JSON, RETROFIT, HTTP REQUESTS AND RESTFUL APIS

JSON (JavaScript Object Notation)

Definition: Lightweight data interchange format.

Syntax:

Objects:
{ "name": "John Doe", "age": 30, "is_student": false }

Arrays:
[{ "name": "John" }, { "name": "Jane" }]

Values:
String, number, boolean, null, object, or array.

API (Application Programming Interface)

Definition: Set of rules for software communication.

Components:

Components:
URL paths for API requests.

Methods:
HTTP methods (GET, POST, etc.).

Requests:
Calls to API endpoints.

Responses:
Data returned by the API.

Example:

```
[
{ "id": 1, "name": "John Doe", "email":
"john@example.
com" },
{ "id": 2, "name": "Jane Smith",
"email": "jane@exampl
e.com" }
]
```

API Key:

Purpose:
Authentication, authorization, rate limiting, and tracking.

Example:
"apiKey": "your_api_key_here"

Gradle Scripts

Examples:

Project-level build.gradle:
Configurations common across modules.

Module-level build.gradle:
Module-specific configurations (e.g., application ID, SDK versions, dependencies).

CircularProgressIndicat

Usage: Indicates ongoing background tasks.

Example:

```
@Composable
fun LoadingScreen(isLoading: Boolean) {
if (isLoading) {
CircularProgressIndicator()
Text("Loading, please wait...")
} else {
Text("Content loaded!")
}
}
```

Dependencies

Library Dependencies:
Configurations common across modules.

SDK Dependencies:
AndroidX and Jetpack components.

Remote Dependencies:
Maven and JCenter repositories.

Example:

```
dependencies {
implementation
"org.jetbrains.kotlin:kotlin-stdlib:$ko
tlin_version"
implementation
'com.squareup.retrofit2:retrofit:2.9.0'
implementation
'androidx.recyclerview:recyclerview:1.
2.1'
}
```

Coroutines

Definition: Simplifies asynchronous programming.

Syntax:

```
suspend fun fetchData(): String {
delay(1000)
return "Data fetched successfully"
}
GlobalScope.launch {
val result = fetchData()
println(result)
}
```

Retrofit Builder

Setup:

```
val retrofit = Retrofit.Builder()
.baseUrl("<https://api.example.com/>")
.addConverterFactory(GsonConverterFactor
y.create())
.build()
```

Displaying Remote Images

Usage:

```
@Composable
fun DisplayRemoteImage(imageUrl: String)
{
Image(painter =
rememberAsyncImagePainter(imageUrl), c
ontentDescription = null)
}
```

With Placeholder:

```
@Composable
fun
DisplayRemoteImageWithPlaceholder(imageU
rl: String) {
val painter =
rememberAsyncImagePainter(imageUrl) {
placeholder(painterResource(id =
R.drawable.placeh
older))
}
Image(painter = painter,
contentDescription = null)
}
```

LazyVerticalGrid

Definition: Displays items in a grid format.

Example:

```
@Composable
fun SimpleGrid() {
val items = List(100) { "Item $it" }
LazyVerticalGrid(cells =
GridCells.Fixed(3)) {
items(items) { item →
Text(item)
}
}
}
```

Gradle Scripts

Definition: Type-safe HTTP client for Android.

tutorials.EU

Integration:

```
val retrofit = Retrofit.Builder()
.baseUrl("
<https://jsonplaceholder.typicode.com/>"
)
.addConverterFactory(GsonConverterFactor
y.create())
.build()
interface ApiService {

Day 9 3

@GET("users/1")
suspend fun getUser(): User
}
val apiService =
retrofit.create(ApiService::class.java)
val user = apiService.getUser()
```

@GET and HTTP GET Request

Definition: Annotation for HTTP GET requests in Retrofit.

Syntax:

```
interface ApiService {
@GET("users/1")
suspend fun getUser(): User
}
```

Try, Catch, and Finally

Definition: Handles exceptions gracefully.

Example:

```
try {
val result = 10 / 0
} catch (e: ArithmeticException) {
println("Cannot divide by zero")
} finally {
println("Execution complete")
}
```

Android Manifest and Internet Permission

Definition: Essential file containing app info and permissions.

Example:

```
<uses-permission
name="android.permission.INTERNE
T" />
```

Location: Within the <manifest> tags, but outside the <application> tags.

Example:

```
<manifest android="
<http://schemas.android.com/apk/r
es/android>"
package="com.example.myapp">
<uses-permission
name="android.permission.INTE
RNET" />
<application
icon="@mipmap/ic_launcher"
label="@string/app_name"
theme="@style/AppTheme">
<!— Other application settings and
activities —>
</application>
</manifest>
```