Module 6

Networking and APIs

- **Structure of a REST API**

A REST API (Representational State Transfer API) follows a client-server architecture where resources are accessed via HTTP requests. It is stateless, meaning each request is independent.

- **Key Components of a REST API:**

1. Base URL – The main API address (e.g., https://api.example.com/).

2. Endpoints – Specific paths for resources (e.g., /users, /posts).

3. HTTP Methods – Used to perform operations:

- GET → Retrieve data

- POST → Create new data

- PUT → Update existing data

- DELETE → Remove data

4. Headers – Contain metadata like authentication tokens.

5. Query Parameters – Used for filtering or sorting data.

6. Request Body – Data sent with POST or PUT requests.

7. Response – The server's response, usually in JSON or XML format.

- **What is Retrofit in Android?**

Retrofit is a type-safe HTTP client for Android, developed by Square, used to simplify API calls. It handles HTTP requests, converts responses into Kotlin/Java objects, and supports asynchronous operations.

- **How Retrofit Simplifies API Calls?**

1. Easy API Definition – API calls are defined as interface methods.

2. Automatic JSON Parsing – Converts API responses into objects.

3. Asynchronous Requests – Supports background execution.

4. Error Handling – Provides clear error messages.

5. Interceptors – Handles authentication and logging efficiently.

6. Scalability – Works well with dependency injection (Dagger/Hilt).

Firebase Integration

Benefits of Using Firebase in Android Development: Firebase is a backend-as-a-service (BaaS) platform provided by Google that simplifies app development by offering various tools and services.

- Key Benefits:

1. Real-time Database – Syncs data across devices in real-time.

2. Cloud Firestore – A scalable NoSQL database for offline and online data sync.

3. Firebase Authentication – Provides secure user authentication via email, phone, Google, and more.

4. Cloud Storage – Stores user-generated content like images and videos.

5. Cloud Messaging (FCM) – Enables push notifications.

6. Crashlytics – Monitors and fixes app crashes in real-time.

7. Performance Monitoring – Analyzes app performance metrics.

8. Machine Learning Kit – Offers AI features like text recognition and face detection.

9. Analytics – Provides insights into user behavior.

- **Firebase Authentication**

Firebase Authentication is a secure, easy-to-use authentication system that allows users to log in using:

- Email & Password

- Phone Number OTP

- Google, Facebook, Twitter, GitHub, and Apple

- **Integrating Firebase Authentication in an Android App**

1. Add Firebase to Your Project: Go to Firebase Console and create a project. Register your Android app and download the google-services.json file. Add Firebase dependencies to the project.

2. Add Dependencies: Include Firebase Authentication and Google Sign-In dependencies in the project.

3. Initialize Firebase Authentication: Initialize Firebase Authentication instance in the application.

4. Implement Email & Password Authentication: Allow users to sign up and log in using email and password.

5. Implement Google Sign-In: Configure Google Sign-In and authenticate users using their Google account.

Services in Android (Theory)

In Android, a Service is a component that runs in the background to perform long-running operations without interacting with the user interface. A service can run independently of an Activity and does not have a visual interface. It can continue to run even if the user switches to another app.

- **Types of Services**

1. Foreground Services: A foreground service performs tasks that are noticeable to the user and requires an ongoing notification. It has higher priority and is less likely to be killed by the system during resource constraints.

2. Background Services: A background service runs without user interaction and does not require a notification. It operates with lower priority and can be killed by the system when resources are needed.

- **Differences Between Foreground and Background Services**

Visibility:

- Foreground Service: It requires an ongoing notification visible to the user to show that it is running.

- Background Service: It does not need to show a notification to the user and operates silently in the background.

Priority:

- Foreground Service: It has a higher priority in the system, meaning it is less likely to be interrupted or killed by the system due to low resources.

- Background Service: It has a lower priority and can be stopped by the system when resources are needed elsewhere.

Resource Management:

- Foreground Service: Uses more system resources because it must maintain visibility and notification to the user.

- Background Service: Uses fewer resources but can be killed by the system if needed to free up resources for more important tasks.

- **When to Use**

Foreground Services: Use when the task is critical and needs to run continuously, such as playing music or tracking location.

Background Services: Use for non-interactive tasks that can be interrupted, such as syncing data or downloading files.

Material Design and Animations

- **Material Design Principles**

Material Design is a design language developed by Google to create visually appealing, intuitive, and consistent user interfaces across all platforms.

- **Principles of Material Design**

1. Material Metaphor: Material Design draws inspiration from real-world materials, such as paper and ink. Elements like cards, sheets, and floating action buttons are treated as physical objects with properties like elevation, shadows, and boundaries.

2. Bold, Graphic, Intentional: Material Design emphasizes bold, vibrant colors, large typography, and clear visual elements.

3. Motion Provides Meaning: Motion is used to communicate changes in the app and guide the user's focus.

4. Authentic Motion and Transitions: Material Design uses natural motion to provide continuity and feedback.

- **Key Elements of Material Design**

1. Color: Material Design uses a vibrant color palette, making UI elements stand out.

2. Typography: Clear and readable fonts are essential. Material Design recommends the use of Roboto and Noto fonts.

3. Grid-Based Layout: Material Design relies on a responsive, grid-based layout system, ensuring that elements are arranged in a way that maintains balance.

4. Elevation: Elevation is represented by the use of shadows.

5. Components: Material Design defines standard UI components like buttons, sliders, menus, cards, and text fields.

6. Ripple Effects: Ripple effects are used for feedback when a user interacts with elements like buttons or touchable areas.

- **How Material Design Improves User Experience**

1. Consistency: Material Design provides a consistent design language across apps.

2. Intuitive Interactions: By simulating real-world interactions, Material Design makes it easier for users to understand the behavior of the interface.

3. Clear Visual Hierarchy: The use of bold colors, clear typography, and distinct component designs helps users understand the layout.

4. Responsiveness: Material Design ensures that apps look and feel great on different devices and screen sizes.