5COSC023W - Tutorial 10 Exercises - Sample Solutions

The Cocktails App - Another Web Services Example

Make sure that you include the Internet permission in the manifest file.

File `MainActivity.kt`:

```
package uk.ac.westminster.cocktailscomposableapp

import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.material3.Button
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.rememberCoroutineScope
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.RectangleShape
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
```

```
import org.json.JSONObject
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.URL

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

@Preview
@Composable
fun GUI() {
    var ingredient by remember { mutableStateOf("") }
    var cocktail_name by remember { mutableStateOf("") }
    var coctails_list: MutableList<String> by remember { mutableStateOf(mutableListOf()) }

    val scope = rememberCoroutineScope()
    val context = LocalContext.current

    Column(
        modifier = Modifier
            .fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        TextField(label = { Text(text = "Ingredient name") },
            value = ingredient, onValueChange = { ingredient = it })

        Button(
            shape = RectangleShape,
            onClick = {
                // get the cocktail names which contain this ingredient
                // in a new coroutine
                scope.launch {
                    coctails_list = getNames(ingredient)
                }
            }) {
            Text(text = "List Cocktails")
        }

        LazyColumn(
            modifier = Modifier
                        .height(500.dp)
                        .padding(5.dp)
        )
```

```
        {
            for (i in coctails_list) {
                item {
                    Row(modifier = Modifier
                        .fillMaxWidth()
                        .clickable(onClick = {
                            cocktail_name = i
                        }))
                    {
                        Text(text = i)
                    }
                }
            }
        }

        TextField(label = { Text(text = "Cocktail name") },
            value = cocktail_name, onValueChange = { cocktail_name = it })

        Button(
            shape = RectangleShape,
            onClick = {
                if (cocktail_name.trim() != "")
                    getRecipes(cocktail_name, context)
                else
                Toast.makeText(context, "Requested cocktail name cannot be blank",
                               Toast.LENGTH_SHORT).show()
            }) {
            Text(text = "Get Recipe")
        }
    }
}


suspend fun getNames(ingredient_requested: String): MutableList<String> {
    val stb = StringBuilder("") // contains all json
    val cocktails_list = mutableListOf<String>()

    // run this in a new thread
    withContext(Dispatchers.IO) {
        val url = URL(
            "https://www.thecocktaildb.com/api/json/v1/1/filter.php?i=" +
                    ingredient_requested.trim()
        )
        val con = url.openConnection()
        val bf = BufferedReader(InputStreamReader(con.getInputStream()))

        // read all lines JSON info  in a stringbuilder
        var line = bf.readLine()
        while (line != null) {
```

```
            stb.append(line)
            line = bf.readLine()
        }

        // now do the JSON parsing
        if (stb.toString() == "")
            return@withContext

        val json = JSONObject(stb.toString())
        val jsonArray = json.getJSONArray("drinks")

        // find the cocktail names entries and put them in the list
        for (i in 0..jsonArray.length() - 1) {
            val json_drink = jsonArray[i] as JSONObject
            val cocktail_name = json_drink["strDrink"] as String

            cocktails_list.add(cocktail_name)
        }
    }

    return cocktails_list
}


fun getRecipes(cocktail_name: String, context: Context) {
    val i = Intent(context, MainActivity2::class.java)
    i.putExtra("name", cocktail_name.trim())
    context.startActivity(i)
}
```

File `MainActivity2.kt`:

```
package uk.ac.westminster.cocktailscomposableapp

import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
```

```
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.ImageBitmap
import androidx.compose.ui.graphics.asImageBitmap
import androidx.compose.ui.graphics.painter.BitmapPainter
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.withContext
import org.json.JSONArray
import org.json.JSONObject
import java.io.BufferedInputStream
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL


// pointing to the url containing the cocktail photo to be displayed
var picture_url: String? = null

class MainActivity2 : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        val cocktail_name = intent.getStringExtra("name")
        setContent {
            displayRecipePhoto(cocktail_name)
        }
    }
}


// display the cocktail recipe and its photo
@Composable
fun displayRecipePhoto(cocktail_name: String?) {
    var recipe by remember{mutableStateOf( "")} // contains the recipe extracted from JSON
    // the bitmap containing the photo of the cocktail
    var cocktail_picture: ImageBitmap? by remember{ mutableStateOf(null) }

    val context = LocalContext.current

    LaunchedEffect(cocktail_name) {
        withContext(Dispatchers.IO) {
            picture_url = null
```

```
        val url = URL(
            "https://www.thecocktaildb.com/api/json/v1/1/search.php?s=" +
                    cocktail_name?.trim()
        )
        val con = url.openConnection()
        val bf = BufferedReader(InputStreamReader(con.getInputStream()))

        var stb = StringBuilder("")  // contains all JSON

        // read all lines JSON info  in the stb stringbuilder
        var line = bf.readLine()
        while (line != null) {
            stb.append(line + "\n")
            line = bf.readLine()
        }

        /* do the JSON parsing */
        val json = JSONObject(stb.toString())

        //val jsonArray: JSONArray = json.getJSONArray("drinks")
        var jsonArray = json["drinks"]

        if (jsonArray is JSONArray) {
            jsonArray = jsonArray as JSONArray

            // find the matching cocktail name entry and extract recipe
            for (i in 0..jsonArray.length() - 1) {
                val json_drink = jsonArray.getJSONObject(i)
                val drinkName = json_drink.getString("strDrink")
                if (drinkName.lowercase() == cocktail_name?.lowercase()) {
                    recipe = json_drink.getString("strInstructions")
                    picture_url = json_drink.getString("strDrinkThumb")
                }
                else
                    recipe = "Drink not found in DB!"
            }

            if (picture_url != null)
                cocktail_picture = getBitmapPicture()
        }
        else {  // deal with the case that the user requested a non-existing drink
            withContext(Dispatchers.Main) {
                Toast.makeText(context, "Drink not found in DB!",
                                        Toast.LENGTH_SHORT).show()
            }
            return@withContext
        }
    }
```

```
    }

    Column (Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally){
        // Cocktail title
        if (cocktail_name != null) {
            Text(text = cocktail_name, fontSize = 30.sp, fontWeight = FontWeight.Bold)

            // the recipe
            Text(text = recipe.toString(), fontSize = 24.sp)

            // the photo of the cocktail
            if (cocktail_picture != null) {
                Image(
                    modifier = Modifier.padding(50.dp),
                    painter = BitmapPainter(cocktail_picture!!),
                    contentDescription = null
                )
            }
        }
    }
}


// retrieve a bitmap image from the URL in JSON
fun getBitmapPicture(): ImageBitmap {
    var bitmap: Bitmap? = null

    val url = URL(picture_url)
    val con = url.openConnection() as HttpURLConnection
    val bfstream = BufferedInputStream(con.inputStream)

    bitmap = BitmapFactory.decodeStream(bfstream)

    return bitmap.asImageBitmap()
}
```