SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

MOBILE COMPUTING LABORATORY

Course Code: ICT402

Semester: VII

Lab Manual

2025-26

SHANMUGHA ARTS, SCIENCE, TECHNOLOGY AND RESEARCH ACADEMY
(SASTRA Deemed to be) University
Tirumalaisamudram, Thanjavur-613 401
School of Computing

Course Objective:

The course will help the learner to simulate mobile ad hoc networks and Dynamic source routing protocols. It will help the user to design UI elements, event handling, database connectivity, OpenGL, widgets and notifications.

Course Learning Outcomes:

Upon successful completion of each unit, the learner will be able to

- Simulation of mobile ad hoc networks, Dynamic Source Routing protocols with the required parameters customized by the user
- Demonstrate various layouts in Android, event handling of UI, exploit the features of intents and handling of menus
- Develop an app to read the SMS, to read and display phone contacts
- To demonstrate the database connections of SQLite, SQL Server, Firebase
- Exploit the OpenGL library in android app to display 3D graphics, Location information
- To leverage the uses of accelerometer, gyro meter and use other environmental sensors in custom Android apps

List of Experiments:

1. Simulation of mobile ad hoc networks
2. Simulation of Dynamic Source routing protocols
3. Android application development using Intents
4. Adding menu and settings to an Android application
5. Android application to access the phone contents and contacts
6. Android application with data base connectivity (Internal Database)
7. Android application with data base connectivity (External Database)
8. Android application development using OpenGL
9. Application development using Widgets
10. Firebase Push notification with Android
11. Adding location, maps and contextual awareness to Android apps use of environmental sensors in Android apps

1. Simulation of Mobile Ad Hoc Networks (MANETs)

🎯 Aim

To simulate a **Mobile Ad Hoc Network (MANET)** using **NETSIM** and analyze key parameters like **packet delivery ratio**, **throughput**, and **end-to-end delay**.

🧰 Requirements

- Windows PC with **NETSIM** software installed (preferably version 13 or later)
- Basic understanding of wireless communication and routing protocols
- Optional: Spreadsheet software (Excel) for result visualization

---

📋 Theory

🔸 What is a MANET?

A **Mobile Ad Hoc Network** (MANET) is a self-configuring network of mobile nodes connected wirelessly without any fixed infrastructure.

🔸 Key Characteristics:

- Dynamic topologies
- Limited bandwidth
- Multi-hop communication
- Energy-constrained operation

🔸 Common Protocols in NETSIM for MANET:

- **AODV (Ad hoc On-demand Distance Vector)**
- **DSDV (Destination-Sequenced Distance-Vector)**
- **OLSR (Optimized Link State Routing)**

🛠️ Simulation Procedure

🔹 Step 1: Launch NETSIM

1. Open **NETSIM**.
2. Click on **"New Simulation"**.
3. Choose **"MANET"** as the simulation type and click **OK**.

🔹 Step 2: Set Up the Network Topology

1. In the topology design area, **drag and drop 5 to 10 mobile nodes**.
2. Select **Mobile Nodes** (they can move randomly based on mobility model).
3. Define node mobility by choosing a **mobility model**:
    - Random Waypoint
    - Gauss-Markov (optional)
4. Add a **Traffic Generator** (e.g., CBR or FTP):
    - Source: Node 1
    - Destination: Node 5
    - Packet Size: 512 bytes
    - Packet Rate: 100 packets/second

🔹 Step 3: Choose Routing Protocol

1. Click on any node → go to the **Routing** tab.

2. Select **AODV**, **DSDV**, or **OLSR** from the dropdown.
3. Apply the same routing protocol to all nodes.

🔹 **Step 4: Configure Simulation Parameters**

- Simulation Time: **100 seconds**
- Propagation Model: **Two-ray Ground or Free Space**
- Transmission Range: e.g., 250 meters
- Mobility: Enabled with speed range (e.g., 1–10 m/s)

🔹 **Step 5: Run the Simulation**

1. Click on **Run Simulation**.
2. Observe the **Packet flow animation**.
3. After simulation, NETSIM will automatically open the **Results Dashboard**.

📈 **Analysis of Results**

Key metrics from the result dashboard:

Metric	Description
Throughput	Data delivered over time (Kbps or Mbps)
End-to-End Delay	Average delay from source to destination (ms)
Packet Delivery Ratio	Ratio of delivered packets to sent packets (%)
Routing Overhead	Control packet usage (e.g., for AODV)

You can also **export graphs** and tables for further analysis in Excel.

🖼️ **Expected Output**

- Graphs of **Throughput vs. Time**
- Comparison of **PDR (Packet Delivery Ratio)** for different protocols
- Log file showing packet trace and protocol behavior

✅ **Viva Questions**

1. What is the significance of mobility in MANETs?
2. How does AODV differ from DSDV?
3. What are the advantages of reactive routing protocols?
4. Explain the impact of node speed on packet delivery.
5. What is the routing overhead in MANETs?

---

2. Simulation of Dynamic Source Routing (DSR) Protocol

🎯 Aim

To simulate a **Mobile Ad Hoc Network (MANET)** using the **DSR (Dynamic Source Routing)** protocol in **NETSIM**, and analyze network performance in terms of **packet delivery ratio (PDR)**, **end-to-end delay**, and **throughput**.

🧰 Requirements

- Windows PC with **NETSIM** installed (preferably v13+)
- Basic knowledge of MANETs and routing protocols
- Optional: Spreadsheet software for analysis (Excel, Google Sheets)

📖 Theory

🔹 What is DSR?

Dynamic Source Routing (DSR) is a **reactive routing protocol** used in MANETs. It dynamically discovers a source route across multiple nodes to reach a destination, using two main phases:

- **Route Discovery**
- **Route Maintenance**

DSR embeds the full path to the destination in the packet header.

📋 Procedure

🔸 Step 1: Launch NETSIM and Start a New MANET Simulation

1. Open **NETSIM**.
2. Click **New Simulation** → Select **MANET** → Click **OK**.

🔸 Step 2: Create the Network Topology

1. **Drag and drop 10 mobile nodes** from the left sidebar into the simulation area.
2. Select each node → Set mobility model:
    - **Random Waypoint**
    - Speed: 2–10 m/s
    - Pause Time: 10 s (optional)
3. Place nodes in such a way that they can connect within a **250-meter range**.

🔸 Step 3: Configure DSR Protocol

1. Click each node → Go to **Routing** tab.
2. Set the **Routing Protocol** to **DSR**.
3. Apply to all nodes.

🔸 Step 4: Add Application Traffic

1. Go to the **Application Layer** panel.
2. Add an **application** (e.g., **CBR** or **FTP**):
    - Source: Node 1
    - Destination: Node 8
    - Packet size: 512 bytes

- Packet rate: 100 packets/sec
- Start time: 1 sec
- End time: 100 sec

🔸 **Step 5: Configure Simulation Parameters**

Parameter	Value
Simulation time	100 seconds
Propagation model	Two-ray Ground
Transmission range	250 meters
Mobility enabled	Yes

🔸 **Step 6: Run the Simulation**

1. Click **Run Simulation**.
2. Wait for simulation to complete.
3. NETSIM will auto-open the **Results Dashboard**.

📊 **Post-Simulation Analysis**

After simulation, analyze the following key metrics:

Metric	Description
Packet Delivery Ratio	(Packets Received / Packets Sent) × 100
End-to-End Delay	Avg. time for a packet to reach destination (ms)
Throughput	Successful delivery rate (Kbps or Mbps)
Routing Overhead	Ratio of control packets to data packets

To view:

- Open **Performance Metrics**
- View **Routing Statistics**, **Packet Flow Animation**, and **Application Metrics**

You can export results as .csv for further analysis in Excel.

🖼️ **Expected Output**

- Graph showing **PDR vs Time**
- Table of routing protocol overhead (AODV vs DSR if compared)
- Animation of mobile nodes establishing and maintaining routes

---

🧠 **Viva Questions**

1. What type of routing protocol is DSR?
2. How does DSR differ from AODV?
3. What are the key phases of DSR?
4. How is route maintenance handled in DSR?
5. What are the advantages and disadvantages of source routing?

---

3. Android applications development using explicit Intent

Aim
To develop an Android application with a single activity that uses explicit intents to perform:

- Opening a URL
- Opening a location in Google Maps
- Opening the dial pad
- Sending an SMS via the default messaging app

🧰 Requirements

- Android Studio
- Android Emulator or physical device
- Android SDK (API 21+)
- Internet connection (for browser & maps)

📁 Project Structure

📂 com.example.explicitintentslab/

- MainActivity.java (only one activity)

🛠️ Step-by-Step Implementation

✅ Step 1: Create New Android Project

- Open Android Studio → New Project → Empty Activity
- Project name: ExplicitIntentsLab
- Language: Java
- Minimum SDK: API 21 or above

✅ Step 2: Design Layout with 4 Buttons

res/layout/activity_main.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp">

    <Button
        android:id="@+id/btnOpenUrl"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Open URL" />

    <Button
        android:id="@+id/btnOpenMap"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Open Location in Google Maps" />
```

```
    <Button
        android:id="@+id/btnDialNumber"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Open Dial Pad" />

    <Button
        android:id="@+id/btnSendSms"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Send SMS" />
</LinearLayout>
```

✅ Step 3: Add Functionality to Buttons

MainActivity.java

```
package com.example.explicitintentslab;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    Button btnOpenUrl, btnOpenMap, btnDialNumber, btnSendSms;

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

        btnOpenUrl = findViewById(R.id.btnOpenUrl);
        btnOpenMap = findViewById(R.id.btnOpenMap);
        btnDialNumber = findViewById(R.id.btnDialNumber);
        btnSendSms = findViewById(R.id.btnSendSms);

        // Open URL
        btnOpenUrl.setOnClickListener(v -> {
            Intent intent = new Intent(Intent.ACTION_VIEW);
            intent.setData(Uri.parse("https://www.google.com"));
            startActivity(intent);
        });

        // Open Location in Google Maps
        btnOpenMap.setOnClickListener(v -> {
            Uri mapUri = Uri.parse("geo:0,0?q=India+Gate,Delhi");
            Intent mapIntent = new Intent(Intent.ACTION_VIEW, mapUri);
            mapIntent.setPackage("com.google.android.apps.maps");

            if (mapIntent.resolveActivity(getPackageManager()) != null) {
                startActivity(mapIntent);
```

```
        }
    });


    // Open Dial Pad
    btnDialNumber.setOnClickListener(v -> {
        Intent dialIntent = new Intent(Intent.ACTION_DIAL);
        dialIntent.setData(Uri.parse("tel:1234567890"));
        startActivity(dialIntent);
    });

    // Send SMS
    btnSendSms.setOnClickListener(v -> {
        Intent smsIntent = new Intent(Intent.ACTION_VIEW);
        smsIntent.setData(Uri.parse("smsto:1234567890"));
        smsIntent.putExtra("sms_body", "Hello! This is a test message.");
        startActivity(smsIntent);
    });
  }
}
```

✅ Step 4: Manifest File

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.explicitintentslab">

    <application
        android:allowBackup="true"
        android:label="ExplicitIntentsLab"
        android:icon="@mipmap/ic_launcher"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ExplicitIntentsLab">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

📝 Output Summary

Action	Triggered by Button	Result
Open URL	btnOpenUrl	Launches browser and opens Google
Open Map Location	btnOpenMap	Opens Google Maps with location query
Dial Number	btnDialNumber	Opens dial pad with number

Action	Triggered by Button	Result
Send SMS	btnSendSms	Opens SMS app with number & message

🎓 Viva Questions

1. What is an intent in Android?
2. What is the difference between implicit and explicit intent?
3. What is the URI format for Google Maps?
4. Is permission required for opening the dialer using ACTION_DIAL?
5. How do you send a message using an intent?

---

4. Android application using implicit intent

🎯 Aim

To develop an Android app with two activities that invoke each other using implicit intents.

🧰 Requirements

- Android Studio
- Emulator or Device
- Android SDK API 21+

Procedure

Step 1: Create New Android Project

- Name: ImplicitIntentTwoWay
- Language: Java
- Minimum SDK: API 21+

Step 2: Create layouts

No special UI needed, just simple layouts for testing.

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:padding="20dp">

    <Button
        android:id="@+id/btnToSecond"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go to Second Activity" />
</LinearLayout>
```

activity_second.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:padding="20dp">

    <Button
        android:id="@+id/btnToMain"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:text="Go back to Main Activity" />
</LinearLayout>
```

Step 3: Implement MainActivity.java

```
package com.example.implicittwoway;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

   private static final String ACTION_TO_SECOND = "com.example.ACTION_TO_SECOND";

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

       Button btnToSecond = findViewById(R.id.btnToSecond);

       btnToSecond.setOnClickListener(v -> {
          Intent intent = new Intent(ACTION_TO_SECOND);
          startActivity(intent);
       });
   }
}
```

Step 4: Implement SecondActivity.java

```
package com.example.implicittwoway;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {

   private static final String ACTION_TO_MAIN = "com.example.ACTION_TO_MAIN";

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

      Button btnToMain = findViewById(R.id.btnToMain);
```

```
        btnToMain.setOnClickListener(v -> {
            Intent intent = new Intent(ACTION_TO_MAIN);
            startActivity(intent);
        });
    }
}
```

Step 5: Define intent filters in AndroidManifest.xml

```
<application ... >

    <activity android:name=".SecondActivity">
        <intent-filter>
            <action android:name="com.example.ACTION_TO_SECOND" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

        <intent-filter>
            <action android:name="com.example.ACTION_TO_MAIN" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>

</application>
```

Explanation:

- MainActivity starts SecondActivity via implicit intent with action "com.example.ACTION_TO_SECOND".
- SecondActivity starts MainActivity via implicit intent with action "com.example.ACTION_TO_MAIN".
- Both activities declare intent filters to handle those actions.

Output

Step	Result
Click "Go to Second"	Opens SecondActivity
Click "Go back to Main"	Opens MainActivity

Viva Questions

1. What is an implicit intent?
2. How does Android find the activity to launch for an implicit intent?
3. What is the role of intent filters?
4. Can multiple activities respond to the same intent? What happens?
5. Why use implicit intents for communication between app's own activities?

5. Adding menu and settings to an Android application

🎯 Aim

To create an Android app with an Options Menu containing multiple submenus and display a Toast message showing the clicked menu/submenu name.

🧰 Requirements

- Android Studio
- Basic knowledge of Android menus and Toasts
- Java or Kotlin

Procedure

Step 1: Create Menu XML with Submenus

Create a menu XML file:

res/menu/main_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/menu_file"
        android:title="File">
        <menu>
            <item android:id="@+id/menu_new" android:title="New" />
            <item android:id="@+id/menu_open" android:title="Open" />
            <item android:id="@+id/menu_exit" android:title="Exit" />
        </menu>
    </item>

    <item
        android:id="@+id/menu_edit"
        android:title="Edit">
        <menu>
            <item android:id="@+id/menu_cut" android:title="Cut" />
            <item android:id="@+id/menu_copy" android:title="Copy" />
            <item android:id="@+id/menu_paste" android:title="Paste" />
        </menu>
    </item>

    <item
        android:id="@+id/menu_help"
        android:title="Help">
        <menu>
            <item android:id="@+id/menu_about" android:title="About" />
        </menu>
    </item>

</menu>
```

Step 2: Inflate Menu and Handle Clicks in MainActivity.java

```
package com.example.menudemo;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_menu, menu);
    return true;
  }
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    // Show Toast with the clicked menu title
    Toast.makeText(this, item.getTitle(), Toast.LENGTH_SHORT).show();
    return super.onOptionsItemSelected(item);
  }}
```

Step 3: Run and Test

- Launch the app
- Tap the menu button (or toolbar menu)
- You will see File, Edit, and Help menus
- Each has submenu items
- Click any menu or submenu → a Toast appears showing that item's title

---

Output

Action	Result
Open menu	Shows File, Edit, Help menus
Click "New" submenu	Toast: "New"
Click "Paste" submenu	Toast: "Paste"
Click "About" submenu	Toast: "About"

Viva Questions

1. How do you create a submenu in Android menus?
2. What method inflates a menu resource in an activity?
3. How to handle menu item clicks?
4. How to show a Toast message?
5. Can menus have nested submenus?

6. Android Application to Access Phone Contents and Contacts

🎯 **Aim**

To develop an Android application that accesses and displays the device contacts using content providers.

🧰 **Requirements**

- Android Studio
- Android device or emulator with contacts
- Android SDK API 21+
- Permission to read contacts

Procedure

Step 1: Create a New Android Project

- Create a new project in Android Studio (Empty Activity)
- Name: ContactsAccessApp
- Language: Java
- Minimum SDK: API 21+

Step 2: Add Permission in Manifest

Open **AndroidManifest.xml** and add the permission:

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

Step 3: Design Layout to Display Contacts

Edit **res/layout/activity_main.xml** to add a ListView:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <ListView
      android:id="@+id/contactsListView"
      android:layout_width="match_parent"
      android:layout_height="match_parent"/>
</LinearLayout>
```

Step 4: Request Runtime Permission

Since Android 6.0 (API 23+), permissions must be requested at runtime.

Step 5: Implement MainActivity.java

```
package com.example.contactsaccessapp;

import android.Manifest;
import android.content.pm.PackageManager;
import android.database.Cursor;
```

```
import android.os.Bundle;
import android.provider.ContactsContract;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

   private static final int PERMISSIONS_REQUEST_READ_CONTACTS = 100;

   ListView contactsListView;
   ArrayList<String> contactsList;
   ArrayAdapter<String> adapter;

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

       contactsListView = findViewById(R.id.contactsListView);
       contactsList = new ArrayList<>();

       adapter = new ArrayAdapter<>(this,
             android.R.layout.simple_list_item_1, contactsList);

       contactsListView.setAdapter(adapter);

       // Check permission
       if (ContextCompat.checkSelfPermission(this,
             Manifest.permission.READ_CONTACTS) != PackageManager.PERMISSION_GRANTED) {
          ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.READ_CONTACTS},
                PERMISSIONS_REQUEST_READ_CONTACTS);
       } else {
          loadContacts();
       } }
 private void loadContacts() {
       Cursor cursor = getContentResolver().query(
             ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
             null, null, null, ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME + " ASC");

       if (cursor != null) {
          contactsList.clear();
          while (cursor.moveToNext()) {
```

```
            String name = cursor.getString(cursor.getColumnIndex(
                    ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME));
            String phone = cursor.getString(cursor.getColumnIndex(
                    ContactsContract.CommonDataKinds.Phone.NUMBER));

            contactsList.add(name + "\n" + phone);
        }
        cursor.close();
        adapter.notifyDataSetChanged();
    } else {
        Toast.makeText(this, "No contacts found", Toast.LENGTH_SHORT).show();
    }
}

// Handle permission result
@Override
public void onRequestPermissionsResult(int requestCode,
                                       @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == PERMISSIONS_REQUEST_READ_CONTACTS) {
        if (grantResults.length > 0 &&
            grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            loadContacts();
        } else {
            Toast.makeText(this,
                    "Permission denied to read contacts",
                    Toast.LENGTH_SHORT).show();
        }
    }
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
}}
```

Explanation:

- The app requests **READ_CONTACTS** permission at runtime.
- If granted, it queries the contacts database using ContentResolver.
- It retrieves the contact name and phone number.
- Displays the list in a ListView.

Output

Step	Result
Launch app	Requests permission if not granted
If permission granted	Displays contacts name and number
If permission denied	Shows permission denied Toast

Viva Questions

1. What is a content provider in Android?
2. How do you query contacts using content resolver?
3. Why is runtime permission needed?
4. How to handle permission request results?
5. What is the URI used to access contacts?

7. Android Application with Internal Database Connectivity

🎯 **Aim**

To develop an Android application with SQLite internal database connectivity and buttons to perform **Insert**, **Update**, **Delete**, and **View** operations on the database.

🧰 **Requirements**

- Android Studio
- Java (or Kotlin)
- Basic SQLite knowledge

Procedure

Step 1: Create a New Android Project

- Create new project with Empty Activity
- Name: InternalDBApp
- Language: Java
- Minimum SDK: API 21+

Step 2: Design the Layout

res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp">

   <EditText
      android:id="@+id/etId"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="ID (Integer)"
      android:inputType="number" />

   <EditText
      android:id="@+id/etName"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Name" />

   <EditText
      android:id="@+id/etEmail"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Email"
```

```
    android:inputType="textEmailAddress"/>

  <Button
    android:id="@+id/btnInsert"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Insert" />

  <Button
    android:id="@+id/btnUpdate"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Update" />

  <Button
    android:id="@+id/btnDelete"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Delete" />

  <Button
    android:id="@+id/btnView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="View All" />

  <TextView
    android:id="@+id/tvData"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"
    android:paddingTop="10dp" />

</LinearLayout>
```

Step 3: Create Database Helper Class

Create a new Java class named DatabaseHelper.java:

```
package com.example.internaldbapp;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {

  private static final String DATABASE_NAME = "UserDB.db";
  private static final String TABLE_NAME = "users";
  private static final int DATABASE_VERSION = 1;
```

```
private static final String COL_ID = "id";
private static final String COL_NAME = "name";
private static final String COL_EMAIL = "email";

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

// Called when DB is created
@Override
public void onCreate(SQLiteDatabase db) {
   String createTable = "CREATE TABLE " + TABLE_NAME +
         " (" + COL_ID + " INTEGER PRIMARY KEY, " +
         COL_NAME + " TEXT, " +
         COL_EMAIL + " TEXT)";
   db.execSQL(createTable);
}

// Called when DB version is updated
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
   db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
   onCreate(db);
}

// Insert record
public boolean insertData(int id, String name, String email) {
   SQLiteDatabase db = this.getWritableDatabase();
   ContentValues cv = new ContentValues();
   cv.put(COL_ID, id);
   cv.put(COL_NAME, name);
   cv.put(COL_EMAIL, email);

   long result = db.insert(TABLE_NAME, null, cv);
   return result != -1;
}

// Update record
public boolean updateData(int id, String name, String email) {
   SQLiteDatabase db = this.getWritableDatabase();
   ContentValues cv = new ContentValues();
   cv.put(COL_NAME, name);
   cv.put(COL_EMAIL, email);

   int rows = db.update(TABLE_NAME, cv, COL_ID + "=?", new String[]{String.valueOf(id)});
   return rows > 0;
}

// Delete record
public boolean deleteData(int id) {
   SQLiteDatabase db = this.getWritableDatabase();
```

```
        int rows = db.delete(TABLE_NAME, COL_ID + "=?", new String[]{String.valueOf(id)});
        return rows > 0;
    }

    // Get all records
    public Cursor getAllData() {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
    }
}
```

Step 4: Implement MainActivity

MainActivity.java

```
package com.example.internaldbapp;

import android.database.Cursor;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    EditText etId, etName, etEmail;
    Button btnInsert, btnUpdate, btnDelete, btnView;
    TextView tvData;

    DatabaseHelper dbHelper;

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

        dbHelper = new DatabaseHelper(this);

        etId = findViewById(R.id.etId);
        etName = findViewById(R.id.etName);
        etEmail = findViewById(R.id.etEmail);

        btnInsert = findViewById(R.id.btnInsert);
        btnUpdate = findViewById(R.id.btnUpdate);
        btnDelete = findViewById(R.id.btnDelete);
        btnView = findViewById(R.id.btnView);

        tvData = findViewById(R.id.tvData);
```

```
        btnInsert.setOnClickListener(v -> {
            String idStr = etId.getText().toString();
            String name = etName.getText().toString();
            String email = etEmail.getText().toString();

            if (TextUtils.isEmpty(idStr) || TextUtils.isEmpty(name) || TextUtils.isEmpty(email)) {
                Toast.makeText(this, "Please enter all fields", Toast.LENGTH_SHORT).show();
                return;
            }

            int id = Integer.parseInt(idStr);
            boolean inserted = dbHelper.insertData(id, name, email);
            if (inserted) {
                Toast.makeText(this, "Data Inserted", Toast.LENGTH_SHORT).show();
                clearFields();
            } else {
                Toast.makeText(this, "Insertion Failed (ID may already exist)",
Toast.LENGTH_SHORT).show();
            }
        });

        btnUpdate.setOnClickListener(v -> {
            String idStr = etId.getText().toString();
            String name = etName.getText().toString();
            String email = etEmail.getText().toString();

            if (TextUtils.isEmpty(idStr) || TextUtils.isEmpty(name) || TextUtils.isEmpty(email)) {
                Toast.makeText(this, "Please enter all fields", Toast.LENGTH_SHORT).show();
                return;
            }

            int id = Integer.parseInt(idStr);
            boolean updated = dbHelper.updateData(id, name, email);
            if (updated) {
                Toast.makeText(this, "Data Updated", Toast.LENGTH_SHORT).show();
                clearFields();
            } else {
                Toast.makeText(this, "Update Failed (Record not found)",
Toast.LENGTH_SHORT).show();
            }
        });

        btnDelete.setOnClickListener(v -> {
            String idStr = etId.getText().toString();

            if (TextUtils.isEmpty(idStr)) {
                Toast.makeText(this, "Please enter ID to delete", Toast.LENGTH_SHORT).show();
                return;
            }
```

```
            int id = Integer.parseInt(idStr);
            boolean deleted = dbHelper.deleteData(id);
            if (deleted) {
                Toast.makeText(this, "Data Deleted", Toast.LENGTH_SHORT).show();
                clearFields();
            } else {
                Toast.makeText(this, "Delete Failed (Record not found)",
Toast.LENGTH_SHORT).show();
            }
        });

        btnView.setOnClickListener(v -> {
            Cursor cursor = dbHelper.getAllData();
            if (cursor.getCount() == 0) {
                tvData.setText("No Data Found");
                return;
            }

            StringBuilder builder = new StringBuilder();
            while (cursor.moveToNext()) {
                builder.append("ID: ").append(cursor.getInt(0)).append("\n");
                builder.append("Name: ").append(cursor.getString(1)).append("\n");
                builder.append("Email: ").append(cursor.getString(2)).append("\n\n");
            }
            tvData.setText(builder.toString());
            cursor.close();
        });
    }

    private void clearFields() {
        etId.setText("");
        etName.setText("");
        etEmail.setText("");
    }
}
```

Explanation:

- The **DatabaseHelper** class manages the SQLite DB and CRUD operations.
- The **MainActivity** has input fields and buttons to insert, update, delete, and view data.
- Data is displayed in a TextView.
- Appropriate Toast messages are shown for success/failure.
- Input validation checks that fields are not empty.

Output

Button	Action	Result
Insert	Inserts new record	Toast: Data Inserted
Update	Updates existing record by ID	Toast: Data Updated
Delete	Deletes record by ID	Toast: Data Deleted
View	Displays all records	Shows list of all data in TextView

Viva Questions

3. What is SQLite and why use it in Android?
4. How do you create and upgrade a database?
5. What are CRUD operations?
6. How do you perform insert, update, delete, and select queries?
7. What is a ContentValues object?
8. How to handle Cursor and prevent memory leaks?

8. Android application with data base connectivity (External Database)

🎯 **Aim**

To develop an Android application using JDBC in MainActivity.java to connect to a **remote MySQL database**, performing **Insert**, **Update**, **Delete**, and **View** operations using buttons.

🧰 **Requirements**

- Android Studio
- MySQL server (remote/local with open TCP port 3306)
- MySQL JDBC driver (mysql-connector-java-x.x.x.jar)
- Internet permission
- ADB over network or emulator with host access

Step-by-Step Lab Manual

Step 1: Setup MySQL Table

```
CREATE DATABASE androiddb;
USE androiddb;

CREATE TABLE users (
   id INT PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100)
);
```

Step 2: Add Internet Permission

Add this to AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Step 3: Add MySQL JDBC Driver

1. Download the MySQL JDBC jar file: https://dev.mysql.com/downloads/connector/j/
2. Place it in the libs/ folder in your Android project.
3. In build.gradle, add:

```
dependencies {
   implementation files('libs/mysql-connector-java-8.0.xx.jar')
}
```

Step 4: Design Layout

Same as before — a form with EditTexts for ID, Name, Email, and buttons for Insert, Update, Delete, and View.

Step 5: MainActivity.java

```
package com.example.mysqldirectapp;

import android.os.AsyncTask;
import android.os.Bundle;
import android.widget.*;
```

```
import androidx.appcompat.app.AppCompatActivity;

import java.sql.*;
import java.util.Objects;

public class MainActivity extends AppCompatActivity {

    EditText etId, etName, etEmail;
    TextView tvResult;
    Button btnInsert, btnUpdate, btnDelete, btnView;

    // JDBC connection config
    static final String DB_URL = "jdbc:mysql://<YOUR_SERVER_IP>:3306/androiddb";
    static final String USER = "your_db_user";
    static final String PASS = "your_db_password";

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

        etId = findViewById(R.id.etId);
        etName = findViewById(R.id.etName);
        etEmail = findViewById(R.id.etEmail);
        tvResult = findViewById(R.id.tvData);

        btnInsert = findViewById(R.id.btnInsert);
        btnUpdate = findViewById(R.id.btnUpdate);
        btnDelete = findViewById(R.id.btnDelete);
        btnView = findViewById(R.id.btnView);

        btnInsert.setOnClickListener(v -> performDatabaseOperation("insert"));
        btnUpdate.setOnClickListener(v -> performDatabaseOperation("update"));
        btnDelete.setOnClickListener(v -> performDatabaseOperation("delete"));
        btnView.setOnClickListener(v -> performDatabaseOperation("view"));
    }

    private void performDatabaseOperation(String operation) {
        String id = etId.getText().toString();
        String name = etName.getText().toString();
        String email = etEmail.getText().toString();

        new AsyncTask<String, Void, String>() {
            @Override
            protected String doInBackground(String... params) {
                try {
                    Class.forName("com.mysql.jdbc.Driver");
                    Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
                    Statement stmt = conn.createStatement();

                    switch (operation) {
```

```
                case "insert":
                    String insertQuery = "INSERT INTO users VALUES (" + id + ", '" + name + "', '" +
email + "')";
                    stmt.executeUpdate(insertQuery);
                    return "Inserted!";
                case "update":
                    String updateQuery = "UPDATE users SET name='" + name + "', email='" + email
+ "' WHERE id=" + id;
                    stmt.executeUpdate(updateQuery);
                    return "Updated!";
                case "delete":
                    String deleteQuery = "DELETE FROM users WHERE id=" + id;
                    stmt.executeUpdate(deleteQuery);
                    return "Deleted!";
                case "view":
                    ResultSet rs = stmt.executeQuery("SELECT * FROM users");
                    StringBuilder sb = new StringBuilder();
                    while (rs.next()) {
                        sb.append("ID: ").append(rs.getInt("id")).append("\n")
                          .append("Name: ").append(rs.getString("name")).append("\n")
                          .append("Email: ").append(rs.getString("email")).append("\n\n");
                    }
                    return sb.toString();
                }
                conn.close();
            } catch (Exception e) {
                return "Error: " + e.getMessage();
            }
            return "Done.";
        }

        @Override
        protected void onPostExecute(String result) {
            if (operation.equals("view")) {
                tvResult.setText(result);
            } else {
                Toast.makeText(MainActivity.this, result, Toast.LENGTH_SHORT).show();
            }
        }
    }.execute();
  }
}
```

✅ **Output**

Button	Action	Result
Insert	Adds record to MySQL DB	Shows "Inserted!" Toast
Update	Modifies existing record	Shows "Updated!" Toast
Delete	Removes record by ID	Shows "Deleted!" Toast
View	Lists all records	Shows in TextView

🧠 Viva Questions

1. Why is JDBC not used in real Android apps for MySQL?
2. What is the purpose of AsyncTask in this project?
3. How do you handle SQL exceptions in Java?
4. How would you secure DB credentials in production apps?
5. What's the alternative to direct MySQL access from Android?

9. Android Application Using OpenGL with Simple Object Rotation

---

🎯 Aim

To develop an Android application using **OpenGL ES** to render a simple 3D object (like a triangle or cube) with continuous **rotation animation**.

🧰 Requirements

- Android Studio
- OpenGL ES 2.0
- Java / Kotlin
- Android SDK (API 21+)

📋 Procedure

🔹 Step 1: Create a New Android Project

- Project Name: OpenGLRotationApp
- Language: Java
- Minimum SDK: API 21+
- Template: Empty Activity

🔹 Step 2: Add OpenGL Support

No need to add extra dependencies — OpenGL ES is part of Android SDK.

🔹 Step 3: Project Structure

```
app/
 ├─ java/com.example.openglrotationapp/
 |    ├─ MainActivity.java
 |    ├─ MyGLSurfaceView.java
 |    └─ MyGLRenderer.java
 └─ res/layout/activity_main.xml
```

🔹 Step 4: Layout File

res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</FrameLayout>
```

🔹 Step 5: Main Activity

MainActivity.java

```
package com.example.openglrotationapp;

import android.app.Activity;
import android.opengl.GLSurfaceView;
import android.os.Bundle;
```

```
public class MainActivity extends Activity {

    private MyGLSurfaceView glSurfaceView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        glSurfaceView = new MyGLSurfaceView(this);
        setContentView(glSurfaceView);
    }

    @Override
    protected void onPause() {
        super.onPause();
        glSurfaceView.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
        glSurfaceView.onResume();
    }
}
```

🔹 **Step 6: GLSurfaceView Class**

MyGLSurfaceView.java

```
package com.example.openglrotationapp;

import android.content.Context;
import android.opengl.GLSurfaceView;

public class MyGLSurfaceView extends GLSurfaceView {

    private final MyGLRenderer renderer;

    public MyGLSurfaceView(Context context) {
        super(context);

        setEGLContextClientVersion(2); // OpenGL ES 2.0
        renderer = new MyGLRenderer();
        setRenderer(renderer);
        setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
    }
}
```

🔹 **Step 7: OpenGL Renderer**

MyGLRenderer.java

```
package com.example.openglrotationapp;

import android.opengl.GLES20;
```

```
import android.opengl.GLSurfaceView;
import android.opengl.Matrix;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;

import javax.microedition.khronos.opengles.GL10;
import javax.microedition.khronos.egl.EGLConfig;

public class MyGLRenderer implements GLSurfaceView.Renderer {

    private FloatBuffer vertexBuffer;
    private final float[] triangleCoords = { // in counterclockwise order:
            0.0f, 0.622008459f, 0.0f,   // top
           -0.5f, -0.311004243f, 0.0f,  // bottom left
            0.5f, -0.311004243f, 0.0f   // bottom right
    };

    private final float[] color = {0.63671875f, 0.76953125f, 0.22265625f, 1.0f};

    private int program;
    private int positionHandle;
    private int colorHandle;
    private int mvpMatrixHandle;

    private float[] rotationMatrix = new float[16];
    private float[] modelMatrix = new float[16];
    private float[] viewMatrix = new float[16];
    private float[] projectionMatrix = new float[16];
    private float[] mvpMatrix = new float[16];

    private float angle = 0f;

    private final String vertexShaderCode =
            "uniform mat4 uMVPMatrix;" +
            "attribute vec4 vPosition;" +
            "void main() {" +
            "  gl_Position = uMVPMatrix * vPosition;" +
            "}";

    private final String fragmentShaderCode =
            "precision mediump float;" +
            "uniform vec4 vColor;" +
            "void main() {" +
            "  gl_FragColor = vColor;" +
            "}";

    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        // Set background frame color
```

```
    GLES20.glClearColor(0f, 0f, 0f, 1.0f);

    ByteBuffer bb = ByteBuffer.allocateDirect(triangleCoords.length * 4);
    bb.order(ByteOrder.nativeOrder());
    vertexBuffer = bb.asFloatBuffer();
    vertexBuffer.put(triangleCoords);
    vertexBuffer.position(0);

    int vertexShader = loadShader(GLES20.GL_VERTEX_SHADER, vertexShaderCode);
    int fragmentShader = loadShader(GLES20.GL_FRAGMENT_SHADER, fragmentShaderCode);

    program = GLES20.glCreateProgram();             // create empty OpenGL Program
    GLES20.glAttachShader(program, vertexShader);   // add the vertex shader
    GLES20.glAttachShader(program, fragmentShader); // add the fragment shader
    GLES20.glLinkProgram(program);                  // creates OpenGL program executables
}

@Override
public void onDrawFrame(GL10 gl) {
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT | GLES20.GL_DEPTH_BUFFER_BIT);

    // Use program
    GLES20.glUseProgram(program);

    // Get handles
    positionHandle = GLES20.glGetAttribLocation(program, "vPosition");
    colorHandle = GLES20.glGetUniformLocation(program, "vColor");
    mvpMatrixHandle = GLES20.glGetUniformLocation(program, "uMVPMatrix");

    // Pass vertex data
    GLES20.glEnableVertexAttribArray(positionHandle);
    GLES20.glVertexAttribPointer(positionHandle, 3, GLES20.GL_FLOAT, false, 0, vertexBuffer);

    // Pass color
    GLES20.glUniform4fv(colorHandle, 1, color, 0);

    // Apply rotation
    angle += 0.5f;
    Matrix.setRotateM(rotationMatrix, 0, angle, 0, 0, 1.0f);
    Matrix.setIdentityM(modelMatrix, 0);
    Matrix.multiplyMM(modelMatrix, 0, modelMatrix, 0, rotationMatrix, 0);

    // Combine matrices
    Matrix.multiplyMM(mvpMatrix, 0, viewMatrix, 0, modelMatrix, 0);
    Matrix.multiplyMM(mvpMatrix, 0, projectionMatrix, 0, mvpMatrix, 0);

    // Pass transformation to shader
    GLES20.glUniformMatrix4fv(mvpMatrixHandle, 1, false, mvpMatrix, 0);

    // Draw
    GLES20.glDrawArrays(GLES20.GL_TRIANGLES, 0, 3);
```

```
        GLES20.glDisableVertexAttribArray(positionHandle);
    }

    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        GLES20.glViewport(0, 0, width, height);

        float ratio = (float) width / height;

        Matrix.frustumM(projectionMatrix, 0, -ratio, ratio, -1, 1, 3, 7);
        Matrix.setLookAtM(viewMatrix, 0,
                0, 0, -5, // eye position
                0, 0, 0,  // look at
                0, 1, 0   // up vector
        );
    }

    private int loadShader(int type, String shaderCode) {
        int shader = GLES20.glCreateShader(type);
        GLES20.glShaderSource(shader, shaderCode);
        GLES20.glCompileShader(shader);
        return shader;
    }
}
```

🎨 **Output**

A colored **triangle rotates** continuously on the screen using OpenGL ES 2.0.

✅ **Viva Questions**

1. What is OpenGL ES?
2. How does OpenGL differ from OpenGL ES?
3. What is the role of a **GLSurfaceView**?
4. What is a **shader** and why do we use it?
5. How do you rotate an object in OpenGL?

---

10. Application development using Widgets

🎯 **Aim**

To design and implement a **custom clock widget** (analog or digital) in an Android application that displays the current time and can be added to the **device home screen**.

🧰 **Requirements**

- Android Studio
- Java/Kotlin
- Android SDK (API 21+)
- Basic knowledge of AppWidgetProvider and RemoteViews

📋 **Procedure**

🔹 **Step 1: Create a New Android Project**

- Name: ClockWidgetApp
- Language: Java
- Minimum SDK: 21
- Template: **Empty Activity**

🔹 **Step 2: Add Widget Metadata in res/xml**

Create an XML folder inside res (if not already present), then add:

📄 **res/xml/clock_widget_info.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="150dp"
    android:minHeight="60dp"
    android:updatePeriodMillis="60000"
    android:initialLayout="@layout/widget_clock"
    android:resizeMode="horizontal|vertical"
    android:widgetCategory="home_screen" />
```

🔹 **Step 3: Create Widget Layout**

📄 **res/layout/widget_clock.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#222222">

    <TextView
        android:id="@+id/tvClock"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textSize="24sp"
        android:gravity="center"
```

```
    android:textColor="#FFFFFF"
    android:text="00:00" />
</FrameLayout>
```

🔹 **Step 4: Create Widget Provider Class**

📄 **ClockWidget.java**

```
package com.example.clockwidgetapp;

import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.widget.RemoteViews;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class ClockWidget extends AppWidgetProvider {

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[]
appWidgetIds) {

        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm", Locale.getDefault());
        String time = sdf.format(new Date());

        for (int appWidgetId : appWidgetIds) {
            RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.widget_clock);
            views.setTextViewText(R.id.tvClock, time);

            // Refresh on click
            Intent intent = new Intent(context, ClockWidget.class);
            intent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
            intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, appWidgetIds);

            PendingIntent pendingIntent = PendingIntent.getBroadcast(
                    context,
                    0,
                    intent,
                    PendingIntent.FLAG_UPDATE_CURRENT | (Build.VERSION.SDK_INT >= 31 ?
PendingIntent.FLAG_MUTABLE : 0)
            );

            views.setOnClickPendingIntent(R.id.tvClock, pendingIntent);

            appWidgetManager.updateAppWidget(appWidgetId, views);
```

```
    }
  }
}
```

🔹 **Step 5: Declare Widget in AndroidManifest.xml**

Add inside the <application> tag:

```
<receiver android:name=".ClockWidget" android:exported="true">
  <intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
  </intent-filter>

  <meta-data
    android:name="android.appwidget.provider"
    android:resource="@xml/clock_widget_info" />
</receiver>
```

🔹 **Step 6: Run the App and Add Widget**

- Install the app on a real/emulated device.
- Long-press on the home screen > Widgets > Find your widget (ClockWidgetApp) > Drag it onto the home screen.

🖼️ **Output**

A custom digital clock that displays the current time and updates every minute. Clicking it refreshes the time manually.

🧠 **Viva Questions**

1. What is the purpose of AppWidgetProvider?
2. How can you update a widget periodically?
3. What is RemoteViews and why is it used?
4. Can a widget display real-time seconds? Why or why not?
5. How do you make a widget interactive?

---

11. Firebase Push Notification in Android

🎯 **Aim**

To develop an Android application that uses **Firebase Cloud Messaging (FCM)** to receive and display **push notifications**.

🧰 **Requirements**

- Android Studio
- Firebase account (https://console.firebase.google.com/)
- Internet connection
- Android device/emulator

📋 **Procedure**

🔹 **Step 1: Create Firebase Project**

1. Go to Firebase Console
2. Click **"Add Project"** → Enter project name → Accept terms → Click **Create Project**
3. Once created, click **"Add app"** → Choose **Android**
4. Enter your app's **package name** (e.g., com.example.fcmpushdemo)
5. Download the google-services.json file and **add it to your project's app/ folder**

🔹 **Step 2: Add Firebase SDK to Project**

project-level build.gradle

```
buildscript {
    dependencies {
        classpath 'com.google.gms:google-services:4.3.15' // latest as of now
    }
}
```

app-level build.gradle

```
plugins {
    id 'com.android.application'
    id 'com.google.gms.google-services'
}

dependencies {
    implementation 'com.google.firebase:firebase-messaging:23.3.1'
}
```

Sync the project.

🔹 **Step 3: Update AndroidManifest.xml**

```
<uses-permission android:name="android.permission.INTERNET" />

<application
    android:allowBackup="true"
    android:label="FCM Demo"
    android:theme="@style/Theme.AppCompat.Light.NoActionBar">
```

```
    <service
        android:name=".MyFirebaseMessagingService"
        android:exported="false">
        <intent-filter>
            <action android:name="com.google.firebase.MESSAGING_EVENT"/>
        </intent-filter>
    </service>

    <meta-data
        android:name="com.google.firebase.messaging.default_notification_channel_id"
        android:value="fcm_default_channel" />
</application>
```

🔹 **Step 4: Create Firebase Messaging Service**

MyFirebaseMessagingService.java

```
package com.example.fcmpushdemo;

import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.os.Build;
import android.util.Log;

import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;

import com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;

public class MyFirebaseMessagingService extends FirebaseMessagingService {

    private static final String CHANNEL_ID = "fcm_default_channel";

    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        Log.d("FCM", "Message received: " + remoteMessage.getNotification().getBody());

        // Show notification
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
                .setSmallIcon(R.drawable.ic_launcher_foreground)
                .setContentTitle(remoteMessage.getNotification().getTitle())
                .setContentText(remoteMessage.getNotification().getBody())
                .setPriority(NotificationCompat.PRIORITY_HIGH);

        NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(this);

        // Create channel for Android 8.0+
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            CharSequence name = "FCM Channel";
            String description = "Firebase Cloud Messaging Channel";
            int importance = NotificationManager.IMPORTANCE_HIGH;
```

```
            NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name,
importance);
            channel.setDescription(description);
            notificationManager.createNotificationChannel(channel);
        }

        notificationManager.notify(1, builder.build());
    }
}
```

🔹 **Step 5: Get FCM Token (Optional - to test directly)**

Add in MainActivity.java:

```
import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.messaging.FirebaseMessaging;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        FirebaseMessaging.getInstance().getToken()
            .addOnCompleteListener(task -> {
                if (!task.isSuccessful()) {
                    Log.w("FCM", "Fetching FCM registration token failed", task.getException());
                    return;
                }
                String token = task.getResult();
                Log.d("FCM Token", token);
            });
    }
}
```

🔧 **Step 6: Sending Push Notification (Testing)**

You can send notifications in 2 ways:

🟢 **Method 1: Firebase Console**

1. Go to Firebase Console → Project → Cloud Messaging
2. Click **"Send your first message"**
3. Enter title & message
4. Choose your app package and send

🟡 **Method 2: Using Postman or curl (Advanced)**

Send to device token via Firebase Admin API (requires server key)

✅ **Output**

When a message is sent via Firebase, the app receives it and displays a **system notification** with the message title and body.

🧠 Viva Questions

1. What is FCM and how is it different from GCM?
2. What is the purpose of FirebaseMessagingService?
3. How does Android handle background notification delivery?
4. What is a notification channel?
5. Can FCM be used to send data messages without showing notifications?

---

12. Adding Location, Maps & Environmental Sensors in Android

🎯 Aim

To develop an Android application that:

1. Displays the user's **current location** on **Google Maps**
2. Detects **contextual awareness** (e.g., walking or driving using Activity Recognition API)
3. Accesses **environmental sensors** like **light**, **temperature**, and **humidity**.

🧰 Requirements

- Android Studio
- Google Maps API key (from Google Cloud Console)
- A physical Android device (for sensor testing)
- Internet connection

📋 Procedure

🔹 PART 1: Display User Location on Google Maps

🔸 Step 1: Get Google Maps API Key

1. Go to https://console.cloud.google.com
2. Create a new project or select an existing one
3. Enable the **Maps SDK for Android**
4. Create **API Key** and restrict to Android apps with your package name + SHA-1

🔸 Step 2: Add Required Permissions

AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>

<application>
   <meta-data
      android:name="com.google.android.geo.API_KEY"
      android:value="YOUR_API_KEY_HERE" />
</application>
```

🔸 Step 3: Add Dependencies in build.gradle

```
implementation 'com.google.android.gms:play-services-maps:18.2.0'
implementation 'com.google.android.gms:play-services-location:21.0.1'
```

🔸 Step 4: Create MapsActivity.java

```
package com.example.locationmapapp;

import android.Manifest;
import android.content.pm.PackageManager;
```

```
import android.location.Location;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.FragmentActivity;

import com.google.android.gms.location.*;
import com.google.android.gms.maps.*;
import com.google.android.gms.maps.model.*;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

   private GoogleMap mMap;
   FusedLocationProviderClient fusedLocationClient;
   final int LOCATION_PERMISSION_REQUEST = 100;

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

      fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

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

   @Override
   public void onMapReady(@NonNull GoogleMap googleMap) {
      mMap = googleMap;

      if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
         != PackageManager.PERMISSION_GRANTED) {
         ActivityCompat.requestPermissions(this,
            new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
LOCATION_PERMISSION_REQUEST);
         return;
      }

      mMap.setMyLocationEnabled(true);

      fusedLocationClient.getLastLocation()
         .addOnSuccessListener(this, location -> {
            if (location != null) {
               LatLng loc = new LatLng(location.getLatitude(), location.getLongitude());
               mMap.addMarker(new MarkerOptions().position(loc).title("You are here"));
               mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(loc, 15));
```

```
            }
        });
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {
        if (requestCode == LOCATION_PERMISSION_REQUEST &&
            grantResults.length > 0 &&
            grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            onMapReady(mMap);
        }
    }
}
```

---

activity_maps.xml

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

🔹 PART 2: Use of Environmental Sensors

🔸 Step 1: Register Sensors

In MainActivity.java:

```
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;

public class MainActivity extends AppCompatActivity implements SensorEventListener {

    SensorManager sensorManager;
    Sensor lightSensor, tempSensor;

    TextView tvLight, tvTemp;

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

        tvLight = findViewById(R.id.tvLight);
        tvTemp = findViewById(R.id.tvTemp);

        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        lightSensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

```
        tempSensor = sensorManager.getDefaultSensor(Sensor.TYPE_AMBIENT_TEMPERATURE);

        if (lightSensor != null) {
            sensorManager.registerListener(this, lightSensor,
SensorManager.SENSOR_DELAY_NORMAL);
        } else {
            tvLight.setText("Light Sensor not available");
        }

        if (tempSensor != null) {
            sensorManager.registerListener(this, tempSensor,
SensorManager.SENSOR_DELAY_NORMAL);
        } else {
            tvTemp.setText("Temp Sensor not available");
        }
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        if (event.sensor.getType() == Sensor.TYPE_LIGHT) {
            tvLight.setText("Light: " + event.values[0] + " lx");
        } else if (event.sensor.getType() == Sensor.TYPE_AMBIENT_TEMPERATURE) {
            tvTemp.setText("Temperature: " + event.values[0] + " °C");
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {}

    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
    }

    @Override
    protected void onResume() {
        super.onResume();
        sensorManager.registerListener(this, lightSensor,
SensorManager.SENSOR_DELAY_NORMAL);
        sensorManager.registerListener(this, tempSensor,
SensorManager.SENSOR_DELAY_NORMAL);
    }
}
```

---

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:padding="16dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

```
    <TextView android:id="@+id/tvLight"
        android:text="Light: "
        android:textSize="18sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView android:id="@+id/tvTemp"
        android:text="Temperature: "
        android:textSize="18sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

✅ **Output**

- Map displays user's real-time location.
- Sensors display live light level and ambient temperature (if available on the device).

🧠 **Viva Questions**

1. How does FusedLocationProviderClient differ from LocationManager?
2. What is the purpose of SensorManager?
3. Name environmental sensors available in Android.
4. What are the limitations of using Activity Recognition APIs?
5. How do you handle sensor data lifecycle in Android?
