Unit 7: Advanced Android Concepts

Local database with SQLite (Establishing connection, creating database and tables, data manipulation), Introduction to API, API Types, Introduction to JSON, Retrieving contents from remote server, Sending contents to remote server, Implementing Google Maps in android application, Procedure for publishing application on Google Play Store.

#Past Questions

1. Provided that an SQLITE database named “College” with a table named Student has the following columns (Roll as Integer, Name as Text, and Address as Text). Develop an Android application to connect to the database, insert five student records, and display the student information. [10, 2020]
2. Provided that an SQLITE database named “Hospital” with a table named Doctor has the following columns (Did as Integer, DName as Text, Specialization as Text, and Experience as Real). Develop an Android application to connect to the database, insert records of doctors, and display the doctor information whose experience is less than 5.5 years. [2+4+4, 2023]
3. What are the different data types supported by SQLite? How to decode JSON data in an Android application? Explain with a suitable example. [2+3, 2023]
4. Develop an Android application to demonstrate the retrieval of content from a remote server. [10, 2021]
5. How to generate a signed APK? Write a program to locate the user’s current location (write only .java and manifest file). [3+7, 2021]

1. Local Database with SQLite

Establishing Connection and Creating Database and Tables

To use SQLite in Android, you need to create a subclass of SQLiteOpenHelper:

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class MyDatabaseHelper extends SQLiteOpenHelper {

   private static final String DATABASE_NAME = "MyDatabase.db";
   private static final int DATABASE_VERSION = 1;

   // SQL statement to create a table
   private static final String CREATE_TABLE_USERS =
      "CREATE TABLE Users (" +
      "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
      "name TEXT, " +
      "age INTEGER)";

   public MyDatabaseHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }

   @Override
```

```
    public void onCreate(SQLiteDatabase db) {
        // Create the Users table
        db.execSQL(CREATE_TABLE_USERS);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Drop older table if it exists
        db.execSQL("DROP TABLE IF EXISTS Users");
        // Create tables again
        onCreate(db);
    }
}
```

Data Manipulation (Insert, Update, Delete, Query)

- **Inserting Data:**

```
import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;

public void insertUser(String name, int age) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("name", name);
    values.put("age", age);
    db.insert("Users", null, values);
    db.close();
}
```

- **Updating Data:**

```
public void updateUser(int id, String name, int age) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("name", name);
    values.put("age", age);
    db.update("Users", values, "id = ?", new String[]{String.valueOf(id)});
    db.close();
}
```

- **Deleting Data:**

```
public void deleteUser(int id) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete("Users", "id = ?", new String[]{String.valueOf(id)});
    db.close();
}
```

- **Querying Data:**

```
import android.database.Cursor;

public Cursor getAllUsers() {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery("SELECT * FROM Users", null);
    return cursor;
}
```

2. Introduction to API

REST API Example

Let's assume you want to perform a simple GET request to fetch data from a REST API using HttpURLConnection.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class ApiExample {

    public static void main(String[] args) {
        try {
            URL url = new URL("https://jsonplaceholder.typicode.com/users");
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            int responseCode = conn.getResponseCode();

            if (responseCode == HttpURLConnection.HTTP_OK) {
                BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                String inputLine;
                StringBuffer response = new StringBuffer();

                while ((inputLine = in.readLine()) != null) {
                    response.append(inputLine);
                }
                in.close();

                // Print the response
                System.out.println(response.toString());
            } else {
                System.out.println("GET request failed");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3. Introduction to JSON

Parsing JSON in Android

You can parse JSON data using the JSONObject and JSONArray classes. Below is an example of how to parse a JSON string:

```
import org.json.JSONArray;
import org.json.JSONObject;

public class JsonExample {

   public static void main(String[] args) {
      String jsonString = "[{\"name\":\"John Doe\",\"age\":30},{\"name\":\"Jane Doe\",\"age\":25}]";

      try {
         JSONArray jsonArray = new JSONArray(jsonString);

         for (int i = 0; i < jsonArray.length(); i++) {
            JSONObject jsonObject = jsonArray.getJSONObject(i);
            String name = jsonObject.getString("name");
            int age = jsonObject.getInt("age");

            System.out.println("Name: " + name + ", Age: " + age);
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

4. Retrieving Contents from a Remote Server

Using HttpURLConnection to perform a GET request and retrieve JSON data:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class FetchDataExample {

   public static void main(String[] args) {
      try {
         URL url = new URL("https://jsonplaceholder.typicode.com/posts");
         HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
         urlConnection.setRequestMethod("GET");
```

```
        BufferedReader in = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
        String inputLine;
        StringBuffer content = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
            content.append(inputLine);
        }

        in.close();
        urlConnection.disconnect();

        // Print out the content
        System.out.println(content.toString());

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

5. Sending Contents to a Remote Server

Using HttpURLConnection to perform a POST request:

```
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class SendDataExample {

  public static void main(String[] args) {
    try {
        URL url = new URL("https://jsonplaceholder.typicode.com/posts");
        HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
        urlConnection.setRequestMethod("POST");
        urlConnection.setRequestProperty("Content-Type", "application/json");
        urlConnection.setDoOutput(true);

        String jsonInputString = "{\"title\":\"foo\",\"body\":\"bar\",\"userId\":1}";

        try(OutputStream os = urlConnection.getOutputStream()) {
            byte[] input = jsonInputString.getBytes("utf-8");
            os.write(input, 0, input.length);
        }

        int responseCode = urlConnection.getResponseCode();
        System.out.println("POST Response Code :: " + responseCode);
```

```
            urlConnection.disconnect();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

6. Implementing Google Maps in an Android Application

First, you need to add the necessary dependencies in your build.gradle file:

```
dependencies {
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
}
```

Next, add a MapFragment in your XML layout:

```
<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

In your activity, implement the OnMapReadyCallback interface:

```
import android.os.Bundle;
import androidx.fragment.app.FragmentActivity;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);

        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        LatLng sydney = new LatLng(-34, 151);
```

```
        googleMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
        googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```

7. Procedure for Publishing an Application on Google Play Store

Steps to Publish an Android App

1. **Create a Developer Account:**
    - Go to the Google Play Console and sign up as a developer. There is a one-time registration fee.
2. **Prepare Your App:**
    - Ensure that your app is signed with a release key. Use Android Studio's built-in tools to generate a signed APK/AAB.
3. **Upload the APK/AAB:**
    - In the Google Play Console, create a new application, and upload the APK or AAB file.
4. **Set Up Store Listing:**
    - Provide the necessary details about your app, such as title, description, screenshots, and promotional graphics.
5. **Content Rating:**
    - Complete the content rating questionnaire to get an official rating for your app.
6. **Set Pricing & Distribution:**
    - Choose whether your app will be free or paid, and select the countries and regions where it will be available.
7. **Review & Publish:**
    - Once everything is set up, review your app information and click "Publish" to make your app available on the Google Play Store.