Assignment Module 6 – Advanced Android Development

1. Explain the structure of a REST API. What is Retrofit in Android, and how does it simplify API calls?

⭕ Structure of a REST API

A REST API (Representational State Transfer) is a way for two systems (like a client and a server) to communicate over the internet. It follows these key principles:

1. **Resources**: Everything is treated as a resource (e.g., users, products, weather data).
2. **Endpoints**: URLs are used to access resources (e.g., https://api.example.com/users).
3. **HTTP Methods**: Actions are performed using HTTP methods:
    - GET: Retrieve data.
    - POST: Create new data.
    - PUT: Update existing data.
    - DELETE: Remove data.
4. **Stateless**: Each request is independent; the server doesn't store client state.
5. **Response**: Data is usually returned in JSON or XML format.

Example:

- GET https://api.openweathermap.org/data/2.5/weather?q=London fetches weather data for London.

⭕ What is Retrofit in Android?

Retrofit is a library in Android that makes it easy to connect your app to a REST API. It simplifies the process of making network requests and handling responses.

⭘ How Retrofit Simplifies API Calls

1. Declarative Interface: You define the API endpoints using an interface, and Retrofit handles the implementation.

   ```
   @GET("weather")
   fun getWeather(@Query("q") city: String): Call<WeatherResponse>
   ```

2. Automatic Parsing: Retrofit automatically converts JSON responses into Kotlin data classes.

   ```
   data class WeatherResponse(val name: String, val main: Main)
   ```

3. Asynchronous Support: Retrofit works well with coroutines or callbacks, making it easy to perform network calls without blocking the main thread.

4. Error Handling: Retrofit provides built-in mechanisms to handle errors and failed requests.

5. Customization: You can add interceptors, custom converters (e.g., for XML), and more.

2. What are the benefits of using Firebase in Android development? Explain Firebase Authentication and how it can be integrated with an Android app.

⭕ Benefits of Using Firebase in Android Development

Firebase is a Backend-as-a-Service (BaaS) by Google that makes Android app development faster and easier. Key benefits:

1. No Server Setup – Firebase handles backend tasks (database, authentication, storage) so you don't need to write server-side code.

2. Real-time Database – Data syncs instantly across devices (great for chat apps, live updates).
3. Authentication – Easy login with Google, Facebook, email, etc.

4. Cloud Storage – Store and retrieve files (images, videos) securely.

5. Analytics & Crash Reporting – Track user behavior and fix crashes easily.

6. Free & Scalable – Works for small apps and scales for millions of users.

⭕ Firebase Authentication

Firebase Authentication provides secure user sign-in without writing complex backend code

- Supported Login Methods

  - Email & Password
  - Google Sign-In
  - Facebook, Twitter, GitHub
  - Phone Number (OTP)
  - Anonymous (temporary users)

⭕ How to Integrate Firebase Auth in Android (Simple Steps)

1. **Add Firebase to Your App**

- Go to Firebase Console → Create a project.
- Add your Android app (package name) and download google-services.json.
- Add Firebase dependencies in build.gradle:

```
implementation 'com.google.firebase:firebase-auth-ktx:22.3.1'
implementation 'com.google.android.gms:play-services-auth:20.7.0' // For Google Sign-In
```

2. **Enable Authentication Method**

- In Firebase Console → **Authentication** → **Sign-in method** → Enable (e.g., Email/Password, Google).

3. **Implement Sign-In (Example: Email/Password)**

```
// Sign Up
        Firebase.auth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    // User created!
                }
            }

// Sign In
        Firebase.auth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    // User logged in!
                }
            }
```

4. **Check Current User (Auto-Login)**

```
val currentUser = Firebase.auth.currentUser
if (currentUser != null) {
    // User is already logged in
}
```

5. **Sign Out**

```
Firebase.auth.signOut()
```

3. Explain the concept of services in Android. What are the differences between foreground and background services, and when should each be used?

⭘ Services in Android

Services are Android components that run tasks **in the background** (without a user interface). They keep working even when the app is closed.

Example Uses:

- Playing music in the background.
- Downloading files while the app is minimized.
- Syncing data with a server periodically.

⭘ Types of Services

1. **Foreground Service**

- **Visible to the user** (shows a notification).
- **High priority** (less likely to be killed by Android).
- **Used for:**
    - Music players (e.g., Spotify running in the background).
    - Ongoing tasks (e.g., fitness tracking, file downloads).

2. **Background Service**

- **Runs silently** (no notification).
- **Low priority** (Android may kill it to save battery).
- **Used for:**
    - Syncing data occasionally.
    - Logging events (e.g., sending analytics).

4. Describe the principles of Material Design. What are the key elements, and how do they improve the user experience?

Material Design is Google's design system for apps and websites. It makes apps look **clean, consistent, and intuitive** by using real-world physics (like paper and ink) in a digital space.

✪ Key Principles of Material Design

1. **Material is the Metaphor**
   - UI elements behave like real-world objects (e.g., shadows show depth, cards "stack" like paper).

2. **Bold & Intentional**
   - Bright colors, big typography, and clear layouts guide users.

3. **Motion Provides Meaning**
   - Smooth animations (like button ripples) show actions and transitions.

4. **Adaptive & Consistent**
   - Works on all devices (phone, tablet, web) with the same familiar patterns.

5. **Accessible**
   - High contrast, readable fonts, and touch-friendly sizes for all users.