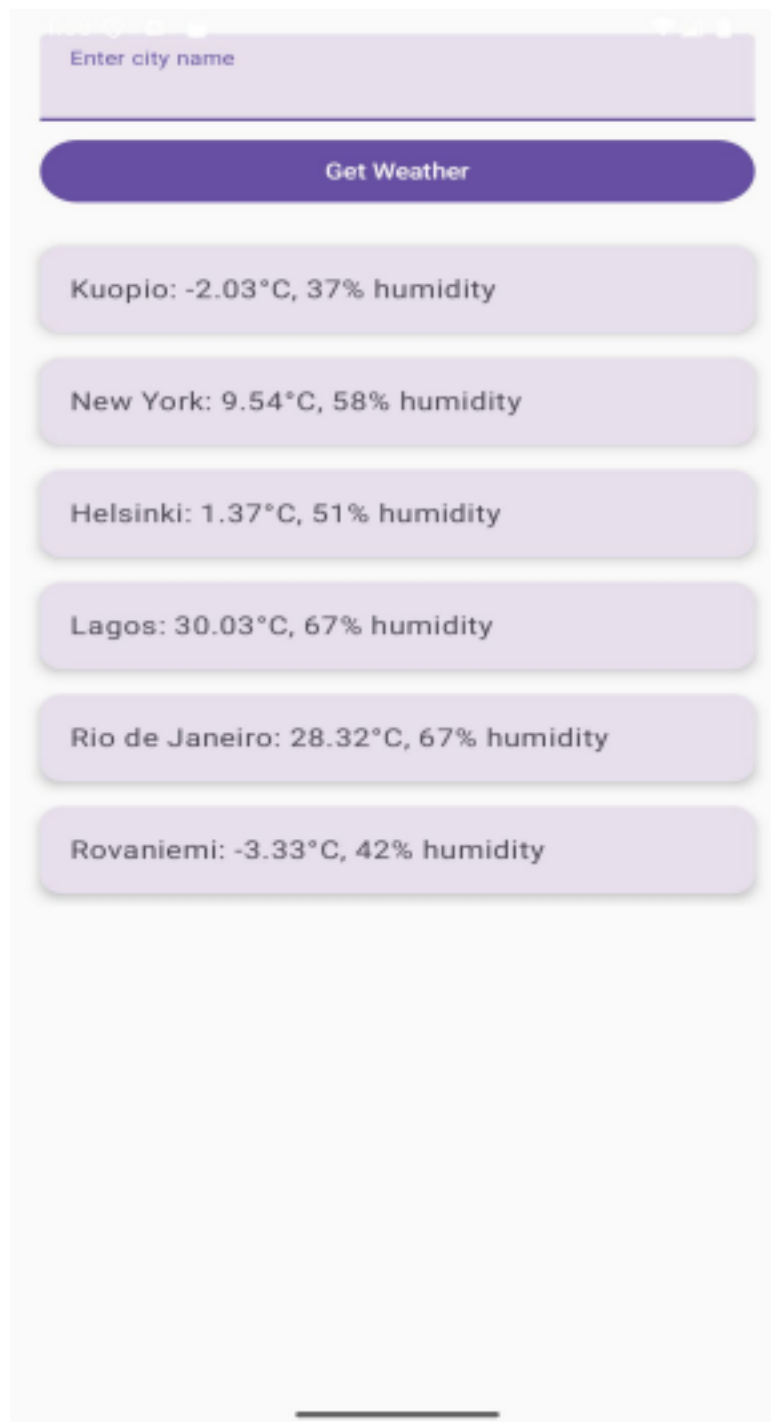


# Mobile Technology and Programming

## Assignment L8\_EX1

### Output



### Code

```
package com.example.l8_test
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
```

```

import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn import
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewmodel.compose.viewModel import
import androidx.lifecycle.viewmodelScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory import
import retrofit2.http.GET
import retrofit2.http.Query
// 1. Data Model
data class WeatherResponse(
    val name: String,
    val main: Main
)
data class Main(
    val temp: Float,
    val humidity: Int
)
// 2. Retrofit API Interface
interface WeatherApiService {
    @GET("weather")
    suspend fun getWeatherByCity(
        @Query("q") city: String,
        @Query("appid") apiKey: String,
        @Query("units") units: String = "metric"
    ): WeatherResponse
}
// 3. Retrofit Instance
object RetrofitInstance {
    private const val BASE_URL = "https://api.openweathermap.org/data/2.5/" val api:
    WeatherApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create()) .build()
            .create(WeatherApiService::class.java)
    }
}
// 4. ViewModel
class WeatherViewModel : ViewModel() {
    private val _weatherList = mutableListOf<String>() val weatherList:
    List<String> get() = _weatherList private val apiKey =
    "070121aebeedc093dda550c513215a89" fun fetchWeather(city: String) {
        viewModelScope.launch(Dispatchers.IO) {
            try {
                val trimmedCity = city.trim()

```

```

Log.d("FETCH_WEATHER", "City passed to API: '$trimmedCity'") val response =
RetrofitInstance.api.getWeatherByCity( city = trimmedCity,
apiKey = apiKey,
units = "metric"
)
val weatherText = "${response.name}: ${response.main.temp}°C,
${response.main.humidity}% humidity"
withContext(Dispatchers.Main) {
    weatherList.add(weatherText)
}
} catch (e: retrofit2.HttpException) {
withContext(Dispatchers.Main) {
    weatherList.add("HTTP ${e.code()}: ${e.message()}")
} catch (e: Exception) {
withContext(Dispatchers.Main) {
    weatherList.add("Failed to fetch weather for \"$city\":
${e.localizedMessage}")
}
}
}
}
}

// 5. Main Activity
class MainActivity : ComponentActivity() { override fun
onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContent {
WeatherScreen()
}
}

// 6. Jetpack Compose UI
@Composable
fun WeatherScreen(viewModel: WeatherViewModel = viewModel()) { var cityInput by
remember { mutableStateOf("") } Column(
modifier = Modifier
.fillMaxSize()
.padding(16.dp)
) {
TextField(
value = cityInput,
onValueChange = { cityInput = it },
label = { Text("Enter city name") },
singleLine = true,
modifier = Modifier.fillMaxWidth()
)
Spacer(modifier = Modifier.height(8.dp))
Button(
onClick = {
if (cityInput.isNotBlank()) {
viewModel.fetchWeather(cityInput.trim())
cityInput = ""
}
},

```

```
modifier = Modifier.fillMaxWidth()
) {
Text("Get Weather")
}
Spacer(modifier = Modifier.height(16.dp))
LazyColumn {
items(viewModel.weatherList) { weatherInfo -> Card(
modifier = Modifier
.fillMaxWidth()
.padding(vertical = 8.dp),
elevation = CardDefaults.cardElevation(defaultElevation = 4.dp) ) {
Text(
text = weatherInfo,
modifier = Modifier.padding(16.dp),
style = MaterialTheme.typography.bodyLarge
)
}
}
}
}
```