    )
```

```
    Row(
        modifier = Modifier.fillMaxWidth(),
        horizontalArrangement = Arrangement.SpaceEvenly
    ) {
        Button(onClick = {}) {
            Text("OK")
        }
        Button(onClick = {}) {
            Text("Cancel")
        }
    }

    Box(
        modifier = Modifier
            .fillMaxWidth()
            .height(100.dp)
            .background(Color.LightGray)
            .border(2.dp, Color.DarkGray),
        contentAlignment = Alignment.Center
    ) {
        Text("This is a Box")
    }

    Spacer(modifier = Modifier.height(20.dp))

    Button(
        onClick = {},
        modifier = Modifier.fillMaxWidth()
    ) {
        Text("Submit")
    }
}
```

```
    }
}


@Preview(showBackground = true)
@Composable
fun PreviewExercise4UI() {
    MaterialTheme {
        Exercise4UI()
    }
}
```

My thoughts:
Jetpack Compose feels modern and intuitive. UI code is easy to read, and modifiers give precise control over layout and styling.

Conclusion

This exercise helped me understand the differences between web and mobile apps and introduced me to modern Android development. Jetpack Compose feels powerful and developer-friendly, and I'm confident it will make UI development faster and more enjoyable.