

**BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG  
TRƯỜNG ĐẠI HỌC THỦY LỢI**



**BÀI TẬP THỰC HÀNH SỐ 5**  
**PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG**  
**NỘI DUNG BỔ SUNG: ỨNG DỤNG VỚI CHỦ ĐỀ NÂNG CAO**

STT	Mã sinh viên	Họ và tên	Lớp
1	2251172530	Nguyễn Tiên Trọng	64KTPM3

**Hà Nội, năm 2025**

## BÀI TẬP 1: Content Providers

### Mục tiêu:

- Tìm hiểu cách sử dụng Content Providers để truy cập dữ liệu từ ứng dụng khác (ứng dụng Danh bạ).
- Hiển thị danh sách tên các liên hệ trong danh bạ lên ứng dụng của mình.

### Các bước thực hiện:

#### 1. Thiết lập quyền truy cập:

- Mở file `AndroidManifest.xml` của ứng dụng.
- Thêm quyền `READ_CONTACTS` để xin phép ứng dụng được đọc dữ liệu danh bạ.

#### XML

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

#### 2. Thiết kế giao diện (layout):

- Tạo một `ListView` trong file layout (ví dụ: `activity_main.xml`) để hiển thị danh sách tên liên hệ.

#### XML

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

#### 3. Đọc dữ liệu từ Content Provider:

- Trong Activity chính (ví dụ: `MainActivity.kt`), sử dụng `ContentResolver` để truy vấn dữ liệu từ Content Provider của ứng dụng Danh bạ.
- URI của Content Provider Danh bạ là:  
`ContactsContract.Contacts.CONTENT_URI`
- Sử dụng phương thức `query()` của `ContentResolver` để lấy dữ liệu. Phương thức này trả về một `Cursor` chưa kết quả truy vấn.
- Duyệt `Cursor` để lấy tên của từng liên hệ và lưu vào một `ArrayList<String>`.

#### 4. Hiển thị dữ liệu lên ListView:

- Tạo một  `ArrayAdapter<String>` để đưa dữ liệu từ `ArrayList<String>` lên `ListView`.
- Gán  `ArrayAdapter<String>` cho `ListView`.

### Code ví dụ (`MainActivity.kt`):

#### Kotlin

```
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 androidx.appcompat.app.AppCompatActivity  
import androidx.core.app.ActivityCompat  
import androidx.core.content.ContextCompat
```

```

class MainActivity : AppCompatActivity() {

    private lateinit var listViewContacts: ListView
    private lateinit var contactsList: ArrayList<String>
    private lateinit var contactsAdapter: ArrayAdapter<String>

    private val REQUEST_READ_CONTACTS_PERMISSION = 100

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        listViewContacts = findViewById(R.id.listViewContacts)
        contactsList = ArrayList()

        // Kiểm tra và xin quyền READ_CONTACTS
        if (ContextCompat.checkSelfPermission(
                this,
                Manifest.permission.READ_CONTACTS
            ) != PackageManager.PERMISSION_GRANTED
        ) {
            ActivityCompat.requestPermissions(
                this,
                arrayOf(Manifest.permission.READ_CONTACTS),
                REQUEST_READ_CONTACTS_PERMISSION
            )
        } else {
            loadContacts()
        }
    }

    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<String>,
        grantResults: IntArray
    ) {
        super.onRequestPermissionsResult(requestCode, permissions,
        grantResults)
        if (requestCode == REQUEST_READ_CONTACTS_PERMISSION) {
            if (grantResults.isNotEmpty() && grantResults[0] ==
            PackageManager.PERMISSION_GRANTED) {
                loadContacts()
            } else {
                // Xử lý trường hợp người dùng từ chối cấp quyền
                // Ví dụ: Hiển thị thông báo cho người dùng
                // Giải thích lý do cần quyền truy cập danh bạ
                Toast.makeText(this, "Permission denied",
                Toast.LENGTH_SHORT).show()
            }
        }
    }

    private fun loadContacts() {
        // Lấy dữ liệu từ Content Provider
        val cursor: Cursor? = contentResolver.query(
            ContactsContract.Contacts.CONTENT_URI,
            null,
            null,
            null,
            null
        )

        cursor?.use {

```

```

        if (it.count > 0) {
            while (it.moveToFirst()) {
                val nameIndex =
                    it.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME)
                val name = it.getString(nameIndex)
                contactsList.add(name)
            }
        }
    }

    // Hiển thị dữ liệu lên ListView
    contactsAdapter = ArrayAdapter(this,
        android.R.layout.simple_list_item_1, contactsList)
    listViewContacts.adapter = contactsAdapter
}
}

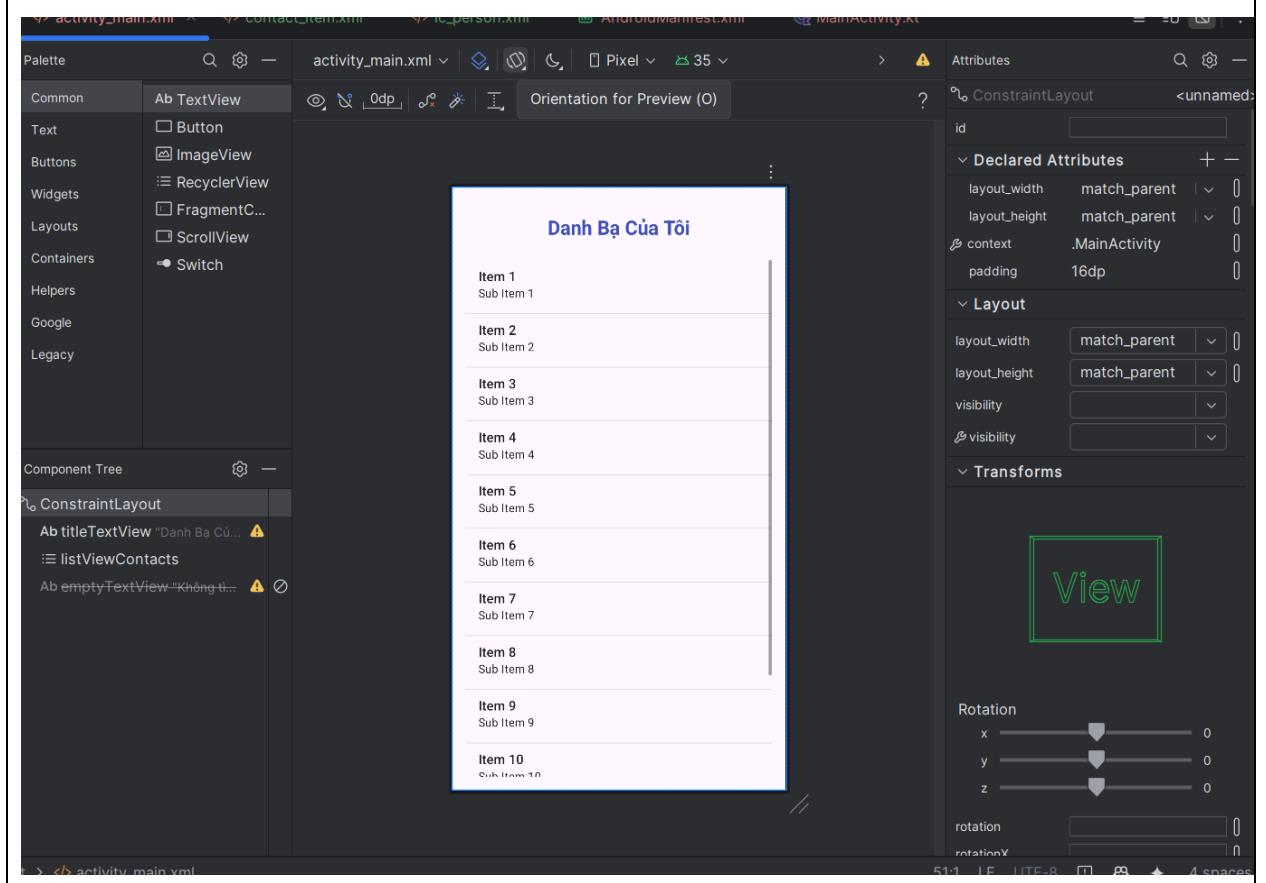
```

### Giải thích chi tiết (Kotlin):

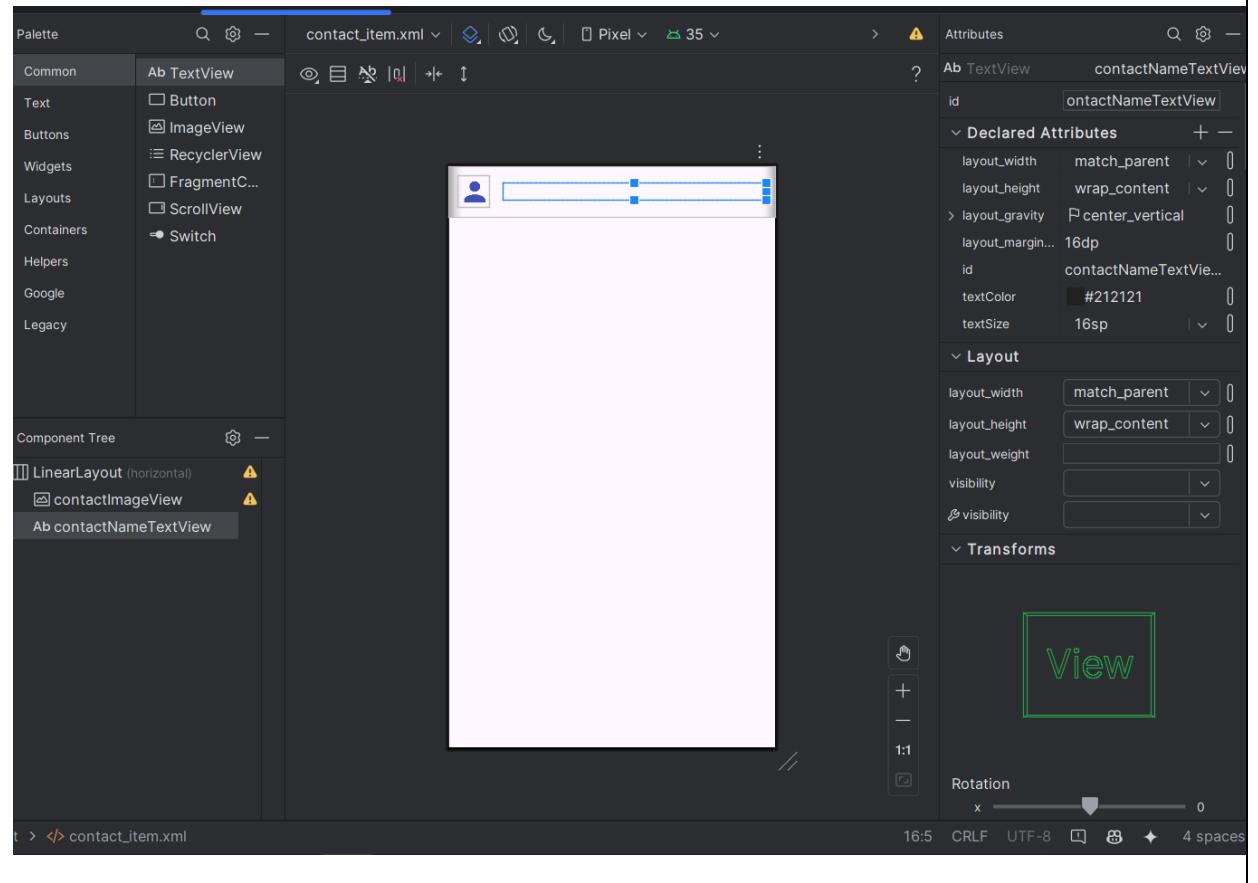
- `Manifest.permission.READ_CONTACTS`: Khai báo quyền truy cập danh bạ trong `AndroidManifest.xml`.
- `ContextCompat.checkSelfPermission()`: Kiểm tra xem ứng dụng đã được cấp quyền `READ_CONTACTS` hay chưa.
- `ActivityCompat.requestPermissions()`: Xin quyền `READ_CONTACTS` từ người dùng nếu chưa được cấp.
- `onRequestPermissionsResult()`: Xử lý kết quả trả về khi người dùng cấp hoặc từ chối quyền.
- `contentResolver`: Đối tượng `ContentResolver` cho phép ứng dụng tương tác với Content Providers.
- `ContactsContract.Contacts.CONTENT_URI`: URI của Content Provider chứa dữ liệu về các liên hệ.
- `Cursor`: Một interface đại diện cho tập kết quả của một truy vấn cơ sở dữ liệu. Trong trường hợp này, nó chứa dữ liệu từ Content Provider.
- `cursor?.use {}`: Sử dụng `use` block để tự động đóng `Cursor` sau khi sử dụng, tránh rò rỉ tài nguyên.
- `it.count`: Lấy số lượng hàng trong `Cursor`.
- `it.moveToNext()`: Di chuyển đến hàng tiếp theo trong `Cursor`.
- `it.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME)`: Lấy chỉ số của cột chứa tên hiển thị của liên hệ.
- `it.getString(nameIndex)`: Lấy giá trị kiểu String từ cột tại chỉ số đã cho.
- `ArrayAdapter<String>`: Adapter để hiển thị danh sách các chuỗi (tên liên hệ) lên `ListView`.
- `listViewContacts.adapter`: Gán `ArrayAdapter` cho `ListView` để hiển thị dữ liệu.

**Hướng dẫn Bài tập 01:** Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

## Tạo giao diện cho main:



## Tạo item cho list view



## Thêm yêu cầu quyền trong file AndroidManifest

```
</> activity_main.xml    </> contact_item.xml    </> ic_person.xml    M AndroidManifest.xml x   MainActivity.kt  
1  <?xml version="1.0" encoding="utf-8"?>  
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  
3      xmlns:tools="http://schemas.android.com/tools">  
4  
5  <uses-permission android:name="android.permission.READ_CONTACTS" />  
6  //  
7  <!--  
8      android:allowBackup="true"  
9      android:dataExtractionRules="@xml/data_extraction_rules"  
10     android:fullBackupContent="@xml/backup_rules"  
11     android:icon="@mipmap/ic_launcher"
```

## Viết hàm yêu cầu quyền truy cập vào danh bạ

```
46  override fun onRequestPermissionsResult(  
47      requestCode: Int,  
48      permissions: Array<out String>,  
49      grantResults: IntArray  
50  ) {  
51      super.onRequestPermissionsResult(requestCode, permissions, grantResults)  
52  
53      if (requestCode == PERMISSIONS_REQUEST_READ_CONTACTS) {  
54          if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {  
55              // Quyền đã được cấp, đọc danh bạ  
56              loadContacts()  
57          } else {  
58              // Quyền bị từ chối  
59              Toast.makeText(  
60                  context, this,  
61                  text: "Bạn cần cấp quyền truy cập danh bạ để sử dụng tính năng này",  
62                  Toast.LENGTH_LONG  
63              ).show()  
64              emptyTextView.text = "Không có quyền truy cập danh bạ"  
65              emptyTextView.visibility = View.VISIBLE  
66          }  
67      }  
68  }  
69 }
```

## Viết hàm lấy dữ liệu từ danh bạ sau khi đã có quyền

```
57
58
59
60
61
62
63
64
65
66
67
68
69
70     private fun loadContacts() {
71         // Danh sách lưu tên các liên hệ
72         val contactsList = ArrayList<String>()
73
74         // Truy vấn danh bạ
75         val cursor: Cursor? = contentResolver.query(
76             ContactsContract.Contacts.CONTENT_URI,
77             projection: null,
78             selection: null,
79             selectionArgs: null,
80             sortOrder: ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME + " ASC"
81         )
82
83         cursor?.use {
84             if (it.count > 0) {
85                 while (it.moveToNext()) {
86                     val nameIndex = it.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME_PRIMARY)
87                     if (nameIndex >= 0) {
88                         val name = it.getString(nameIndex)
89                         contactsList.add(name)
90                     }
91                 }
92             }
93         }
94
95         // Hiển thị kết quả
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2027
2028
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2057
2058
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2078
2079
2079
2080
2081
2082
208
```

Gọi các hàm vừa tạo ở trong onCreate để khi chạy sẽ thực hiện các logic vừa tạo ở các hàm

```
private val PERMISSIONS_REQUEST_READ_CONTACTS = 100
private lateinit var listViewContacts: ListView
private lateinit var emptyTextView: TextView

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    listViewContacts = findViewById(R.id.listViewContacts)
    emptyTextView = findViewById(R.id.emptyTextView)

    // Kiểm tra quyền truy cập danh bạ
    if (ContextCompat.checkSelfPermission(context, Manifest.permission.READ_CONTACTS) != PackageManager.PERMISSION_GRANTED) {

        // Yêu cầu quyền nếu chưa được cấp
        ActivityCompat.requestPermissions(
            activity, this,
            arrayof(Manifest.permission.READ_CONTACTS),
            PERMISSIONS_REQUEST_READ_CONTACTS
        )
    } else {
        // Đã có quyền, đọc danh bạ
        loadContacts()
    }
}
```

## BÀI TẬP 2: Ứng dụng tự động trả lời tin nhắn cuộc gọi nhỡ

### Mục tiêu:

- Sử dụng Broadcast Receiver để lắng nghe sự kiện cuộc gọi đến.
- Sử dụng Telephony API để lấy thông tin về cuộc gọi (số điện thoại).
- Sử dụng SMS API để gửi tin nhắn SMS.

### Mô tả:

Ứng dụng sẽ tự động gửi một tin nhắn SMS đến số điện thoại của người gọi nhỡ, với nội dung thông báo rằng bạn đang bận và sẽ gọi lại sau.

### Các bước thực hiện:

#### 1. Khai báo quyền:

- Thêm các quyền cần thiết vào `AndroidManifest.xml`:
  - `android.permission.READ_PHONE_STATE` (để theo dõi trạng thái cuộc gọi)
  - `android.permission.SEND_SMS` (để gửi tin nhắn SMS)
  - `android.permission.RECEIVE_SMS` (nếu muốn xử lý cả tin nhắn đến)

#### 2. Tạo Broadcast Receiver:

- Tạo một class kế thừa `BroadcastReceiver` để lắng nghe sự kiện `android.intent.action.PHONE_STATE`.
- Trong phương thức `onReceive()`:
  - Kiểm tra trạng thái cuộc gọi (`TelephonyManager.EXTRA_STATE_RINGING`).
  - Nếu là cuộc gọi đến, lấy số điện thoại người gọi (`intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER)`).
  - Nếu cuộc gọi bị nhỡ (có thể theo dõi thêm sự kiện `TelephonyManager.CALL_STATE_IDLE` sau khi đổ chuông), gửi tin nhắn SMS đến số điện thoại đó.

#### 3. Gửi tin nhắn SMS:

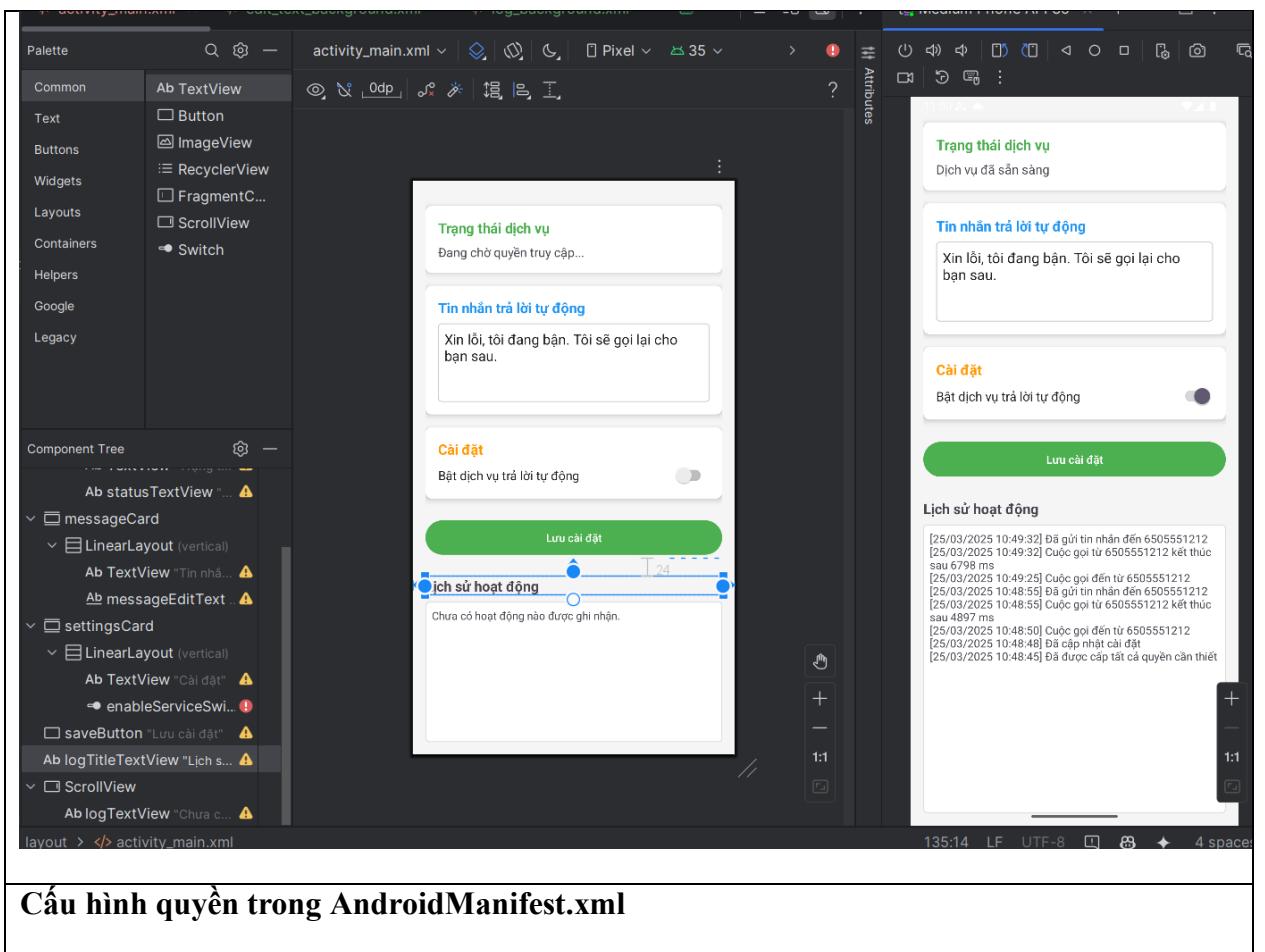
- Sử dụng `SmsManager` để gửi tin nhắn SMS.
- `SmsManager.getDefault().sendTextMessage()` để gửi tin nhắn.

#### 4. Đăng ký Broadcast Receiver:

- Đăng ký receiver trong `AndroidManifest.xml`.

**Hướng dẫn Bài tập 02:** Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

Thiết kế giao diện



## Cấu hình quyền trong AndroidManifest.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools">
4
5     <uses-feature
6         android:name="android.hardware.telephony"
7         android:required="false" />
8
9     <uses-permission android:name="android.permission.READ_PHONE_STATE" />
10    <uses-permission android:name="android.permission.SEND_SMS" />
11    <uses-permission android:name="android.permission.READ_CALL_LOG" />
12    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
13
14    <application
15        android:allowBackup="true"
16        android:dataExtractionRules="@xml/data_extraction_rules"
17        android:fullBackupContent="@xml/backup_rules"
18        android:icon="@mipmap/ic_launcher"
19        android:label="AutoCallResponder"
20        android:roundIcon="@mipmap/ic_launcher_round"
21        android:supportsRtl="true"
22        android:theme="@style/Theme.AutoCallResponder"
23        tools:targetApi="31">
24        <activity
25            android:name=".MainActivity"
26            android:exported="true">
27            <intent-filter>
28                <action android:name="android.intent.action.MAIN" />
29
30                <category android:name="android.intent.category.LAUNCHER" />
31            </intent-filter>
32
33
34        <receiver
35            android:name=".PhoneCallReceiver"
36            android:enabled="true"
37            android:exported="true">
38            <intent-filter>
39                <action android:name="android.telephony.TelephonyManager.ACTION_PHONE_STATE_CHANGED" />
40                <action android:name="android.intent.action.PHONE_STATE" />
41            </intent-filter>
42        </receiver>
43
44        <receiver
45            android:name=".BootReceiver"
46            android:enabled="true"
47            android:exported="true">
48            <intent-filter>
49                <action android:name="android.intent.action.BOOT_COMPLETED" />
50            </intent-filter>
51        </receiver>
52
53    </application>
54
55 </manifest>
```

## Code MainActivity

## Viết 2 hàm để tải các cài đặt và lưu cài đặt để trả lời tự động

```
65
66     private fun loadSettings() {
67         val message = sharedPreferences.getString("message", "Xin lỗi, tôi đang bận. Tôi sẽ gọi lại cho bạn sau.")
68         val isEnabled = sharedPreferences.getBoolean("isEnabled", false)
69         val log = sharedPreferences.getString("log", "Chưa có hoạt động nào được ghi nhận.")
70
71         messageEditText.setText(message)
72         enableServiceSwitch.isChecked = isEnabled
73         logTextView.text = log
74     }
75
76     private fun saveSettings() {
77         val message = messageEditText.text.toString()
78         val isEnabled = enableServiceSwitch.isChecked
79
80         val editor = sharedPreferences.edit()
81         editor.putString("message", message)
82         editor.putBoolean("isEnabled", isEnabled)
83         editor.apply()
84     }
85 }
```

## Kiểm tra quyền và hiển thị thông tin để người dùng biết thông tin

```
private fun checkPermissions() {
    val permissionsToRequest = ArrayList<String>()

    for (permission in requiredPermissions) {
        if (ContextCompat.checkSelfPermission(context, permission) != PackageManager.PERMISSION_GRANTED) {
            permissionsToRequest.add(permission)
        }
    }

    if (permissionsToRequest.isNotEmpty()) {
        ActivityCompat.requestPermissions(
            activity, this,
            permissionsToRequest.toTypedArray(),
            PERMISSIONS_REQUEST_CODE
        )
        statusTextView.text = "Đang chờ cấp quyền...."
    } else {
        statusTextView.text = "Dịch vụ đã sẵn sàng"
    }
}
```

```
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)

    if (requestCode == PERMISSIONS_REQUEST_CODE) {
        var allGranted = true

        for (result in grantResults) {
            if (result != PackageManager.PERMISSION_GRANTED) {
                allGranted = false
                break
            }
        }

        if (allGranted) {
            statusTextView.text = "Dịch vụ đã sẵn sàng"
            addLogEntry(entry: "Đã được cấp tất cả quyền cần thiết")
        } else {
            statusTextView.text = "Thiếu quyền - dịch vụ không thể hoạt động"
            addLogEntry(entry: "Thiếu quyền cần thiết")
            Toast.makeText(
                context, this,
                text: "Bạn cần cấp tất cả quyền để ứng dụng hoạt động",
                Toast.LENGTH_LONG
            ).show()
        }
    }
}
```

## Thêm hàm để ghi lại lịch sử

```
fun addLogEntry(entry: String) {
    val dateFormat = SimpleDateFormat(pattern: "dd/MM/yyyy HH:mm:ss", Locale.getDefault())
    val timestamp = dateFormat.format(Date())

    val currentLog = logTextView.text.toString()
    val newLog = if (currentLog == "Chưa có hoạt động nào được ghi nhận.") {
        "[${timestamp}] $entry"
    } else {
        "[${timestamp}] $entry\n$currentLog"
    }

    logTextView.text = newLog

    // Lưu log vào SharedPreferences
    val editor = sharedpreferences.edit()
    editor.putString("log", newLog)
    editor.apply()
}
```

## Gọi các hàm vừa tạo vào trong onCreate

```
@override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Khởi tạo các thành phần UI
    statusTextView = findViewById(R.id.statusTextView)
    messageEditText = findViewById(R.id.messageEditText)
    enableServiceSwitch = findViewById(R.id.enableServiceSwitch)
    saveButton = findViewById(R.id.saveButton)
    logTextView = findViewById(R.id.logTextView)

    // Khởi tạo SharedPreferences để lưu cài đặt
    sharedpreferences = getSharedPreferences(name: "AutoCallResponderPrefs", MODE_PRIVATE)

    // Tải cài đặt đã lưu
    loadSettings()

    // Kiểm tra quyền
    checkPermissions()

    // Xử lý sự kiện nút Lưu
    saveButton.setOnClickListener {
        saveSettings()
        Toast.makeText(context: this, text: "Đã lưu cài đặt", Toast.LENGTH_SHORT).show()
        addLogEntry(entry: "Đã cập nhật cài đặt")
    }
}
```

## Tạo PhoneCallReceiver để nhận sự kiện cuộc gọi

```

override fun onReceive(context: Context, intent: Intent) {
    val state = intent.getStringExtra(TelephonyManager.EXTRA_STATE)
    val phoneNumber = intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER)

    // Lấy trạng thái dịch vụ từ SharedPreferences
    val sharedpreferences = context.getSharedPreferences(name: "AutoCallResponderPrefs", Context.MODE_PRIVATE)
    val isEnabled = sharedpreferences.getBoolean(key: "isEnabled", defaultValue: false)

    // Nếu dịch vụ không được bật, không làm gì cả
    if (!isEnabled) return

    when (state) {
        TelephonyManager.EXTRA_STATE_RINGING -> {
            if (phoneNumber != null) {
                // Cuộc gọi đến, lưu thời điểm bắt đầu
                callStartTimeMap[phoneNumber] = System.currentTimeMillis()

                // Ghi log
                saveLog(context, entry: "Cuộc gọi đến từ $phoneNumber")
            }
        }

        TelephonyManager.EXTRA_STATE_OFFHOOK -> {
            if (phoneNumber != null) {
                // Cuộc gọi được trả lời
                answeredCalls.add(phoneNumber)

                // Ghi log
                saveLog(context, entry: "Đã trả lời cuộc gọi từ $phoneNumber")
            }
        }
    }
}

```

```

    TelephonyManager.EXTRA_STATE_IDLE -> {
        // Xử lý cuộc gọi kết thúc
        if (phoneNumber != null) {
            handleCallEnded(context, phoneNumber)
        } else {
            // Khi IDLE mà không có số điện thoại, xử lý tất cả các cuộc gọi đang chờ
            handleAllPendingCalls(context)
        }
    }
}

private fun handleCallEnded(context: Context, phoneNumber: String) {
    val startTime = callStartTimeMap[phoneNumber]

    if (startTime != null) {
        val callDuration = System.currentTimeMillis() - startTime

        // Nếu cuộc gọi không được trả lời
        if (!answeredCalls.contains(phoneNumber)) {
            // Gửi tin nhắn SMS
            sendSMS(context, phoneNumber)
        }

        // Xóa số điện thoại khỏi các collections
        callStartTimeMap.remove(phoneNumber)
        answeredCalls.remove(phoneNumber)

        // Ghi log
    }
}

```

```

79         // Ghi log
80         saveLog(context, entry: "Cuộc gọi từ $phoneNumber kết thúc sau $callDuration ms")
81     }
82 }
83
84 private fun handleAllPendingCalls(context: Context) {
85     for (number in callStartTimeMap.keys.toSet()) {
86         val startTime = callStartTimeMap[number]
87         if (startTime != null) {
88             val callDuration = System.currentTimeMillis() - startTime
89
90             // Nếu cuộc gọi không được trả lời
91             if (!answeredCalls.contains(number)) {
92                 // Gửi tin nhắn SMS
93                 sendSMS(context, number)
94             }
95
96             // Xóa số điện thoại khỏi các collections
97             callStartTimeMap.remove(number)
98             answeredCalls.remove(number)
99
100            // Ghi log
101            saveLog(context, entry: "Cuộc gọi từ $number kết thúc sau $callDuration ms")
102        }
103    }
104 }
105
106
107 private fun saveLog(context: Context, entry: String) {
108     try {
109         val sharedpreferences = context.getSharedPreferences( name: "AutoCallResponderPrefs", Context.MODE_PRIVATE)
110         val currentLog = sharedpreferences.getString( key: "log", defValue: "Chưa có hoạt động nào được ghi nhận.") ?: ""
111         val timestamp = java.text.SimpleDateFormat( pattern: "dd/MM/yyyy HH:mm:ss", java.util.Locale.getDefault())
112             .format(java.util.Date())
113
114         val newLog = if (currentLog == "Chưa có hoạt động nào được ghi nhận.") {
115             "[${timestamp}] $entry"
116         } else {
117             "[${timestamp}] $entry\n$currentLog"
118         }
119
120         sharedpreferences.edit().putString("log", newLog).apply()
121     } catch (e: Exception) {
122         Log.e(TAG, msg: "Error saving log", e)
123     }
124 }

```

```

125
126     private fun sendSMS(context: Context, phoneNumber: String) {
127         try {
128             // Lấy tin nhắn từ SharedPreferences
129             val sharedpreferences = context.getSharedPreferences("AutoCallResponderPrefs", Context.MODE_PRIVATE)
130             val message = sharedpreferences.getString("message",
131                 defaultValue: "Xin lỗi, tôi đang bận. Tôi sẽ gọi lại cho bạn sau.")
132
133             // Sử dụng Handler để delay việc gửi SMS một chút
134             Handler(Looper.getMainLooper()).postDelayed({
135                 try {
136                     // Gửi SMS
137                     val smsManager = SmsManager.getDefault()
138                     smsManager.sendTextMessage(phoneNumber, null, message, null, null)
139
140                     // Ghi log
141                     saveLog(context, entry: "Đã gửi tin nhắn đến $phoneNumber")
142                 } catch (e: Exception) {
143                     saveLog(context, entry: "Lỗi khi gửi tin nhắn đến $phoneNumber: ${e.message}")
144                 }
145             }, delayMillis: 500) // Delay 0.5 giây
146
147         } catch (e: Exception) {
148             // Xử lý lỗi
149             saveLog(context, entry: "Lỗi khi chuẩn bị gửi tin nhắn đến $phoneNumber: ${e.message}")
150         }
151     }
152 }
153

```

### Tạo BootReceiver để khởi động dịch vụ khi thiết bị khởi động

```

1 package com.example.autocallresponder
2
3 import android.content.BroadcastReceiver
4 import android.content.Context
5 import android.content.Intent
6
7 class BootReceiver : BroadcastReceiver() {
8     override fun onReceive(context: Context, intent: Intent) {
9         if (intent.action == Intent.ACTION_BOOT_COMPLETED) {
10             // Khởi động lại dịch vụ nếu được bật
11             val sharedpreferences = context.getSharedPreferences("AutoCallResponderPrefs", Context.MODE_PRIVATE)
12             val isEnabled = sharedpreferences.getBoolean("isEnabled", false)
13
14             if (isEnabled) {
15                 // Ghi log
16                 val mainActivity = context.applicationContext as? MainActivity
17                 mainActivity?.addLogEntry(entry: "Dịch vụ đã được khởi động lại sau khi thiết bị khởi động")
18             }
19         }
20     }
21 }
22

```

## BÀI TẬP 3: Ứng dụng chặn cuộc gọi theo số điện thoại

### Mục tiêu:

- Sử dụng Broadcast Receiver để lắng nghe sự kiện cuộc gọi đến.
- Sử dụng Telephony API để lấy thông tin về cuộc gọi (số điện thoại).
- (Nâng cao) Tìm hiểu cách chặn cuộc gọi (có thể cần các phương pháp không chính thức hoặc API riêng của nhà sản xuất điện thoại).

### Mô tả:

Ứng dụng sẽ tự động từ chối hoặc ngắt kết nối các cuộc gọi đến từ một danh sách các số điện thoại bị chặn.

### Các bước thực hiện:

#### 1. Khai báo quyền:

- Thêm quyền android.permission.READ\_PHONE\_STATE vào AndroidManifest.xml.
- (Có thể cần thêm các quyền liên quan đến xử lý cuộc gọi tùy theo phương pháp chặn)

#### 2. Tạo Broadcast Receiver:

- Tạo một class kế thừa BroadcastReceiver để lắng nghe sự kiện android.intent.action.PHONE\_STATE.
- Trong phương thức onReceive():
  - Kiểm tra trạng thái cuộc gọi (TelephonyManager.EXTRA\_STATE\_RINGING).
  - Nếu là cuộc gọi đến, lấy số điện thoại người gọi.
  - Kiểm tra xem số điện thoại đó có nằm trong danh sách chặn hay không.
  - Nếu có trong danh sách chặn, thực hiện hành động chặn cuộc gọi.

#### 3. Chặn cuộc gọi:

- (Phần này có thể phức tạp và phụ thuộc vào phiên bản Android và nhà sản xuất điện thoại)
- Có thể cần sử dụng TelephonyManager hoặc các API khác để thực hiện chặn cuộc gọi.
- Lưu ý rằng việc chặn cuộc gọi có thể bị hạn chế hoặc không được hỗ trợ trên một số thiết bị.

#### 4. Đăng ký Broadcast Receiver:

- Đăng ký receiver trong AndroidManifest.xml.

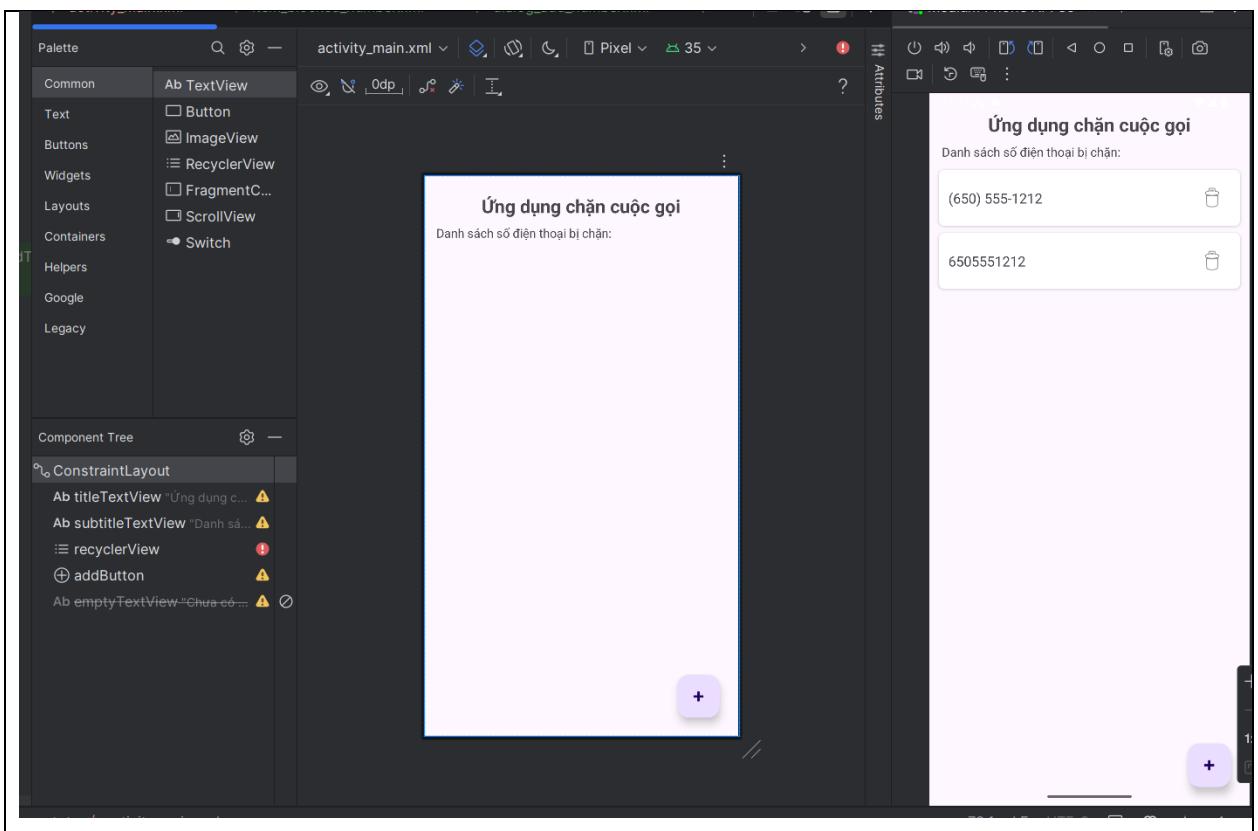
### Lưu ý quan trọng:

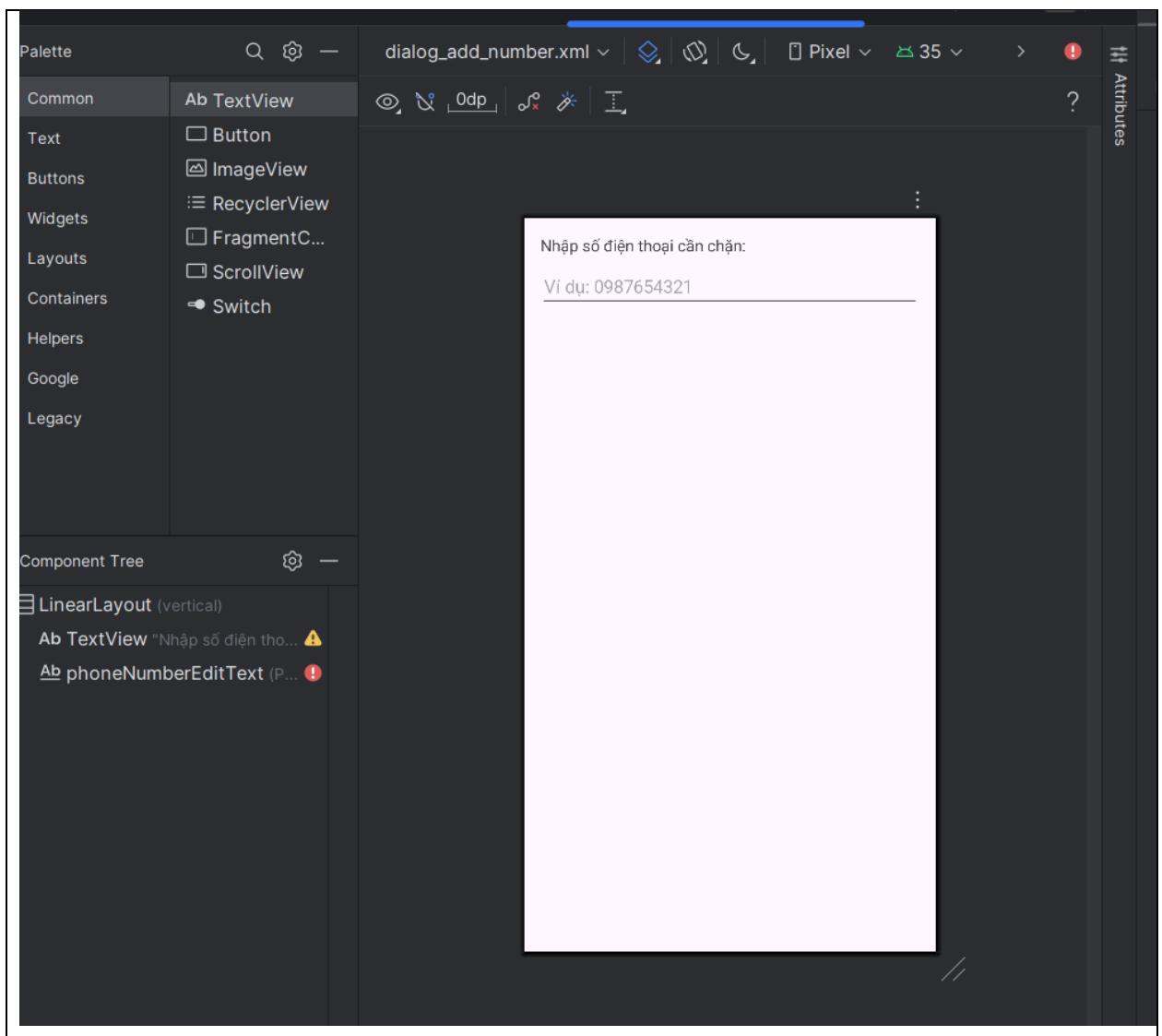
- **Xử lý bất đồng bộ trong onReceive():** Tránh thực hiện các tác vụ tốn thời gian trong phương thức onReceive() của Broadcast Receiver. Nếu cần thực hiện các tác vụ dài, hãy sử dụng Service.
- **Quyền (Permissions):** Các thao tác liên quan đến Telephony và SMS đều yêu cầu các quyền đặc biệt. Đảm bảo bạn đã khai báo đầy đủ các quyền trong AndroidManifest.xml và xử lý việc xin quyền từ người dùng một cách thích hợp (đặc biệt là trên các phiên bản Android mới).
- **Hạn chế của Telephony API:** Một số chức năng liên quan đến Telephony có thể bị hạn chế hoặc không được hỗ trợ trên một số thiết bị hoặc phiên bản Android.

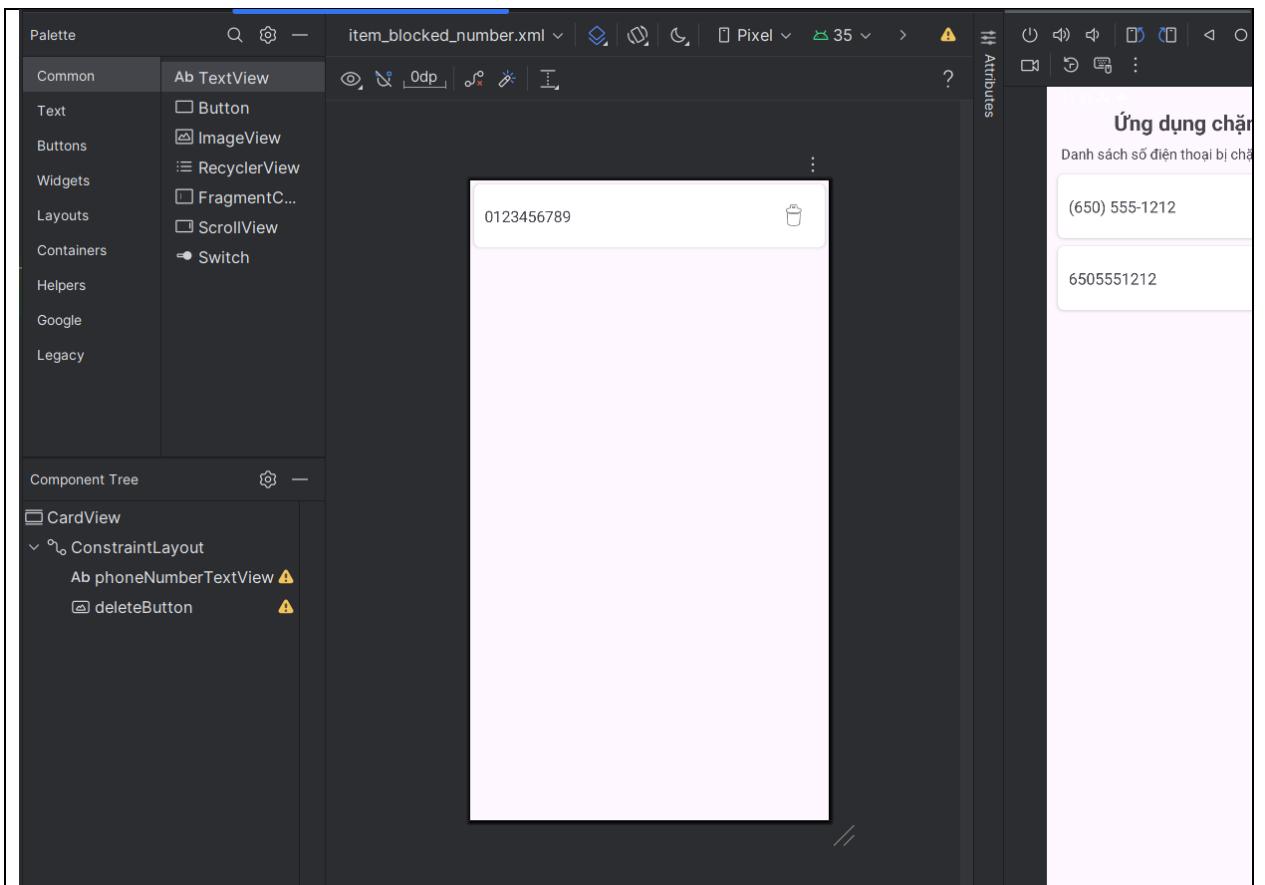
- **SMS PDU:** Khi nhận SMS, dữ liệu thường ở định dạng PDU. Cần xử lý để giải mã và đọc nội dung tin nhắn.

**Hướng dẫn Bài tập 03:** *Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D*

Thiết kế giao diện







**Thêm quyền trong AndroidManifest**

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-feature
        android:name="android.hardware.telephony"
        android:required="false" />

    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.READ_CALL_LOG" />
    <uses-permission android:name="android.permission.ANSWER_PHONE_CALLS" />
    <uses-permission android:name="android.permission.CALL_PHONE" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"

        <receiver
            android:name=".CallReceiver"
            android:enabled="true"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.PHONE_STATE" />
            </intent-filter>
        </receiver>
    </application>
```

## Code Adapter để hiển thị danh sách chặn

```
dialog_add_number.xml    ic_block.xml    MainActivity.kt    CallReceiver.kt    BlockedNumbersAdapter.kt    AndroidManifest.xml
4 import android.view.View
5 import android.view.ViewGroup
6 import android.widget.ImageButton
7 import android.widget.TextView
8 import androidx.recyclerview.widget.RecyclerView
9
10 class BlockedNumbersAdapter(
11     private val blockedNumbers: List<String>,
12     private val onDeleteClick: (Int) -> Unit
13 ) : RecyclerView.Adapter<BlockedNumbersAdapter.ViewHolder>() {
14
15     class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
16         val phoneNumberTextView: TextView = view.findViewById(R.id.phoneNumberTextView)
17         val deleteButton: ImageButton = view.findViewById(R.id.deleteButton)
18     }
19
20     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
21         val view = LayoutInflater.from(parent.context)
22             .inflate(R.layout.item_blocked_number, parent, false)
23         return ViewHolder(view)
24     }
25
26     override fun onBindViewHolder(holder: ViewHolder, position: Int) {
27         val number = blockedNumbers[position]
28         holder.phoneNumberTextView.text = number
29         holder.deleteButton.setOnClickListener {
30             onDeleteClick(position)
31         }
32     }
33
34     override fun getItemCount() = blockedNumbers.size
35 }
```

## Code file main thực hiện các logic

```
19 </> class MainActivity : AppCompatActivity() {
20
21     private lateinit var recyclerView: RecyclerView
22     private lateinit var adapter: BlockedNumbersAdapter
23     private lateinit var addButton: FloatingActionButton
24     private val blockedNumbers = mutableListOf<String>()
25
26     companion object {
27         private const val PERMISSIONS_REQUEST_CODE = 100
28         private val REQUIRED_PERMISSIONS = arrayOf(
29             Manifest.permission.READ_PHONE_STATE,
30             Manifest.permission.READ_CALL_LOG,
31             Manifest.permission.ANSWER_PHONE_CALLS,
32             Manifest.permission.CALL_PHONE
33         )
34     }
35
36     override fun onCreate(savedInstanceState: Bundle?) {
37         super.onCreate(savedInstanceState)
38         setContentView(R.layout.activity_main)
39
40         // Kiểm tra và yêu cầu quyền
41         if (!hasPermissions()) {
42             requestPermissions()
43         }
44
45         // Khởi tạo RecyclerView
46         recyclerView = findViewById(R.id.recyclerView)
```

```

35
36     override fun onCreate(savedInstanceState: Bundle?) {
37         super.onCreate(savedInstanceState)
38         setContentView(R.layout.activity_main)
39
40         // Kiểm tra và yêu cầu quyền
41         if (!hasPermissions()) {
42             requestPermissions()
43         }
44
45         // Khởi tạo RecyclerView
46         recyclerView = findViewById(R.id.recyclerView)
47         recyclerView.layoutManager = LinearLayoutManager(context: this)
48
49         // Tải danh sách số điện thoại bị chặn từ SharedPreferences
50         loadBlockedNumbers()
51
52         // Khởi tạo adapter
53         adapter = BlockedNumbersAdapter(blockedNumbers) { position ->
54             // Xử lý sự kiện xóa số điện thoại
55             removeBlockedNumber(position)
56         }
57         recyclerView.adapter = adapter
58
59         // Nút thêm số điện thoại mới
60         addButton = findViewById(R.id.addButton)
61         addButton.setOnClickListener {
62             showAddNumberDialog()
63         }
64
65     private fun hasPermissions(): Boolean {
66         return REQUIRED_PERMISSIONS.all {
67             ContextCompat.checkSelfPermission(context: this) == PackageManager.PERMISSION_GRANTED
68         }
69     }
70
71     private fun requestPermissions() {
72         ActivityCompat.requestPermissions(activity: this, REQUIRED_PERMISSIONS, PERMISSION_REQUEST_CODE)
73     }
74
75     private fun loadBlockedNumbers() {
76         val sharedPrefs = getSharedPreferences(name: "BlockedNumbers", Context.MODE_PRIVATE)
77         val numbersSet = sharedPrefs.getStringSet(key: "numbers", setOf() ?: setOf())
78         blockedNumbers.addAll(numbersSet)
79     }
80
81     private fun saveBlockedNumbers() {
82         val sharedPrefs = getSharedPreferences(name: "BlockedNumbers", Context.MODE_PRIVATE)
83         val editor = sharedPrefs.edit()
84         editor.putStringSet(key: "numbers", blockedNumbers.toSet())
85         editor.apply()
86     }
87
88 }

```

## BÀI TẬP 4: Ứng dụng tải và hiển thị ảnh từ Internet

## Mục tiêu:

- Sử dụng `AsyncTask` để thực hiện tải ảnh từ một URL trên Internet trong background.
  - Hiển thị ảnh đã tải xuống lên `ImageView` trong UI Thread.
  - Hiển thị progress bar trong khi tải ảnh.

## Mô tả:

Ứng dụng cho phép người dùng nhập một URL ảnh. Sau khi người dùng nhấn nút, ứng dụng sẽ hiển thị một progress bar và bắt đầu tải ảnh từ URL đó trong background. Khi tải xong, ứng dụng sẽ ẩn progress bar và hiển thị ảnh lên ImageView.

### Các bước thực hiện:

#### 1. Thiết kế giao diện:

- o Một EditText để người dùng nhập URL.
- o Một ImageView để hiển thị ảnh.
- o Một ProgressBar để hiển thị tiến trình tải.
- o Một Button để kích hoạt quá trình tải.

#### 2. Tạo AsyncTask:

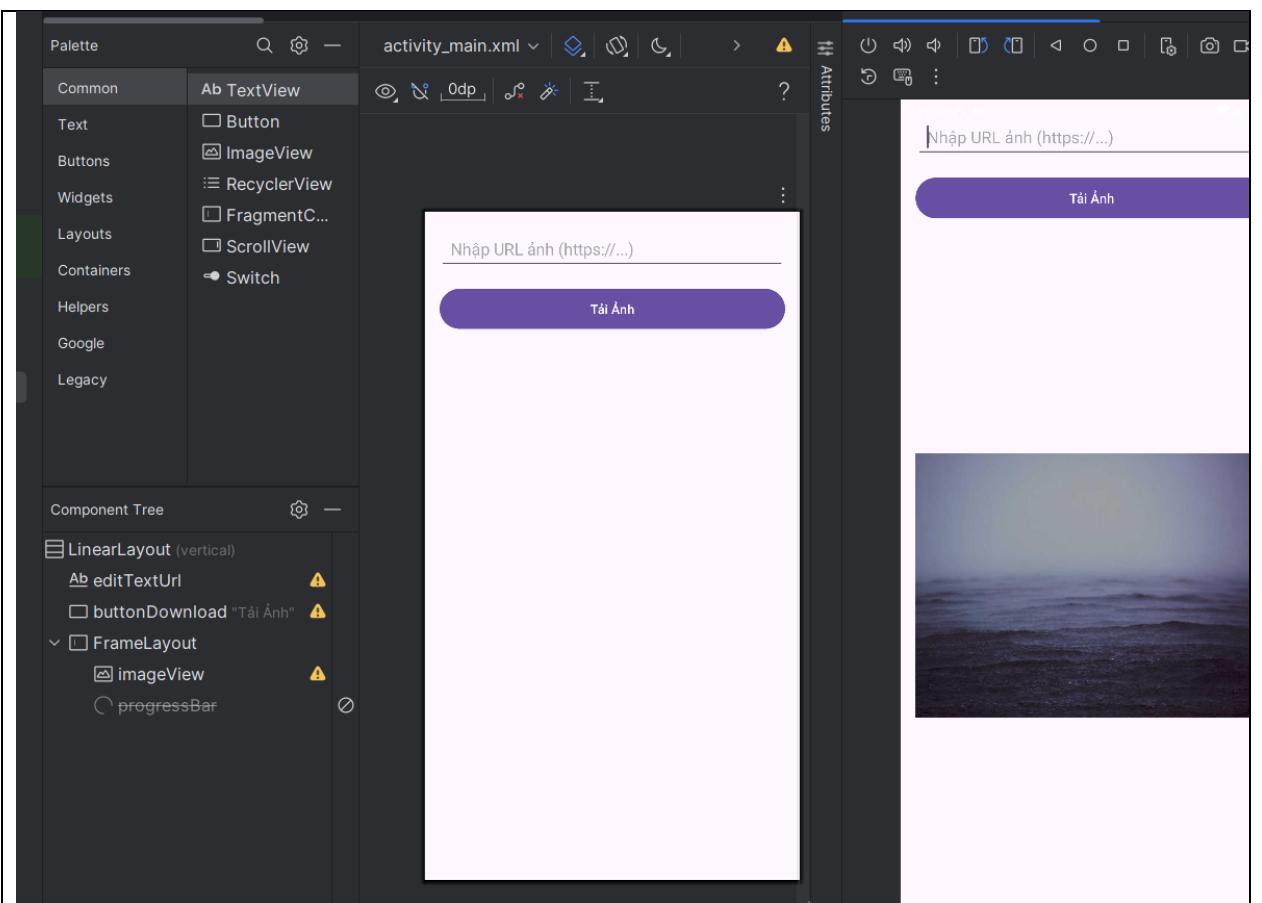
- o Tạo một class kế thừa AsyncTask<String, Integer, Bitmap>.
  - String: URL của ảnh.
  - Integer: Tiến trình tải (ví dụ: phần trăm hoàn thành).
  - Bitmap: Ảnh đã tải.
- o Implement các phương thức:
  - onPreExecute(): Hiển thị progress bar.
  - doInBackground(String... urls):
    - Tải ảnh từ URL (sử dụng các thư viện như HttpURLConnection hoặc OkHttp).
    - Trong quá trình tải, gọi publishProgress() để cập nhật tiến trình (ví dụ: phần trăm tải).
    - Trả về Bitmap của ảnh đã tải.
  - onProgressUpdate(Integer... values): Cập nhật progress bar trên UI thread.
  - onPostExecute(Bitmap result):
    - Ẩn progress bar.
    - Hiển thị ảnh lên ImageView (nếu tải thành công).

#### 3. Xử lý sự kiện Button click:

- o Khi người dùng click nút, lấy URL từ EditText.
- o Tạo một instance của AsyncTask và gọi execute(url).

**Hướng dẫn Bài tập 04:** Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

Thiết kế giao diện



### Thêm quyền trong AndroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/full_backup_content"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportRtl="true"
    android:theme="@style/AppTheme">
```

### Code file main thực hiện các logic

```
48     // AsyncTask để tải ảnh trong background
49     private inner class ImageDownloader : AsyncTask<String, Void, Bitmap?>() {
50
51     @Override
52         override fun onPreExecute() {
53             // Hiển thị progress bar trước khi tải
54             progressBar.visibility = View.VISIBLE
55         }
56
57     @Override
58         override fun doInBackground(vararg urls: String): Bitmap? {
59             val imageUrl = urls[0]
60             var bitmap: Bitmap? = null
61
62             try {
63                 // Tải ảnh từ URL
64                 val inputStream: InputStream = URL(imageUrl).openStream()
65                 bitmap = BitmapFactory.decodeStream(inputStream)
66                 inputStream.close()
67             } catch (e: Exception) {
68                 e.printStackTrace()
69             }
70
71             return bitmap
72     }
73 }
```

```

        }

        override fun onPostExecute(result: Bitmap?) {
            // Ẩn progress bar sau khi tải xong
            progressBar.visibility = View.GONE

            if (result != null) {
                // Hiển thị ảnh đã tải
                imageView.setImageBitmap(result)
                Toast.makeText(context: this@MainActivity, text: "Tải ảnh thành công!", Toast.LENGTH_SHORT).show()
            } else {
                // Thông báo lỗi nếu tải thất bại
                Toast.makeText(context: this@MainActivity, text: "Không thể tải ảnh. Kiểm tra URL và kết nối internet")
            }
        }
    }
}

17 ></> class MainActivity : AppCompatActivity() {
18
19     private lateinit var editTextUrl: EditText
20     private lateinit var buttonDownload: Button
21     private lateinit var imageView: ImageView
22     private lateinit var progressBar: ProgressBar
23
24     override fun onCreate(savedInstanceState: Bundle?) {
25         super.onCreate(savedInstanceState)
26         setContentView(R.layout.activity_main)
27
28         // Khởi tạo các thành phần giao diện
29         editTextUrl = findViewById(R.id.editTextUrl)
30         buttonDownload = findViewById(R.id.buttonDownload)
31         imageView = findViewById(R.id.imageView)
32         progressBar = findViewById(R.id.progressBar)
33
34         // Thêm URL mẫu để học sinh dễ thử nghiệm
35         editTextUrl.setText("https://picsum.photos/800/600")
36
37         // Xử lý sự kiện khi nhấn nút tải
38         buttonDownload.setOnClickListener {
39             val url = editTextUrl.text.toString()
40             if (url.isNotEmpty()) {
41                 ImageDownloader().execute(url)
42             } else {
43                 Toast.makeText(context: this, text: "Vui lòng nhập URL ảnh", Toast.LENGTH_SHORT).show()
44             }
45         }
46     }
}

```

## BÀI TẬP 5: Ứng dụng đếm giờ và cập nhật giao diện

### Mục tiêu:

- Sử dụng Handler và Runnable để cập nhật UI định kỳ từ một thread khác.
- Hiển thị thời gian đã trôi qua trên TextView.

### Mô tả:

Ứng dụng hiển thị một TextView để hiển thị thời gian đã trôi qua (ví dụ: số giây). Một thread nền sẽ tăng giá trị thời gian và sử dụng Handler để gửi thông tin cập nhật lên UI thread để hiển thị.

### Các bước thực hiện:

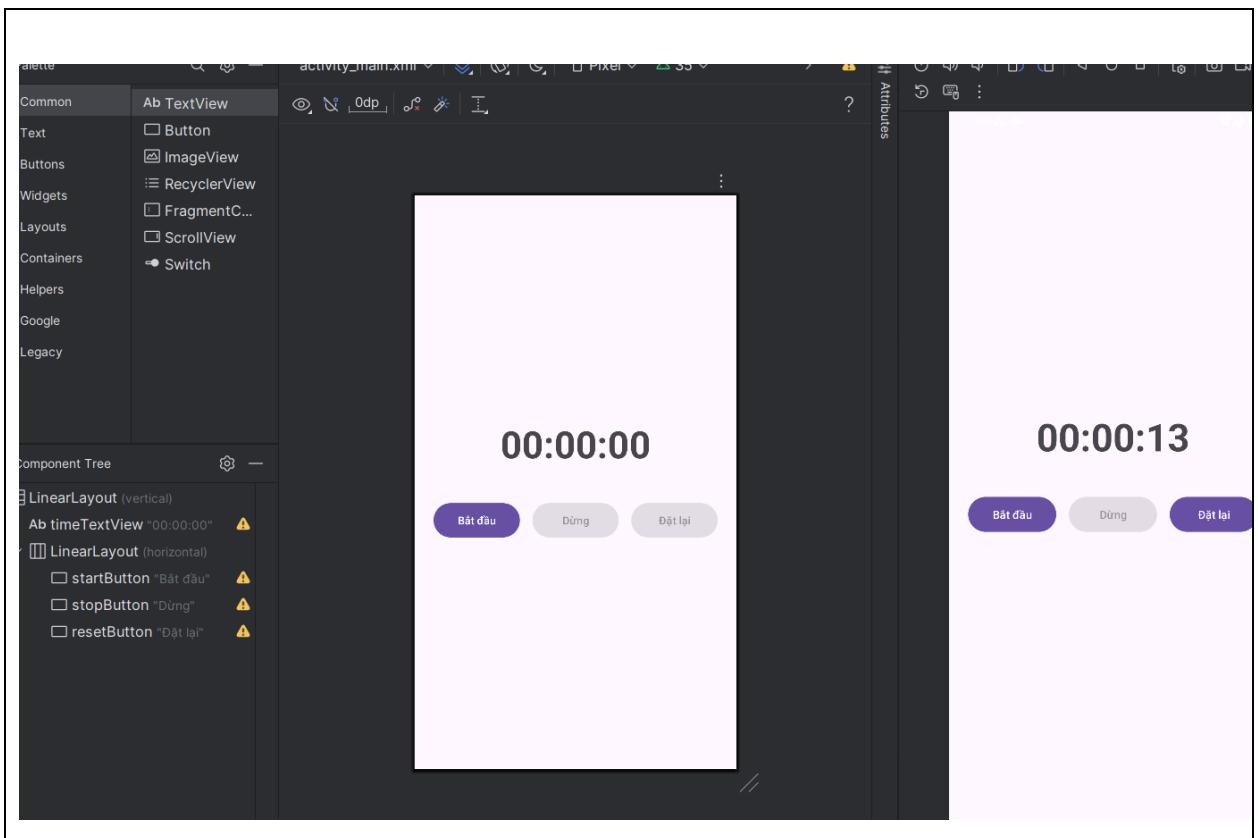
1. **Thiết kế giao diện:**
  - o Một TextView để hiển thị thời gian.
2. **Tạo Handler:**
  - o Tạo một Handler trong Activity chính.
  - o Override phương thức `handleMessage()` (nếu dùng Message) hoặc tạo một Runnable.
3. **Tạo Thread:**
  - o Tạo một Thread mới.
  - o Trong `run()` method:
    - Lặp lại việc tăng giá trị thời gian.
    - Sử dụng `Handler.post(runnable)` để gửi một Runnable lên UI thread để cập nhật TextView.
4. **Cập nhật TextView:**
  - o Trong Runnable, cập nhật `TextView.setText()` với giá trị thời gian hiện tại.

### Lưu ý:

- **UI Thread:** Chỉ có UI thread mới được phép trực tiếp cập nhật các thành phần giao diện người dùng.
- **Tránh các tác vụ dài trên UI Thread:** Các tác vụ tốn thời gian (ví dụ: tải dữ liệu mạng, xử lý ảnh) nên được thực hiện trong background thread để tránh làm treo ứng dụng.
- **Sử dụng AsyncTask cho các tác vụ đơn giản:** AsyncTask là một cách dễ dàng để thực hiện các tác vụ nền đơn giản và tương tác với UI thread.
- **Sử dụng Handler và Thread cho các tác vụ phức tạp hơn:** Handler và Thread cung cấp sự linh hoạt cao hơn cho các tác vụ nền phức tạp hoặc cần kiểm soát thread chi tiết hơn.

**Hướng dẫn Bài tập 05:** Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

Thiết kế giao diện



**Code file main thực hiện các logic**

```
11 ></> class MainActivity : AppCompatActivity() {
12
13     private lateinit var timeTextView: TextView
14     private lateinit var startButton: Button
15     private lateinit var stopButton: Button
16     private lateinit var resetButton: Button
17
18     private val handler = Handler(Looper.getMainLooper())
19     private var seconds = 0
20     private var isRunning = false
21
22     // Runnable để cập nhật thời gian
23     private val timeRunnable = object : Runnable {
24         override fun run() {
25             if (isRunning) {
26                 seconds++
27                 updateTimeDisplay()
28                 // Lặp lịch chạy lại sau 1 giây
29                 handler.postDelayed(this, delayMillis: 1000)
30             }
31         }
32     }
33 }
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Khởi tạo các thành phần giao diện
    timeTextView = findViewById(R.id.timeTextView)
    startButton = findViewById(R.id.startButton)
    stopButton = findViewById(R.id.stopButton)
    resetButton = findViewById(R.id.resetButton)

    // Thiết lập sự kiện cho nút Bắt đầu
    startButton.setOnClickListener {
        startTimer()
    }

    // Thiết lập sự kiện cho nút Dừng
    stopButton.setOnClickListener {
        stopTimer()
    }

    // Thiết lập sự kiện cho nút Đặt lại
    resetButton.setOnClickListener {
        resetTimer()
    }

    // Hiển thị thời gian ban đầu
    updateTimeDisplay()
}
```

```

private fun startTimer() {
    if (!isRunning) {
        isRunning = true
        // Bắt đầu đếm thời gian
        handler.post(timeRunnable)

        // Cập nhật trạng thái các nút
        startButton.isEnabled = false
        stopButton.isEnabled = true
        resetButton.isEnabled = true
    }
}

private fun stopTimer() {
    if (isRunning) {
        isRunning = false
        // Dừng đếm thời gian
        handler.removeCallbacks(timeRunnable)

        // Cập nhật trạng thái các nút
        startButton.isEnabled = true
        stopButton.isEnabled = false
    }
}

private fun resetTimer() {
    // Dừng đếm thời gian nếu đang chạy
    stopTimer()

    // Đặt lại giá trị thời gian
    seconds = 0
    updateTimeDisplay()

    // Cập nhật trạng thái các nút
    startButton.isEnabled = true
    stopButton.isEnabled = false
    resetButton.isEnabled = true
}

private fun updateTimeDisplay() {
    // Chuyển đổi tổng số giây thành giờ:phút:giây
    val hours = TimeUnit.SECONDS.toHours(seconds.toLong())
    val minutes = TimeUnit.SECONDS.toMinutes(seconds.toLong()) % 60
    val secs = seconds % 60

    // Hiển thị thời gian với định dạng HH:MM:SS
    val timeString = String.format("%02d:%02d:%02d", hours, minutes, secs)
    timeTextView.text = timeString
}

```

```
    }

    override fun onDestroy() {
        super.onDestroy()
        // Đảm bảo dừng handler khi activity bị hủy
        handler.removeCallbacks(timeRunnable)
    }
}
```

## BÀI TẬP 6: Ứng dụng ghi âm và phát lại

### Mục tiêu:

- Sử dụng `MediaRecorder` để ghi âm từ microphone của thiết bị.
- Lưu file ghi âm vào `MediaStore` để các ứng dụng khác có thể truy cập.
- Sử dụng `MediaPlayer` để phát lại file ghi âm.
- Hiển thị danh sách các file ghi âm đã lưu trong `MediaStore`.

### Mô tả:

Ứng dụng cho phép người dùng ghi âm, xem danh sách các bản ghi đã thực hiện và phát lại chúng.

### Các bước thực hiện:

#### 1. Ghi âm:

- Sử dụng `MediaRecorder` để ghi âm.
- Thiết lập các thông số cần thiết như nguồn âm thanh, định dạng file, nơi lưu trữ.
- Lưu file ghi âm vào một vị trí cụ thể.
- Sử dụng `ContentValues` để thêm thông tin về file ghi âm vào `MediaStore`.

#### 2. Phát lại:

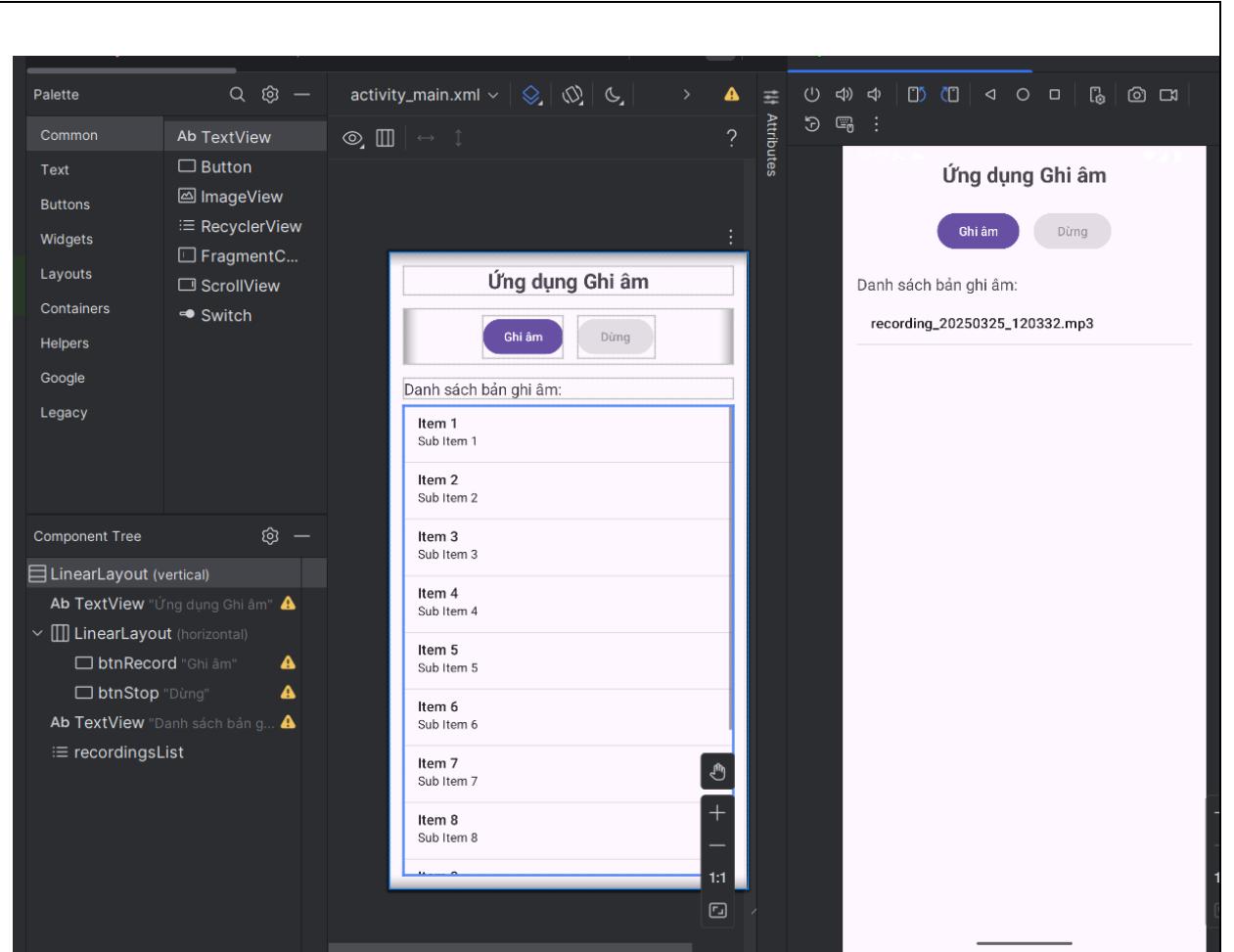
- Sử dụng `MediaPlayer` để phát lại file ghi âm.
- Có thể phát từ file hoặc từ một URI trong `MediaStore`.

#### 3. Hiển thị danh sách file ghi âm:

- Sử dụng `ContentResolver` để truy vấn `MediaStore` và lấy danh sách các file âm thanh.
- Hiển thị danh sách này lên giao diện người dùng (ví dụ: `ListView`).

**Hướng dẫn Bài tập 06:** Chụp lại mà hình từng bước thực hiện (trong tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

Thiết kế giao diện



### Code file main thực hiện các logic

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

    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

21 </> class MainActivity : AppCompatActivity() {

22
23     private val PERMISSION_REQUEST_CODE = 200
24     private var mediaRecorder: MediaRecorder? = null
25     private var mediaPlayer: MediaPlayer? = null
26     private var isRecording = false
27     private var recordingFile: File? = null
28     private lateinit var btnRecord: Button
29     private lateinit var btnStop: Button
30     private lateinit var recordingsList: ListView
31     private val recordings = mutableListOf<Recording>()
32     private lateinit var adapter: ArrayAdapter<Recording>
33
34     override fun onCreate(savedInstanceState: Bundle?) {
35         super.onCreate(savedInstanceState)
36         setContentView(R.layout.activity_main)

37
38         btnRecord = findViewById(R.id.btnRecord)
39         btnStop = findViewById(R.id.btnStop)
40         recordingsList = findViewById(R.id.recordingsList)

41
42         // Thiết lập adapter cho ListView
43         adapter = ArrayAdapter(context: this, android.R.layout.simple_list_item_1, recordings)
44         recordingsList.adapter = adapter

45
46         // Cập nhật danh sách bản ghi
47         loadRecordings()

48
49         // Xử lý sự kiện nút Ghi âm
50         btnRecord.setOnClickListener {

51             // Xử lý sự kiện nút Ghi âm
52             btnRecord.setOnClickListener {
53                 if (checkPermissions()) {
54                     startRecording()
55                 } else {
56                     requestPermissions()
57                 }
58             }

59             // Xử lý sự kiện nút Dừng
60             btnStop.setOnClickListener {
61                 if (isRecording) {
62                     stopRecording()
63                 }
64             }

65             // Xử lý sự kiện khi chọn một bản ghi
66             recordingsList.setOnItemClickListener { _, _, position, _ ->
67                 playRecording(recordings[position].filePath)
68             }
69         }
70     }
}
```



```

21     class MainActivity : AppCompatActivity() {
103         private fun startRecording() {
123             mediaRecorder?.apply {
133                 btnStop.isEnabled = true
134                 Toast.makeText(context: this@MainActivity, text: "Đang ghi âm...", Toast.LENGTH_SHORT).show()
135             }
136             } catch (e: IOException) {
137                 Toast.makeText(context: this@MainActivity, text: "Lỗi khi ghi âm: ${e.message}", Toast.LENGTH_SHORT).show()
138             }
139         }
140
141         private fun stopRecording() {
142             if (mediaRecorder != null) {
143                 try {
144                     mediaRecorder?.apply {
145                         stop()
146                         reset()
147                         release()
148                     }
149                     mediaRecorder = null
150                     isRecording = false
151                     btnRecord.isEnabled = true
152                     btnStop.isEnabled = false
153
154                     // Lưu thông tin bản ghi vào MediaStore
155                     saveToMediaStore()
156
157                     // Cập nhật danh sách
158                     loadRecordings()
159
160                     Toast.makeText(context: this, text: "Đã lưu bản ghi âm", Toast.LENGTH_SHORT).show()
161                 } catch (e: Exception) {
162
21     class MainActivity : AppCompatActivity() {
141         private fun stopRecording() {
159
160             Toast.makeText(context: this, text: "Đã lưu bản ghi âm", Toast.LENGTH_SHORT).show()
161         } catch (e: Exception) {
162             Toast.makeText(context: this, text: "Lỗi khi dừng ghi âm: ${e.message}", Toast.LENGTH_SHORT).show()
163         }
164     }
165
166
167         private fun saveToMediaStore() {
168             recordingFile?.let { file ->
169                 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
170                     val values = ContentValues().apply {
171                         put(MediaStore.Audio.Media.DISPLAY_NAME, file.name)
172                         put(MediaStore.Audio.Media.MIME_TYPE, "audio/mp3")
173                         put(MediaStore.Audio.Media.RELATIVE_PATH, Environment.DIRECTORY_MUSIC + "/MyRecordings")
174                         put(MediaStore.Audio.Media.IS_PENDING, 1)
175                     }
176
177                     val resolver = contentResolver
178                     val uri = resolver.insert(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, values)
179
180                     uri?.let {
181                         values.clear()
182                         values.put(MediaStore.Audio.Media.IS_PENDING, 0)
183                         resolver.update(uri, values, where: null, selectionArgs: null)
184                     }
185                 }
186
187             }
188
189             // Với Android 9 trở xuống, file đã được lưu trực tiếp vào bộ nhớ ngoài
190
191         }

```

```

189     private fun loadRecordings() {
190         recordings.clear()
191         val recordingsDir = File(getExternalFilesDir(Environment.DIRECTORY_MUSIC), "MyRecordings")
192
193         if (recordingsDir.exists()) {
194             recordingsDir.listFiles()?.forEach { file ->
195                 if (file.name.endsWith(".mp3")) {
196                     recordings.add(Recording(file.name, file.absolutePath))
197                 }
198             }
199         }
200     }
201
202     adapter.notifyDataSetChanged()
203 }
204
205     private fun playRecording(filePath: String) {
206         // Dừng phát nếu đang phát
207         mediaPlayer?.release()
208
209         // Tạo MediaPlayer mới
210         mediaPlayer = MediaPlayer().apply {
211             try {
212                 setDataSource(filePath)
213                 prepare()
214                 start()
215                 Toast.makeText(context: this@MainActivity, text: "Đang phát...", Toast.LENGTH_SHORT).show()
216
217                 // Xử lý khi phát xong
218                 setOnCompletionListener {
219
220                     mediaPlayer?.release()
221                     mediaPlayer = null
222                 }
223             } catch (e: Exception) {
224                 Toast.makeText(context: this@MainActivity, text: "Lỗi khi phát: ${e.message}", Toast.LENGTH_SHORT).show()
225             }
226         }
227     }
228
229     override fun onRequestPermissionsResult(
230         requestCode: Int,
231         permissions: Array<out String>,
232         grantResults: IntArray
233     ) {
234         super.onRequestPermissionsResult(requestCode, permissions, grantResults)
235         if (requestCode == PERMISSION_REQUEST_CODE) {
236             if (grantResults.isNotEmpty() && grantResults.all { it == PackageManager.PERMISSION_GRANTED }) {
237                 Toast.makeText(context: this, text: "Đã được cấp quyền", Toast.LENGTH_SHORT).show()
238                 // Tự động bắt đầu ghi âm khi được cấp quyền
239                 startRecording()
240             } else {
241                 Toast.makeText(context: this, text: "Cần cấp quyền để ghi âm", Toast.LENGTH_SHORT).show()
242             }
243         }
244     }
245
246     // Lớp đối tượng để lưu thông tin bản ghi
247     data class Recording(val name: String, val filePath: String) {
248         override fun toString(): String = name
249     }
250
251     override fun onDestroy() {
252         super.onDestroy()
253     }

```

## BÀI TẬP 7: Ứng dụng chơi video đơn giản

### Mục tiêu:

- Sử dụng VideoView và MediaController để phát video.
- Cho phép người dùng chọn video từ MediaStore hoặc từ một URL.

### Mô tả:

Ứng dụng cho phép người dùng xem video từ các nguồn khác nhau trên thiết bị hoặc từ Internet.

### Các bước thực hiện:

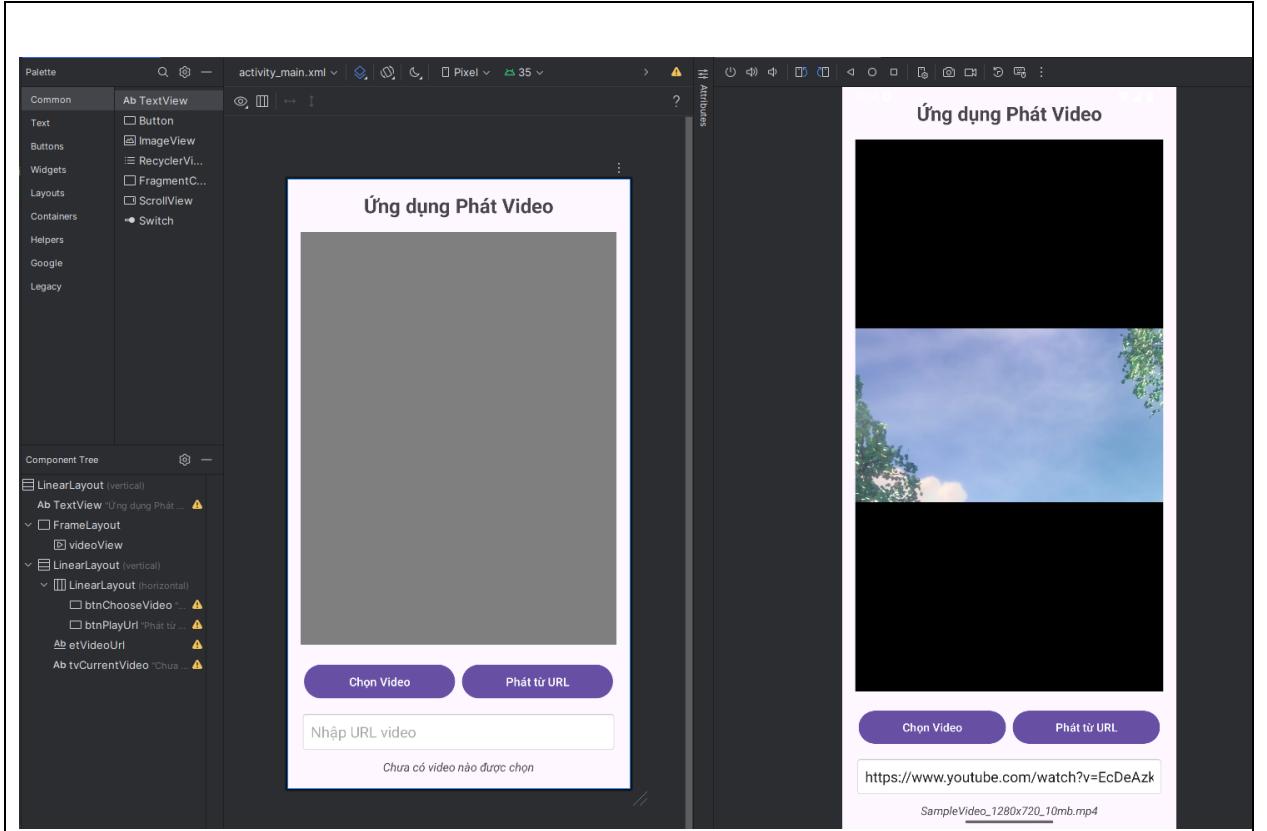
1. **Thêm VideoView và MediaController vào layout.**
2. **Chọn video:**
  - Cho phép người dùng chọn video từ MediaStore bằng cách sử dụng Intent.ACTION\_PICK và MediaStore.Video.Media.EXTERNAL\_CONTENT\_URI.
  - Hoặc cho phép người dùng nhập URL của video.
3. **Phát video:**
  - Sử dụng VideoView.setVideoURI() để thiết lập nguồn video.
  - Sử dụng MediaController để cung cấp các điều khiển phát lại video (play, pause, stop,...).
  - VideoView.start() để bắt đầu phát video.

### Lưu ý:

- **Quyền (Permissions):** Đừng quên khai báo các quyền cần thiết trong AndroidManifest.xml, chẳng hạn như android.permission.RECORD\_AUDIO, android.permission.READ\_EXTERNAL\_STORAGE, android.permission.WRITE\_EXTERNAL\_STORAGE, và android.permission.INTERNET.
- **Xử lý lỗi:** Cần xử lý các trường hợp lỗi có thể xảy ra, chẳng hạn như không tìm thấy file, lỗi khi ghi âm, lỗi khi phát lại, v.v.
- **Vòng đời (Lifecycle):** Quản lý các tài nguyên Media một cách chính xác trong các phương thức lifecycle của Activity/Fragment để tránh rò rỉ bộ nhớ và các vấn đề khác. Ví dụ, cần release() MediaPlayer và MediaRecorder khi không còn sử dụng.
- **MediaStore:** Làm quen với cách truy vấn và sử dụng MediaStore để lấy thông tin về các file media trên thiết bị.

**Hướng dẫn Bài tập 07:** Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

Thiết kế giao diện



## Code file main thực hiện các logic

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_MEDIA_VIDEO" />
</manifest>

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp">
    <VideoView
        android:id="@+id/videoView"
        android:layout_width="match_parent"
        android:layout_height="250dp"/>
    <Button
        android:id="@+id/btnChooseVideo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Chọn video"/>
    <EditText
        android:id="@+id/etVideoUrl"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@+id/tvCurrentVideo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

```
> class MainActivity : AppCompatActivity() {

    private lateinit var videoView: VideoView
    private lateinit var mediaController: MediaController
    private lateinit var btnChooseVideo: Button
    private lateinit var btnPlayUrl: Button
    private lateinit var etVideoUrl: EditText
    private lateinit var tvCurrentVideo: TextView

    private val PERMISSION_REQUEST_CODE = 100
    private val VIDEO_PICK_CODE = 1000

    private var currentVideoUri: Uri? = null
    private var currentVideoTitle: String = "Chưa có video nào được chọn"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Khởi tạo views
        videoView = findViewById(R.id.videoView)
        btnChooseVideo = findViewById(R.id.btnChooseVideo)
        btnPlayUrl = findViewById(R.id.btnPlayUrl)
        etVideoUrl = findViewById(R.id.etVideoUrl)
        tvCurrentVideo = findViewById(R.id.tvCurrentVideo)

        // Thiết lập MediaController
        mediaController = MediaController(context = this)
        mediaController.setAnchorView(videoView)
        videoView.setMediaController(mediaController)

        // Xử lý sự kiện khi video hoàn thành
        videoView.setOnCompletionListener {
            Toast.makeText(context = this, text = "Video đã phát xong", Toast.LENGTH_SHORT).show()
        }
    }
}
```

```
        Toast.makeText( context: this, text: "Video đã phát xong", Toast.LENGTH_SHORT).show()
    }

    // Xử lý lỗi khi phát video
    videoView.setOnErrorListener { _, what, extra ->
        Toast.makeText(
            context: this,
            text: "Lỗi khi phát video: $what, $extra",
            Toast.LENGTH_SHORT
        ).show()
        true
    }

    // Xử lý sự kiện nút Chọn Video
    btnChooseVideo.setOnClickListener {
        if (checkPermissions()) {
            pickVideoFromGallery()
        } else {
            requestPermissions()
        }
    }

    // Xử lý sự kiện nút Phát từ URL
    btnPlayUrl.setOnClickListener {
        showUrlInputDialog()
    }

    // Khôi phục trạng thái nếu có
    if (savedInstanceState != null) {
        val videoPath = savedInstanceState.getString(key: "videoPath")
        val videoPosition = savedInstanceState.getInt(key: "videoPosition", defaultValue: 0)
        val videoTitle = savedInstanceState.getString(key: "videoTitle")

        if (videoPath != null) {
            currentVideoUri = Uri.parse(videoPath)
            currentVideoTitle = videoTitle ?: "Video từ thư viện"
            playVideo(currentVideoUri!!, videoPosition)
        }
    }
}
```

```

71
72     private fun checkPermissions(): Boolean {
73         return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
74             ContextCompat.checkSelfPermission(context, Manifest.permission.READ_MEDIA_VIDEO) ==
75                 PackageManager.PERMISSION_GRANTED
76         } else {
77             ContextCompat.checkSelfPermission(context, Manifest.permission.READ_EXTERNAL_STORAGE) ==
78                 PackageManager.PERMISSION_GRANTED
79         }
80     }
81
82     private fun requestPermissions() {
83         val permissions = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
84             arrayOf(Manifest.permission.READ_MEDIA_VIDEO)
85         } else {
86             arrayOf(Manifest.permission.READ_EXTERNAL_STORAGE)
87         }
88
89         ActivityCompat.requestPermissions(activity, permissions, PERMISSION_REQUEST_CODE)
90     }
91
92     private fun pickVideoFromGallery() {
93         val intent = Intent(Intent.ACTION_PICK, MediaStore.Video.Media.EXTERNAL_CONTENT_URI)
94         startActivityForResult(intent, VIDEO_PICK_CODE)
95     }
96
97
98
99
100    private fun showUrlInputDialog() {
101        val dialogView = LayoutInflater.inflate(R.layout.dialog_url_input, root, false)
102        val etDialogUrl = dialogView.findViewById<EditText>(R.id.etDialogUrl)
103
104        // Thiết lập giá trị mặc định từ EditText chính
105        etDialogUrl.setText(etVideoUrl.text.toString())
106
107        AlertDialog.Builder(context, this)
108            .setView(dialogView)
109            .setPositiveButton("Phát") { _, _ ->
110                val url = etDialogUrl.text.toString().trim()
111                if (url.isNotEmpty()) {
112                    etVideoUrl.setText(url)
113                    playVideoFromUrl(url)
114                } else {
115                    Toast.makeText(context, "Vui lòng nhập URL", Toast.LENGTH_SHORT).show()
116                }
117            }
118            .setNegativeButton("Hủy", null)
119            .show()
120    }
121
122
123
124    private fun playVideoFromUrl(url: String) {
125        try {
126            currentVideoUri = Uri.parse(url)
127            currentVideoTitle = "Video từ URL"
128            playVideo(currentVideoUri!!, position: 0)
129        } catch (e: Exception) {
130            Toast.makeText(context, "Lỗi URL: ${e.message}", Toast.LENGTH_SHORT).show()
131        }
132    }
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148

```

```
148
149     private fun playVideo(uri: Uri, position: Int = 0) {
150         try {
151             videoView.setVideoURI(uri)
152             videoView.requestFocus()
153
154             // Thiết lập lại controller
155             videoView.setMediaController(mediaController)
156             mediaController.show()
157
158             // Cập nhật UI
159             tvCurrentVideo.text = currentVideoTitle
160
161             // Bắt đầu phát video
162             videoView.start()
163
164             // Đặt vị trí phát (nếu có)
165             if (position > 0) {
166                 videoView.seekTo(position)
167             }
168
169             // Hiển thị thông báo
170             Toast.makeText(context, text: "Đang phát: $currentVideoTitle", Toast.LENGTH_SHORT).show()
171         } catch (e: Exception) {
172             Toast.makeText(context, text: "Lỗi khi phát video: ${e.message}", Toast.LENGTH_SHORT).show()
173         }
174     }
175 }

5
6     override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
7         super.onActivityResult(requestCode, resultCode, data)
8
9         if (requestCode == VIDEO_PICK_CODE && resultCode == RESULT_OK && data != null) {
10            val selectedVideoUri = data.data
11            if (selectedVideoUri != null) {
12                // Lấy tên video
13                val cursor = contentResolver.query(
14                    selectedVideoUri,
15                    arrayOf(MediaStore.Video.Media.DISPLAY_NAME),
16                    selection: null,
17                    selectionArgs: null,
18                    sortOrder: null
19                )
20
21
22                currentVideoTitle = if (cursor != null && cursor.moveToFirst()) {
23                    val nameIndex = cursor.getColumnIndex(MediaStore.Video.Media.DISPLAY_NAME)
24                    val name = if (nameIndex >= 0) cursor.getString(nameIndex) else "Video từ thư viện"
25                    cursor.close()
26                    name
27                } else {
28                    "Video từ thư viện"
29                }
30
31                currentVideoUri = selectedVideoUri
32                playVideo(currentVideoUri!!)
33            }
34        }
35    }
36 }
```

```
205
206     @†    override fun onRequestPermissionsResult(
207         requestCode: Int,
208         permissions: Array<out String>,
209         grantResults: IntArray
210     ) {
211         super.onRequestPermissionsResult(requestCode, permissions, grantResults)
212
213         if (requestCode == PERMISSION_REQUEST_CODE) {
214             if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
215                 pickVideoFromGallery()
216             } else {
217                 Toast.makeText(
218                     context: this,
219                     text: "Cần cấp quyền để truy cập thư viện video",
220                     Toast.LENGTH_SHORT
221                 ).show()
222             }
223         }
224     }
225
226     @†    override fun onSaveInstanceState(outState: Bundle) {
227         super.onSaveInstanceState(outState)
228
229         // Lưu trạng thái video hiện tại
230         if (currentVideoUri != null) {
231             outState.putString("videoPath", currentVideoUri.toString())
232             outState.putInt("videoPosition", videoView.currentPosition)
233             outState.putString("videoTitle", currentVideoTitle)
234         }
235     }
236
237     @†    override fun onPause() {
238         super.onPause()
239         // Lưu vị trí phát hiện tại
240         if (videoView.isPlaying) {
241             videoView.pause()
242         }
243     }
244
245     @†    override fun onResume() {
246         super.onResume()
247         // Có thể tiếp tục phát nếu cần
248     }
249
250     @†    override fun onDestroy() {
251         super.onDestroy()
252         // Giải phóng tài nguyên
253         videoView.stopPlayback()
254     }
255 }
```

## BÀI TẬP 8: Ứng dụng đo gia tốc

### Mục tiêu:

- Sử dụng cảm biến gia tốc (TYPE\_ACCELEROMETER) để đo gia tốc của thiết bị theo ba trục x, y, z.
- Hiển thị giá trị gia tốc lên TextView.
- Hiển thị một hình ảnh động (ví dụ: một quả bóng) di chuyển theo hướng gia tốc.

### Mô tả:

Ứng dụng hiển thị các giá trị gia tốc đo được từ cảm biến gia tốc và mô phỏng sự di chuyển của một vật thể dựa trên các giá trị này.

### Các bước thực hiện:

#### 1. Lấy SensorManager và Sensor:

- Lấy SensorManager từ hệ thống.
- Sử dụng SensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER) để lấy đối tượng Sensor.

#### 2. Đăng ký SensorEventListener:

- Đăng ký một SensorEventListener để lắng nghe các sự kiện từ cảm biến gia tốc.
- Implement hai phương thức của SensorEventListener:
  - onSensorChanged(SensorEvent event): Xử lý các sự kiện cảm biến, lấy giá trị gia tốc từ event.values.
  - onAccuracyChanged(Sensor sensor, int accuracy): Xử lý khi độ chính xác của cảm biến thay đổi.

#### 3. Hiển thị giá trị gia tốc:

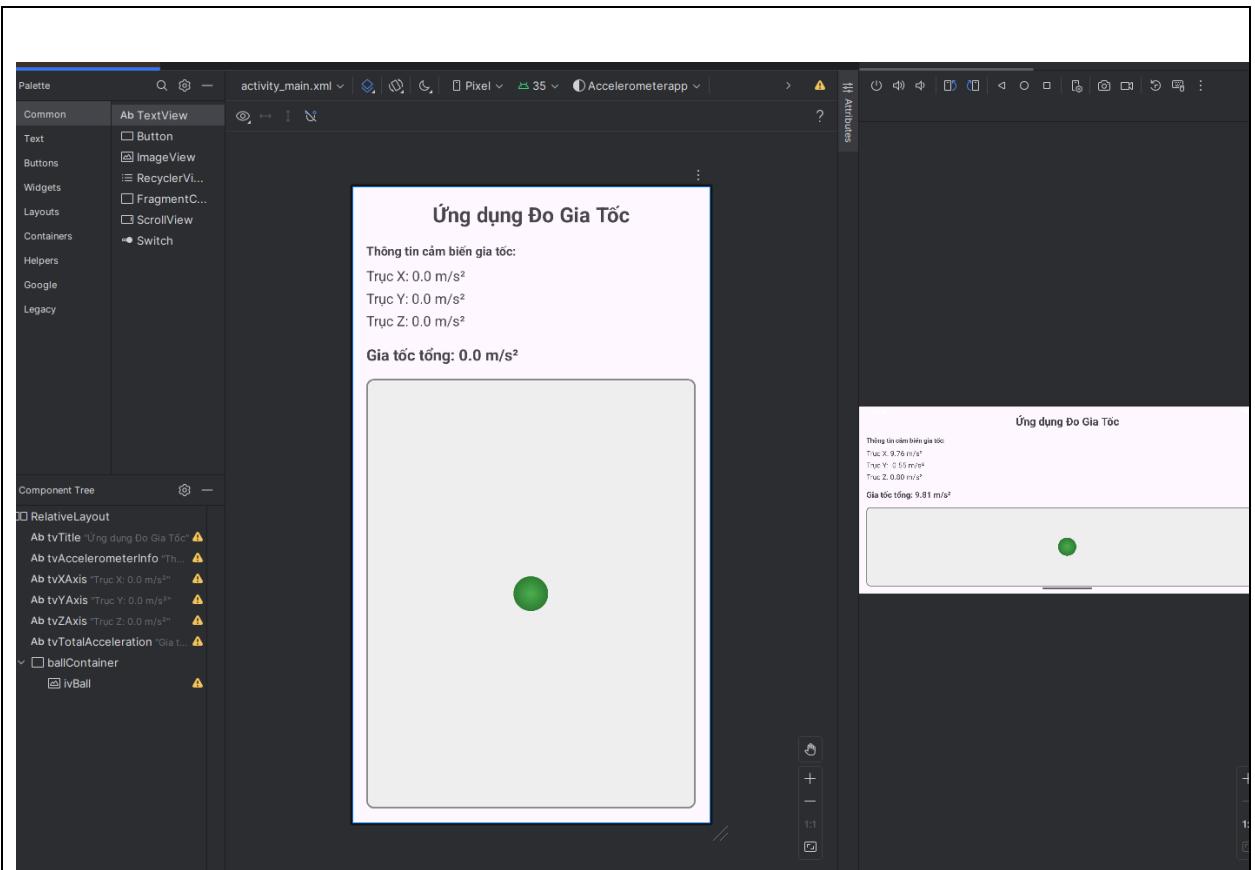
- Cập nhật các TextView để hiển thị giá trị gia tốc theo trục x, y, z.

#### 4. Mô phỏng chuyển động:

- Sử dụng một ImageView để hiển thị hình ảnh động.
- Trong phương thức onSensorChanged(), tính toán sự thay đổi vị trí của hình ảnh dựa trên giá trị gia tốc.
- Cập nhật vị trí của ImageView.

**Hướng dẫn Bài tập 08:** Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

Thiết kế giao diện



## Code file main thực hiện các logic

Lấy giá trị gia tốc theo 3 trục (x, y, z) từ event.values

Tính gia tốc tổng bằng công thức:  $\sqrt{x^2 + y^2 + z^2}$

```
14 import kotlin.math.sqrt
15
16 </> class MainActivity : AppCompatActivity(), SensorEventListener {
17
18     // Khai báo SensorManager và Sensor
19     private lateinit var sensorManager: SensorManager
20     private var accelerometer: Sensor? = null
21
22     // Khai báo các TextView để hiển thị giá trị gia tốc
23     private lateinit var tvXAxis: TextView
24     private lateinit var tvYAxis: TextView
25     private lateinit var tvZAxis: TextView
26     private lateinit var tvTotalAcceleration: TextView
27
28     // Khai báo các thành phần UI để hiển thị quả bóng
29     private lateinit var ballContainer: FrameLayout
30     private lateinit var ivBall: ImageView
31
32     // Các biến để tính toán vị trí của quả bóng
33     private var xPos = 0f
34     private var yPos = 0f
35     private var containerWidth = 0
36     private var containerHeight = 0
37     private var ballWidth = 0
38     private var ballHeight = 0
39
40     // Hệ số giảm chấn và độ nhạy
41     private val dampingFactor = 0.8f
42     private val sensitivity = 5f
```

```
16     class MainActivity : AppCompatActivity(), SensorEventListener {
44     @Override fun onCreate(savedInstanceState: Bundle?) {
45         super.onCreate(savedInstanceState)
46         setContentView(R.layout.activity_main)
47
48         // Khởi tạo các TextView
49         tvXAxis = findViewById(R.id.tvXAxis)
50         tvYAxis = findViewById(R.id.tvYAxis)
51         tvZAxis = findViewById(R.id.tvZAxis)
52         tvTotalAcceleration = findViewById(R.id.tvTotalAcceleration)
53
54         // Khởi tạo các thành phần UI cho quả bóng
55         ballContainer = findViewById(R.id.ballContainer)
56         ivBall = findViewById(R.id.ivBall)
57
58         // Lấy SensorManager và cảm biến gia tốc
59         sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
60         accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
61
62         // Kiểm tra xem thiết bị có cảm biến gia tốc không
63         if (accelerometer == null) {
64             Toast.makeText(context: this, text: "Thiết bị không có cảm biến gia tốc!", Toast.LENGTH_LONG).show()
65             finish()
66         }
67
68         // Thiết lập kích thước ban đầu cho quả bóng
69         ivBall.post {
70             ballWidth = ivBall.width
71             ballHeight = ivBall.height
72         }
73
74         // Thiết lập kích thước container
75         ballContainer.post {
76             containerWidth = ballContainer.width
77             containerHeight = ballContainer.height
78
79             // Đặt quả bóng ở giữa container
80             xPos = (containerWidth - ballWidth) / 2f
81             yPos = (containerHeight - ballHeight) / 2f
82             updateBallPosition()
83         }
84     }
}
```

```

85
86     override fun onResume() {
87         super.onResume()
88         // Đăng ký lắng nghe sự kiện cảm biến khi activity được resume
89         accelerometer?.let {
90             sensorManager.registerListener(listener: this, it, SensorManager.SENSOR_DELAY_GAME)
91         }
92     }
93
94     override fun onPause() {
95         super.onPause()
96         // Hủy đăng ký lắng nghe sự kiện cảm biến khi activity bị pause
97         sensorManager.unregisterListener(listener: this)
98     }
99
100    override fun onSensorChanged(event: SensorEvent) {
101        if (event.sensor.type == Sensor.TYPE_ACCELEROMETER) {
102            // Lấy giá trị giá tốc từ sự kiện cảm biến
103            val x = event.values[0]
104            val y = event.values[1]
105            val z = event.values[2]
106
107            // Tính giá tốc tổng
108            val totalAcceleration = sqrt(x*x + y*y + z*z)
109
110            // Cập nhật TextView để hiển thị giá trị giá tốc
111            tvXAxis.text = String.format("Trục X: %.2f m/s²", x)
112            tvYAxis.text = String.format("Trục Y: %.2f m/s²", y)
113            tvZAxis.text = String.format("Trục Z: %.2f m/s²", z)
114            tvTotalAcceleration.text = String.format("Gia tốc tổng: %.2f m/s²", totalAcceleration)
115
116            // Cập nhật vị trí của quả bóng dựa trên giá trị giá tốc
117            // Lưu ý: Chúng ta đảo ngược giá trị x và y để quả bóng di chuyển theo hướng nghiêng của thiết bị
118            updateBallPosition(x, y)
119        }
120    }
121
122    override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {
123        // Xử lý khi độ chính xác của cảm biến thay đổi (không cần thiết cho ứng dụng này)
124    }
125
126    private fun updateBallPosition(accelerationX: Float = 0f, accelerationY: Float = 0f) {
127        // Tính toán vị trí mới của quả bóng dựa trên giá tốc
128        // Nhân với sensitivity để điều chỉnh độ nhạy của chuyển động
129        // Nhân với dampingFactor để tạo hiệu ứng giảm chấn (làm cho chuyển động mượt mà hơn)
130        xPos -= accelerationX * sensitivity * dampingFactor
131        yPos += accelerationY * sensitivity * dampingFactor
132
133        // Giới hạn vị trí của quả bóng trong container
134        xPos = xPos.coerceIn(0f, (containerWidth - ballWidth).toFloat())
135        yPos = yPos.coerceIn(0f, (containerHeight - ballHeight).toFloat())
136
137        // Cập nhật vị trí của quả bóng trên giao diện
138        ivBall.translationX = xPos
139        ivBall.translationY = yPos
140    }
141
142

```

## BÀI TẬP 9: Ứng dụng la bàn

### Mục tiêu:

- Sử dụng cảm biến từ trường (TYPE\_MAGNETIC\_FIELD) và cảm biến gia tốc (TYPE\_ACCELEROMETER) để xác định hướng bắc.
- Hiển thị hướng bắc trên một ImageView (ví dụ: kim la bàn).
- Hiển thị góc lệch so với hướng bắc.

### Mô tả:

Ứng dụng hiển thị la bàn và hướng bắc dựa trên dữ liệu từ cảm biến từ trường và cảm biến gia tốc.

### Các bước thực hiện:

#### 1. Lấy SensorManager và Sensor:

- Lấy SensorManager từ hệ thống.
- Sử dụng SensorManager.getDefaultSensor(Sensor.TYPE\_MAGNETIC\_FIELD) để lấy đối tượng Sensor từ trường.
- Sử dụng SensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER) để lấy đối tượng Sensor gia tốc.

#### 2. Đăng ký SensorEventListener:

- Đăng ký một SensorEventListener để lắng nghe các sự kiện từ cảm biến từ trường và gia tốc.
- Trong phương thức onSensorChanged():
  - Lấy giá trị từ cả hai cảm biến.
  - Sử dụng các hàm SensorManager.getRotationMatrix() và SensorManager.getOrientation() để tính toán hướng bắc và góc lệch.

#### 3. Hiển thị la bàn:

- Sử dụng một ImageView để hiển thị hình ảnh la bàn.
- Sử dụng phương thức ImageView.setRotation() để xoay hình ảnh la bàn theo hướng bắc.

#### 4. Hiển thị góc lệch:

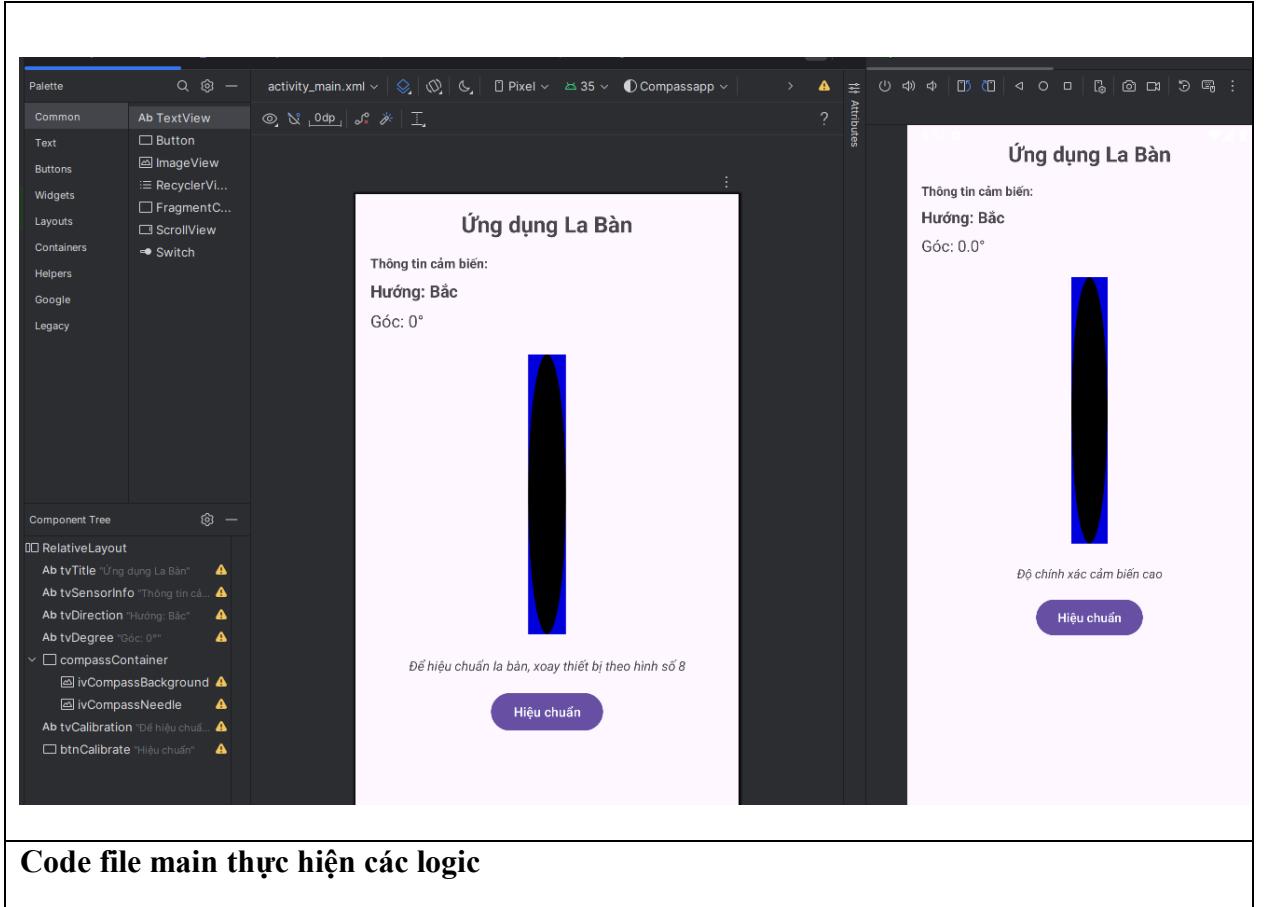
- Hiển thị giá trị góc lệch so với hướng bắc lên một TextView.

### Lưu ý:

- **Quyền (Permissions):** Khai báo các quyền cần thiết trong AndroidManifest.xml, bao gồm quyền sử dụng cảm biến.
- **Độ chính xác:** Độ chính xác của cảm biến có thể bị ảnh hưởng bởi nhiều từ môi trường. Cần có các biện pháp lọc dữ liệu hoặc hiệu chỉnh để cải thiện độ chính xác.
- **Tiết kiệm pin:** Hủy đăng ký SensorEventListener khi không sử dụng cảm biến để tiết kiệm pin.
- **Kiểm tra cảm biến:** Kiểm tra xem thiết bị có hỗ trợ các cảm biến cần thiết hay không trước khi sử dụng.

**Hướng dẫn Bài tập 09:** Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

Thiết kế giao diện



Code file main thực hiện các logic

```
17  
18 </> class MainActivity : AppCompatActivity(), SensorEventListener {  
19  
20     // Khai báo SensorManager và các Sensor  
21     private lateinit var sensorManager: SensorManager  
22     private var accelerometer: Sensor? = null  
23     private var magnetometer: Sensor? = null  
24  
25     // Khai báo các TextView để hiển thị thông tin  
26     private lateinit var tvDirection: TextView  
27     private lateinit var tvDegree: TextView  
28     private lateinit var tvCalibration: TextView  
29  
30     // Khai báo ImageView để hiển thị la bàn  
31     private lateinit var ivCompassNeedle: ImageView  
32  
33     // Khai báo Button hiệu chuẩn  
34     private lateinit var btnCalibrate: Button  
35  
36     // Mảng lưu trữ giá trị cảm biến  
37     private val accelerometerReading = FloatArray(size: 3)  
38     private val magnetometerReading = FloatArray(size: 3)  
39  
40     // Mảng ma trận xoay và ma trận định hướng  
41     private val rotationMatrix = FloatArray(size: 9)  
42     private val orientationAngles = FloatArray(size: 3)  
43  
44     // Biến lưu góc hiện tại  
45     private var currentDegree = 0f  
46  
47     // Biến kiểm tra trạng thái cảm biến  
48     private var isSensorAvailable = true  
49     private var isCalibrating = false  
50
```

```

51     override fun onCreate(savedInstanceState: Bundle?) {
52         super.onCreate(savedInstanceState)
53         setContentView(R.layout.activity_main)
54
55         // Khởi tạo các TextView
56         tvDirection = findViewById(R.id.tvDirection)
57         tvDegree = findViewById(R.id.tvDegree)
58         tvCalibration = findViewById(R.id.tvCalibration)
59
60         // Khởi tạo ImageView
61         ivCompassNeedle = findViewById(R.id.ivCompassNeedle)
62
63         // Khởi tạo Button
64         btnCalibrate = findViewById(R.id.btnCalibrate)
65
66         // Lấy SensorManager và các cảm biến
67         sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
68         accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
69         magnetometer = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD)
70
71         // Kiểm tra xem thiết bị có cảm biến cần thiết không
72         if (accelerometer == null || magnetometer == null) {
73             isSensorAvailable = false
74             Toast.makeText(context, text: "Thiết bị không có cảm biến cần thiết cho la bàn!", Toast.LENGTH_LONG).show()
75             tvCalibration.text = "Thiết bị không hỗ trợ la bàn"
76             btnCalibrate.isEnabled = false
77         }
78
79         // Thiết lập sự kiện cho nút hiệu chuẩn
80         btnCalibrate.setOnClickListener {
81             if (isSensorAvailable) {
82                 isCalibrating = true
83                 tvCalibration.text = "Đang hiệu chuẩn... Xoay thiết bị theo hình số 8"
84                 Toast.makeText(context, text: "Đang hiệu chuẩn la bàn...", Toast.LENGTH_SHORT).show()
85
86                 // Sau 5 giây, kết thúc hiệu chuẩn
87                 ivCompassNeedle.postDelayed({
88                     isCalibrating = false
89                     tvCalibration.text = "Hiệu chuẩn hoàn tất"
90                     Toast.makeText(context, text: "Hiệu chuẩn hoàn tất!", Toast.LENGTH_SHORT).show()
91                 }, delayMillis: 5000)

```

```
90         }
91     }
92 }
93 }
94 }
95
96 @Override fun onResume() {
97     super.onResume()
98
99     // Đăng ký lắng nghe sự kiện cảm biến khi activity được resume
100    if (isSensorAvailable) {
101        accelerometer?.let {
102            sensorManager.registerListener(
103                listener: this,
104                it,
105                SensorManager.SENSOR_DELAY_GAME
106            )
107        }
108
109        magnetometer?.let {
110            sensorManager.registerListener(
111                listener: this,
112                it,
113                SensorManager.SENSOR_DELAY_GAME
114            )
115        }
116    }
117 }
118
119 @Override fun onPause() {
```

```
118
119     override fun onPause() {
120         super.onPause()
121
122         // Hủy đăng ký lắng nghe sự kiện cảm biến khi activity bị pause
123         sensorManager.unregisterListener(listener: this)
124     }
125
126     override fun onSensorChanged(event: SensorEvent) {
127         // Xử lý dữ liệu từ cảm biến
128         when (event.sensor.type) {
129             Sensor.TYPE_ACCELEROMETER -> {
130                 System.arraycopy(event.values, srcPos: 0, accelerometerReading, destPos: 0, accelerometerReading.size)
131             }
132             Sensor.TYPE_MAGNETIC_FIELD -> {
133                 System.arraycopy(event.values, srcPos: 0, magnetometerReading, destPos: 0, magnetometerReading.size)
134             }
135         }
136
137         // Cập nhật hướng la bàn
138         updateOrientationAngles()
139     }
140
141     override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {
142         // Xử lý khi độ chính xác của cảm biến thay đổi
143         if (sensor.type == Sensor.TYPE_MAGNETIC_FIELD) {
144             when (accuracy) {
145                 SensorManager.SENSOR_STATUS_UNRELIABLE -> {
146                     tvCalibration.text = "Cảm biến không đáng tin cậy, cần hiệu chuẩn"
147                 }
148                 SensorManager.SENSOR_STATUS_ACCURACY_LOW -> {
149                     tvCalibration.text = "Độ chính xác cảm biến thấp, cần hiệu chuẩn"
150                 }
151                 SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM -> {
152                     tvCalibration.text = "Độ chính xác cảm biến trung bình"
153                 }
154                 SensorManager.SENSOR_STATUS_ACCURACY_HIGH -> {
155                     tvCalibration.text = "Độ chính xác cảm biến cao"
156                 }
157             }
158         }
159     }
160 }
```

```
160
161     private fun updateOrientationAngles() {
162         // Cập nhật ma trận xoay từ dữ liệu cảm biến
163         SensorManager.getRotationMatrix(
164             rotationMatrix,
165             null,
166             accelerometerReading,
167             magnetometerReading
168         )
169
170         // Lấy góc định hướng từ ma trận xoay
171         SensorManager.getOrientation(rotationMatrix, orientationAngles)
172
173         // Chuyển đổi góc từ radian sang độ và lấy góc azimuth (hướng bắc)
174         // Azimuth: góc giữa hướng bắc từ tinh và hướng thiết bị
175         val azimuthInRadians = orientationAngles[0]
176         val azimuthInDegrees = (toDegrees(azimuthInRadians.toDouble()) + 360) % 360
177
178         // Cập nhật TextView hiển thị góc
179         tvDegree.text = String.format("Góc: %.1f°", azimuthInDegrees)
180
181         // Cập nhật TextView hiển thị hướng
182         val direction = getDirectionFromDegree(azimuthInDegrees.toFloat())
183         tvDirection.text = "Hướng: $direction"
184
185         // Tạo hiệu ứng xoay cho kim la bàn
186         val rotateAnimation = RotateAnimation(
187             currentDegree,
188             -azimuthInDegrees.toFloat(),
189             Animation.RELATIVE_TO_SELF, pivotXValue: 0.5f,
190             Animation.RELATIVE_TO_SELF, pivotYValue: 0.5f
191         )
192
193         // Thiết lập thời gian và kiểu nội suy cho hiệu ứng
194         rotateAnimation.duration = 250
195         rotateAnimation.fillAfter = true
196
197         // Áp dụng hiệu ứng xoay cho kim la bàn
198         ivCompassNeedle.startAnimation(rotateAnimation)
```

```
191     )
192
193     // Thiết lập thời gian và kiểu nội suy cho hiệu ứng
194     rotateAnimation.duration = 250
195     rotateAnimation.fillAfter = true
196
197     // Áp dụng hiệu ứng xoay cho kim la bàn
198     ivCompassNeedle.startAnimation(rotateAnimation)
199
200     // Cập nhật góc hiện tại
201     currentDegree = -azimuthInDegrees.toFloat()
202 }
203
204     private fun getDirectionFromDegree(degree: Float): String {
205         return when {
206             degree >= 337.5 || degree < 22.5 -> "Bắc"
207             degree >= 22.5 && degree < 67.5 -> "Đông Bắc"
208             degree >= 67.5 && degree < 112.5 -> "Đông"
209             degree >= 112.5 && degree < 157.5 -> "Đông Nam"
210             degree >= 157.5 && degree < 202.5 -> "Nam"
211             degree >= 202.5 && degree < 247.5 -> "Tây Nam"
212             degree >= 247.5 && degree < 292.5 -> "Tây"
213             degree >= 292.5 && degree < 337.5 -> "Tây Bắc"
214             else -> "Không xác định"
215         }
216     }
217 }
218 }
```

## BÀI TẬP 10: ỨNG DỤNG CLIENT-SERVER ĐƠN GIẢN SỬ DỤNG TCP

### Mục tiêu:

- Xây dựng một ứng dụng Android (Client) có thể gửi và nhận dữ liệu từ một ứng dụng Server (có thể chạy trên PC hoặc thiết bị Android khác) sử dụng giao thức TCP.
- Ứng dụng Client cho phép người dùng nhập tin nhắn và gửi đến Server.
- Ứng dụng Server nhận tin nhắn và hiển thị chúng.

### Mô tả:

Ứng dụng này minh họa giao tiếp cơ bản giữa Client và Server sử dụng TCP, tập trung vào việc thiết lập kết nối, gửi và nhận dữ liệu.

### Các bước thực hiện:

#### Phía Server:

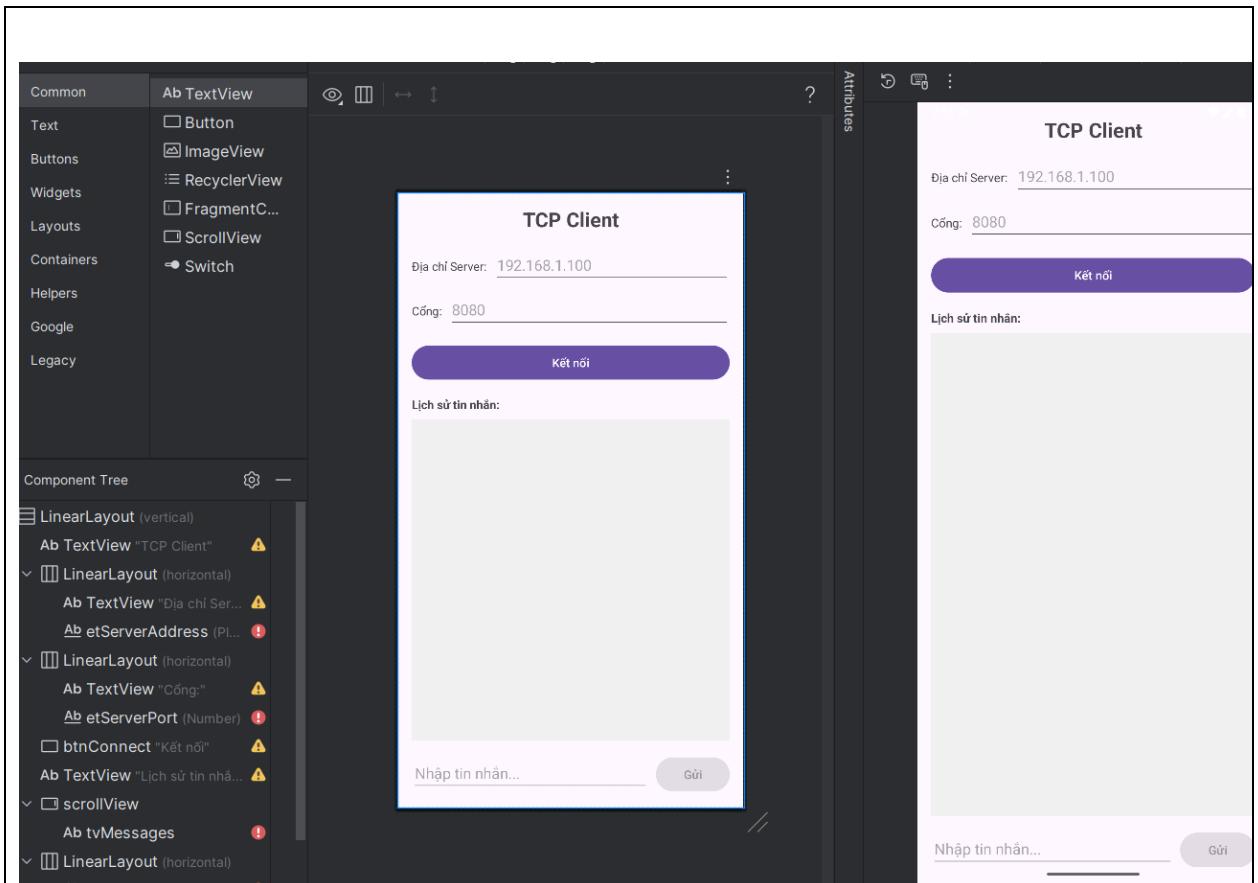
1. **Tạo ServerSocket:** Sử dụng `ServerSocket` để lắng nghe kết nối đến trên một cổng cụ thể.
2. **Chấp nhận kết nối:** Sử dụng `ServerSocket.accept()` để chấp nhận kết nối từ Client.
3. **Tạo luồng (Thread):** Tạo một luồng mới để xử lý giao tiếp với mỗi Client.
4. **Giao tiếp:** Sử dụng `InputStream` và `OutputStream` của `Socket` để nhận và gửi dữ liệu.
5. **Đóng kết nối:** Đóng `Socket` và `ServerSocket` khi hoàn tất giao tiếp.

#### Phía Client (Android):

1. **Tạo Socket:** Sử dụng `Socket` để kết nối đến Server trên địa chỉ IP và cổng cụ thể.
2. **Giao tiếp:** Sử dụng `InputStream` và `OutputStream` của `Socket` để gửi và nhận dữ liệu.
3. **Gửi dữ liệu:** Lấy dữ liệu từ người dùng (ví dụ: một `EditText`) và gửi đến Server.
4. **Nhận dữ liệu:** Nhận phản hồi từ Server và hiển thị cho người dùng.
5. **Đóng kết nối:** Đóng `Socket` khi hoàn tất giao tiếp.

**Hướng dẫn Bài tập 10:** Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

Thiết kế giao diện



### Code file main thực hiện các logic

```
class MainActivity : AppCompatActivity() {

    private val TAG = "TCPClient"

    private lateinit var etServerAddress: EditText
    private lateinit var etServerPort: EditText
    private lateinit var btnConnect: Button
    private lateinit var tvMessages: TextView
    private lateinit var etMessage: EditText
    private lateinit var btnSend: Button
    private lateinit var scrollView: ScrollView

    private var clientSocket: Socket? = null
    private var reader: BufferedReader? = null
```

```

private var writer: BufferedWriter? = null
private var isConnected = false

private val executorService = Executors.newSingleThreadExecutor()
private val handler = Handler(Looper.getMainLooper())

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Khởi tạo các thành phần UI
    etServerAddress = findViewById(R.id.etServerAddress)
    etServerPort = findViewById(R.id.etServerPort)
    btnConnect = findViewById(R.id.btnConnect)
    tvMessages = findViewById(R.id.tvMessages)
    etMessage = findViewById(R.id.etMessage)
    btnSend = findViewById(R.id.btnSend)
    scrollView = findViewById(R.id.scrollView)

    // Thiết lập sự kiện cho nút Kết nối
    btnConnect.setOnClickListener {
        val serverAddress = etServerAddress.text.toString().trim()
        val serverPortStr = etServerPort.text.toString().trim()

        if (serverAddress.isEmpty() || serverPortStr.isEmpty()) {
            Toast.makeText(this, "Vui lòng nhập địa chỉ và cổng",
                    Toast.LENGTH_SHORT).show()
        }
    }
}

```

```
        return@setOnClickListener

    }

    val serverPort = serverPortStr.toIntOrNull()

    if (serverPort == null || serverPort <= 0 || serverPort > 65535) {

        Toast.makeText(this, "Cổng không hợp lệ", Toast.LENGTH_SHORT).show()

        return@setOnClickListener
    }

    if (isConnected) {

        disconnectFromServer()

        btnConnect.text = "Kết nối"

        etMessage.isEnabled = false

        btnSend.isEnabled = false

    } else {

        connectToServer(serverAddress, serverPort)
    }
}

// Thiết lập sự kiện cho nút Gửi

btnSend.setOnClickListener {
    sendMessageFromUI()
}

// Thêm sự kiện khi nhấn Enter trên bàn phím

etMessage.setOnEditorActionListener { _, actionBarId, _ ->

    if (actionId == EditorInfo.IME_ACTION_SEND) {
```

```

sendMessageFromUI()

    return@setOnEditorActionListener true

}

false

}

}

private fun sendMessageFromUI() {

    val message = etMessage.text.toString().trim()

    if (message.isNotEmpty()) {

        Log.d(TAG, "Đang gửi tin nhắn từ UI: \"$message\"")

        // Kiểm tra kết nối trước khi gửi

        synchronized(this) {

            if (!isConnected || clientSocket == null || clientSocket!!.isClosed) {

                Log.e(TAG, "Không thể gửi tin nhắn: Kết nối không hợp lệ")

                Toast.makeText(this, "Không thể gửi tin nhắn: Kết nối đã mất",
                    Toast.LENGTH_SHORT).show()

                // Đảm bảo UI được cập nhật

                handler.post {

                    disconnectFromServer()

                    btnConnect.text = "Kết nối"

                    etMessage.isEnabled = false

                    btnSend.isEnabled = false

                }

            }

        }

    }

}

```

```

    }

    // Nếu kết nối vẫn ổn, gửi tin nhắn
    sendMessage(message)
    etMessage.text.clear()
}

} else {
    Toast.makeText(this, "Vui lòng nhập tin nhắn", Toast.LENGTH_SHORT).show()
}

}

private fun connectToServer(serverAddress: String, serverPort: Int) {
    addMessage("Đang kết nối đến $serverAddress:$serverPort...")

    executorService.execute {
        try {
            Log.d(TAG, "Đang thử kết nối đến $serverAddress:$serverPort")

            // Tạo socket với timeout
            clientSocket = Socket()
            Log.d(TAG, "Đã tạo đối tượng socket")

            // Thiết lập timeout kết nối
            clientSocket!!.connect(InetSocketAddress(serverAddress, serverPort), 5000)
            Log.d(TAG, "Đã kết nối socket thành công")
        }
    }
}

```

```

// Thiết lập timeout đọc

clientSocket!!.soTimeout = 10000

Log.d(TAG, "Đã thiết lập timeout đọc: 10000ms")

// Tạo reader và writer với UTF-8

reader    =    BufferedReader(InputStreamReader(clientSocket!!.getInputStream(),
StandardCharsets.UTF_8))

writer    =    BufferedWriter(OutputStreamWriter(clientSocket!!.getOutputStream(),
StandardCharsets.UTF_8))

Log.d(TAG, "Đã tạo reader và writer thành công")

isConnected = true

// Cập nhật UI khi kết nối thành công

handler.post {

    addMessage("Đã kết nối thành công!")

    btnConnect.text = "Ngắt kết nối"

    etMessage.isEnabled = true

    btnSend.isEnabled = true

}

// Bắt đầu nhận tin nhắn từ server

receiveMessages()

} catch (e: Exception) {

    Log.e(TAG, "Lỗi kết nối: ${e.message}", e)
}

```

```

handler.post {

    addMessage("Lỗi kết nối: ${e.message}")

    disconnectFromServer()

}

}

}

}

private fun receiveMessages() {

try {

    Log.d(TAG, "Bắt đầu lắng nghe tin nhắn từ server")

    while (isConnected && clientSocket != null && !clientSocket!!.isClosed) {

        try {

            Log.d(TAG, "Đang đợi tin nhắn...")

            val message = reader?.readLine()

            Log.d(TAG, "Đã nhận tin nhắn: \'$message\'")



if (message != null) {

    handler.post {

        addMessage("Server: $message")

    }

} else {

    // Nếu message là null, server đã đóng kết nối

    Log.d(TAG, "Nhận được null, server có thể đã đóng kết nối")

    break

}

} catch (e: SocketTimeoutException) {

```

```

// Timeout là bình thường, chỉ log ở mức debug

Log.d(TAG, "Timeout khi đọc, tiếp tục chờ")

// Gửi một gói "ping" đến server để kiểm tra kết nối

// Điều này có thể ngăn timeout nếu server vẫn hoạt động

try {

    if (isConnected && writer != null) {

        writer?.write("PING")

        writer?.newLine()

        writer?.flush()

        Log.d(TAG, "Đã gửi PING đến server")

    }

} catch (pingEx: Exception) {

    Log.e(TAG, "Lỗi gửi PING: ${pingEx.message}")

    break // Kết thúc vòng lặp nếu không thể gửi PING

}

continue // Tiếp tục vòng lặp

} catch (e: Exception) {

    Log.e(TAG, "Lỗi khi đọc tin nhắn: ${e.message}", e)

    break

}

}

}

} catch (e: Exception) {

    Log.e(TAG, "Lỗi nhận tin nhắn: ${e.message}", e)

}

} finally {

    Log.d(TAG, "Kết thúc nhận tin nhắn")

```

```

// Đảm bảo đánh dấu kết nối đã mất

val wasConnected = isConnected
isConnected = false

// Chỉ ngắt kết nối từ main thread

handler.post {

    if (wasConnected) {

        disconnectFromServer()

        btnConnect.text = "Kết nối"

        etMessage.isEnabled = false

        btnSend.isEnabled = false

    }

}

}

}

}

private fun sendMessage(message: String) {

    executorService.execute {

        try {

            if (writer == null) {

                Log.e(TAG, "Lỗi: writer là null")

                handler.post {

                    Toast.makeText(this, "Lỗi: Không thể gửi tin nhắn - writer null",
                    Toast.LENGTH_SHORT).show()

                }

            }

        }

    }

}

```

```
        return@execute

    }

    if (clientSocket == null || clientSocket!!.isClosed) {
        Log.e(TAG, "Lỗi: Socket đã đóng hoặc null")
        handler.post {
            Toast.makeText(this, "Lỗi: Socket đã đóng", Toast.LENGTH_SHORT).show()
            disconnectFromServer()
            btnConnect.text = "Kết nối"
            etMessage.isEnabled = false
            btnSend.isEnabled = false
            isConnected = false
        }
        return@execute
    }

    Log.d(TAG, "Chuẩn bị gửi tin nhắn: \"$message\"")
    writer?.write(message)
    Log.d(TAG, "Đã gọi writer.write()")
    writer?.newLine()
    Log.d(TAG, "Đã gọi writer.newLine()")
    writer?.flush()
    Log.d(TAG, "Đã gọi writer.flush()")

    handler.post {
        addMessage("Bạn: $message")
        Log.d(TAG, "Đã cập nhật UI với tin nhắn đã gửi")
    }
}
```

```
    }

} catch (e: Exception) {

    Log.e(TAG, "Lỗi gửi tin nhắn: ${e.message}", e)

    handler.post {

        addMessage("Lỗi gửi tin nhắn: ${e.message}")

        Toast.makeText(this, "Lỗi gửi tin nhắn: ${e.message}", Toast.LENGTH_SHORT).show()

        // Nếu có lỗi gửi, giả định kết nối đã mất
        disconnectFromServer()

        btnConnect.text = "Kết nối"

        etMessage.isEnabled = false

        btnSend.isEnabled = false

        isConnected = false

    }

}

}

}

private fun disconnectFromServer() {

    try {

        Log.d(TAG, "Đang ngắt kết nối từ server")

        isConnected = false

        reader?.close()

        writer?.close()

    }

}
```

```
clientSocket?.close()

Log.d(TAG, "Đã ngắt kết nối thành công")

} catch (e: Exception) {

    Log.e(TAG, "Lỗi khi ngắt kết nối: ${e.message}", e)

    addMessage("Lỗi khi ngắt kết nối: ${e.message}")

} finally {

    reader = null

    writer = null

    clientSocket = null

}

}

private fun addMessage(message: String) {

    Log.d(TAG, "Thêm tin nhắn: $message")

    tvMessages.append("$message\n\n")

    // Cuộn xuống để hiển thị tin nhắn mới nhất

    scrollView.post {

        scrollView.fullScroll(ScrollView.FOCUS_DOWN)

    }

}

override fun onDestroy() {

    super.onDestroy()

    Log.d(TAG, "onDestroy được gọi, đang dọn dẹp tài nguyên")

    disconnectFromServer()

    executorService.shutdown()
```

```
    }  
}
```

## BÀI TẬP 11: Ứng dụng gửi và nhận tin nhắn sử dụng UDP

### Mục tiêu:

- Xây dựng một ứng dụng Android có thể gửi và nhận tin nhắn sử dụng giao thức UDP.
- Ứng dụng cho phép người dùng gửi tin nhắn đến một thiết bị khác trên mạng.
- Ứng dụng có thể nhận tin nhắn từ các thiết bị khác.

### Mô tả:

Ứng dụng này minh họa giao tiếp sử dụng UDP, tập trung vào việc gửi và nhận các gói tin độc lập.

### Các bước thực hiện:

#### Gửi tin nhắn:

1. **Lấy dữ liệu:** Lấy dữ liệu từ người dùng (ví dụ: một EditText).
2. **Tạo DatagramPacket:** Tạo một DatagramPacket chứa dữ liệu cần gửi, địa chỉ IP và cổng của người nhận.
3. **Tạo DatagramSocket:** Tạo một DatagramSocket để gửi gói tin.
4. **Gửi gói tin:** Sử dụng DatagramSocket.send() để gửi DatagramPacket.
5. **Đóng DatagramSocket:** Đóng DatagramSocket sau khi gửi.

#### Nhận tin nhắn:

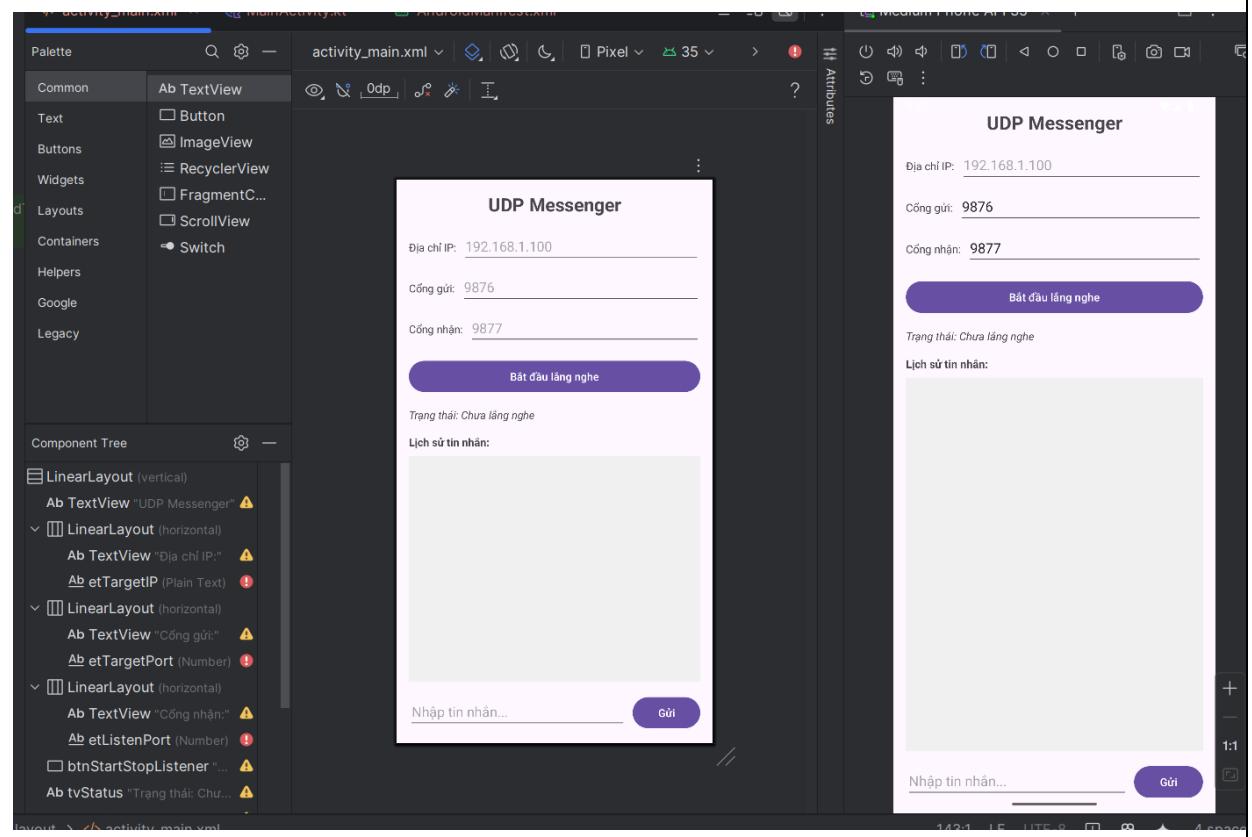
1. **Tạo DatagramSocket:** Tạo một DatagramSocket và lắng nghe trên một cổng cụ thể.
2. **Tạo DatagramPacket:** Tạo một DatagramPacket để chứa dữ liệu nhận được.
3. **Nhận gói tin:** Sử dụng DatagramSocket.receive() để nhận DatagramPacket.
4. **Xử lý dữ liệu:** Trích xuất dữ liệu từ DatagramPacket và hiển thị.
5. **Đóng DatagramSocket:** Đóng DatagramSocket khi không còn cần nhận tin nhắn.

### Lưu ý:

- **Quyền (Permissions):** Đảm bảo khai báo các quyền cần thiết trong `AndroidManifest.xml`, bao gồm `android.permission.INTERNET`.
- **Luồng (Threads):** Thực hiện các hoạt động mạng trong các luồng riêng để tránh làm treo UI thread.
- **Xử lý lỗi:** Xử lý các trường hợp lỗi có thể xảy ra, chẳng hạn như kết nối không thành công, mất kết nối, v.v.
- **Giao thức:** Xác định rõ giao thức giao tiếp giữa Client và Server (ví dụ: định dạng tin nhắn, các lệnh).

**Hướng dẫn Bài tập 11:** Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo :D

### Thiết kế giao diện



### Code file main thực hiện các logic

```
--> 23 </> class MainActivity : AppCompatActivity() {  
24  
25     private val TAG = "UDPMessenger"  
26  
27     // UI components  
28     private lateinit var etTargetIP: EditText  
29     private lateinit var etTargetPort: EditText  
30     private lateinit var etListenPort: EditText  
31     private lateinit var btnStartStopListener: Button  
32     private lateinit var tvStatus: TextView  
33     private lateinit var tvMessages: TextView  
34     private lateinit var scrollView: ScrollView  
35     private lateinit var etMessage: EditText  
36     private lateinit var btnSend: Button  
37  
38     // Network components  
39     private var listenerSocket: DatagramSocket? = null  
40     private var isListening = false  
41     private var listenerThread: Thread? = null  
42  
43     // Thread management  
44     private val executorService: ExecutorService = Executors.newCachedThreadPool()  
45     private val handler = Handler(Looper.getMainLooper())  
46  
47     // Date formatter for timestamps  
48     private val dateFormat = SimpleDateFormat(pattern: "HH:mm:ss", Locale.getDefault())  
49  
50     @Override fun onCreate(savedInstanceState: Bundle?) {  
51         super.onCreate(savedInstanceState)  
52         setContentView(R.layout.activity_main)
```

```
29  class MainActivity : AppCompatActivity() {  
30      @Override  
31      protected void onCreate(Bundle savedInstanceState) {  
32          super.onCreate(savedInstanceState)  
33          setContentView(R.layout.activity_main)  
34  
35          // Initialize UI components  
36          etTargetIP = findViewById(R.id.etTargetIP)  
37          etTargetPort = findViewById(R.id.etTargetPort)  
38          etListenPort = findViewById(R.id.etListenPort)  
39          btnStartStopListener = findViewById(R.id.btnStartStopListener)  
40          tvStatus = findViewById(R.id.tvStatus)  
41          tvMessages = findViewById(R.id.tvMessages)  
42          scrollView = findViewById(R.id.scrollView)  
43          etMessage = findViewById(R.id.etMessage)  
44          btnSend = findViewById(R.id.btnSend)  
45  
46          // Set default values  
47          etTargetPort.setText("9876")  
48          etListenPort.setText("9877")  
49  
50          // Set up button click listeners  
51          btnStartStopListener.setOnClickListener {  
52              if (isListening) {  
53                  stopListener()  
54              } else {  
55                  startListener()  
56              }  
57          }  
58  
59          btnSend.setOnClickListener {  
60              sendMessage()  
61          }  
62  
63      }  
64  
65      // Set default values  
66      etTargetPort.setText("9876")  
67      etListenPort.setText("9877")  
68  
69      // Set up button click listeners  
70      btnStartStopListener.setOnClickListener {  
71          if (isListening) {  
72              stopListener()  
73          } else {  
74              startListener()  
75          }  
76      }  
77  
78      btnSend.setOnClickListener {  
79          sendMessage()  
80      }  
81  }  
com.example.udpmessenger.MainActivity
```

```
81
82         // Set up enter key listener for message input
83         etMessage.setOnEditorActionListener { _, actionId, _ ->
84             if (actionId == EditorInfo.IME_ACTION_SEND) {
85                 sendMessage()
86                 return@setOnEditorActionListener true
87             }
88             false
89         }
90     }
91
92     private fun startListener() {
93         val portStr = etListenPort.text.toString().trim()
94         if (portStr.isEmpty()) {
95             Toast.makeText(context, text: "Vui lòng nhập cổng lắng nghe", Toast.LENGTH_SHORT)
96             return
97         }
98
99         val port = portStr.toIntOrNull()
100        if (port == null || port <= 0 || port > 65535) {
101            Toast.makeText(context, text: "Cổng không hợp lệ", Toast.LENGTH_SHORT)
102            return
103        }
104
105        executorService.execute {
106            try {
107                // Create a new DatagramSocket to listen on the specified port
108                listenerSocket = DatagramSocket(port)
109                isListening = true
110            }
111        }
112    }
113
114    private fun sendMessage() {
115        val message = etMessage.text.toString()
116        if (message.isEmpty()) {
117            return
118        }
119
120        val bytes = message.toByteArray()
121        val packet = DatagramPacket(bytes, bytes.size)
122        listenerSocket.send(packet)
123    }
124
125    private fun stopListening() {
126        isListening = false
127        listenerSocket.close()
128    }
129
130    private fun onReceive(packet: DatagramPacket) {
131        val bytes = packet.data
132        val message = String(bytes)
133        etMessage.append(message)
134    }
135
136    private fun onConnectionEstablished() {
137        etListenPort.isEnabled = true
138        etMessage.isEnabled = true
139    }
140
141    private fun onConnectionLost() {
142        etListenPort.isEnabled = false
143        etMessage.isEnabled = false
144    }
145
146    private fun onPortChanged() {
147        startListener()
148    }
149
150    private fun onMessageReceived(message: String) {
151        etMessage.append(message)
152    }
153
154    private fun onConnectionEstablished() {
155        etListenPort.isEnabled = true
156        etMessage.isEnabled = true
157    }
158
159    private fun onConnectionLost() {
160        etListenPort.isEnabled = false
161        etMessage.isEnabled = false
162    }
163
164    private fun onPortChanged() {
165        startListener()
166    }
167
168    private fun onMessageReceived(message: String) {
169        etMessage.append(message)
170    }
171
172    private fun onConnectionEstablished() {
173        etListenPort.isEnabled = true
174        etMessage.isEnabled = true
175    }
176
177    private fun onConnectionLost() {
178        etListenPort.isEnabled = false
179        etMessage.isEnabled = false
180    }
181
182    private fun onPortChanged() {
183        startListener()
184    }
185
186    private fun onMessageReceived(message: String) {
187        etMessage.append(message)
188    }
189
190    private fun onConnectionEstablished() {
191        etListenPort.isEnabled = true
192        etMessage.isEnabled = true
193    }
194
195    private fun onConnectionLost() {
196        etListenPort.isEnabled = false
197        etMessage.isEnabled = false
198    }
199
200    private fun onPortChanged() {
201        startListener()
202    }
203
204    private fun onMessageReceived(message: String) {
205        etMessage.append(message)
206    }
207
208    private fun onConnectionEstablished() {
209        etListenPort.isEnabled = true
210        etMessage.isEnabled = true
211    }
212
213    private fun onConnectionLost() {
214        etListenPort.isEnabled = false
215        etMessage.isEnabled = false
216    }
217
218    private fun onPortChanged() {
219        startListener()
220    }
221
222    private fun onMessageReceived(message: String) {
223        etMessage.append(message)
224    }
225
226    private fun onConnectionEstablished() {
227        etListenPort.isEnabled = true
228        etMessage.isEnabled = true
229    }
230
231    private fun onConnectionLost() {
232        etListenPort.isEnabled = false
233        etMessage.isEnabled = false
234    }
235
236    private fun onPortChanged() {
237        startListener()
238    }
239
240    private fun onMessageReceived(message: String) {
241        etMessage.append(message)
242    }
243
244    private fun onConnectionEstablished() {
245        etListenPort.isEnabled = true
246        etMessage.isEnabled = true
247    }
248
249    private fun onConnectionLost() {
250        etListenPort.isEnabled = false
251        etMessage.isEnabled = false
252    }
253
254    private fun onPortChanged() {
255        startListener()
256    }
257
258    private fun onMessageReceived(message: String) {
259        etMessage.append(message)
260    }
261
262    private fun onConnectionEstablished() {
263        etListenPort.isEnabled = true
264        etMessage.isEnabled = true
265    }
266
267    private fun onConnectionLost() {
268        etListenPort.isEnabled = false
269        etMessage.isEnabled = false
270    }
271
272    private fun onPortChanged() {
273        startListener()
274    }
275
276    private fun onMessageReceived(message: String) {
277        etMessage.append(message)
278    }
279
280    private fun onConnectionEstablished() {
281        etListenPort.isEnabled = true
282        etMessage.isEnabled = true
283    }
284
285    private fun onConnectionLost() {
286        etListenPort.isEnabled = false
287        etMessage.isEnabled = false
288    }
289
290    private fun onPortChanged() {
291        startListener()
292    }
293
294    private fun onMessageReceived(message: String) {
295        etMessage.append(message)
296    }
297
298    private fun onConnectionEstablished() {
299        etListenPort.isEnabled = true
300        etMessage.isEnabled = true
301    }
302
303    private fun onConnectionLost() {
304        etListenPort.isEnabled = false
305        etMessage.isEnabled = false
306    }
307
308    private fun onPortChanged() {
309        startListener()
310    }
311
312    private fun onMessageReceived(message: String) {
313        etMessage.append(message)
314    }
315
316    private fun onConnectionEstablished() {
317        etListenPort.isEnabled = true
318        etMessage.isEnabled = true
319    }
320
321    private fun onConnectionLost() {
322        etListenPort.isEnabled = false
323        etMessage.isEnabled = false
324    }
325
326    private fun onPortChanged() {
327        startListener()
328    }
329
330    private fun onMessageReceived(message: String) {
331        etMessage.append(message)
332    }
333
334    private fun onConnectionEstablished() {
335        etListenPort.isEnabled = true
336        etMessage.isEnabled = true
337    }
338
339    private fun onConnectionLost() {
340        etListenPort.isEnabled = false
341        etMessage.isEnabled = false
342    }
343
344    private fun onPortChanged() {
345        startListener()
346    }
347
348    private fun onMessageReceived(message: String) {
349        etMessage.append(message)
350    }
351
352    private fun onConnectionEstablished() {
353        etListenPort.isEnabled = true
354        etMessage.isEnabled = true
355    }
356
357    private fun onConnectionLost() {
358        etListenPort.isEnabled = false
359        etMessage.isEnabled = false
360    }
361
362    private fun onPortChanged() {
363        startListener()
364    }
365
366    private fun onMessageReceived(message: String) {
367        etMessage.append(message)
368    }
369
370    private fun onConnectionEstablished() {
371        etListenPort.isEnabled = true
372        etMessage.isEnabled = true
373    }
374
375    private fun onConnectionLost() {
376        etListenPort.isEnabled = false
377        etMessage.isEnabled = false
378    }
379
380    private fun onPortChanged() {
381        startListener()
382    }
383
384    private fun onMessageReceived(message: String) {
385        etMessage.append(message)
386    }
387
388    private fun onConnectionEstablished() {
389        etListenPort.isEnabled = true
390        etMessage.isEnabled = true
391    }
392
393    private fun onConnectionLost() {
394        etListenPort.isEnabled = false
395        etMessage.isEnabled = false
396    }
397
398    private fun onPortChanged() {
399        startListener()
400    }
401
402    private fun onMessageReceived(message: String) {
403        etMessage.append(message)
404    }
405
406    private fun onConnectionEstablished() {
407        etListenPort.isEnabled = true
408        etMessage.isEnabled = true
409    }
410
411    private fun onConnectionLost() {
412        etListenPort.isEnabled = false
413        etMessage.isEnabled = false
414    }
415
416    private fun onPortChanged() {
417        startListener()
418    }
419
420    private fun onMessageReceived(message: String) {
421        etMessage.append(message)
422    }
423
424    private fun onConnectionEstablished() {
425        etListenPort.isEnabled = true
426        etMessage.isEnabled = true
427    }
428
429    private fun onConnectionLost() {
430        etListenPort.isEnabled = false
431        etMessage.isEnabled = false
432    }
433
434    private fun onPortChanged() {
435        startListener()
436    }
437
438    private fun onMessageReceived(message: String) {
439        etMessage.append(message)
440    }
441
442    private fun onConnectionEstablished() {
443        etListenPort.isEnabled = true
444        etMessage.isEnabled = true
445    }
446
447    private fun onConnectionLost() {
448        etListenPort.isEnabled = false
449        etMessage.isEnabled = false
450    }
451
452    private fun onPortChanged() {
453        startListener()
454    }
455
456    private fun onMessageReceived(message: String) {
457        etMessage.append(message)
458    }
459
460    private fun onConnectionEstablished() {
461        etListenPort.isEnabled = true
462        etMessage.isEnabled = true
463    }
464
465    private fun onConnectionLost() {
466        etListenPort.isEnabled = false
467        etMessage.isEnabled = false
468    }
469
470    private fun onPortChanged() {
471        startListener()
472    }
473
474    private fun onMessageReceived(message: String) {
475        etMessage.append(message)
476    }
477
478    private fun onConnectionEstablished() {
479        etListenPort.isEnabled = true
480        etMessage.isEnabled = true
481    }
482
483    private fun onConnectionLost() {
484        etListenPort.isEnabled = false
485        etMessage.isEnabled = false
486    }
487
488    private fun onPortChanged() {
489        startListener()
490    }
491
492    private fun onMessageReceived(message: String) {
493        etMessage.append(message)
494    }
495
496    private fun onConnectionEstablished() {
497        etListenPort.isEnabled = true
498        etMessage.isEnabled = true
499    }
500
501    private fun onConnectionLost() {
502        etListenPort.isEnabled = false
503        etMessage.isEnabled = false
504    }
505
506    private fun onPortChanged() {
507        startListener()
508    }
509
510    private fun onMessageReceived(message: String) {
511        etMessage.append(message)
512    }
513
514    private fun onConnectionEstablished() {
515        etListenPort.isEnabled = true
516        etMessage.isEnabled = true
517    }
518
519    private fun onConnectionLost() {
520        etListenPort.isEnabled = false
521        etMessage.isEnabled = false
522    }
523
524    private fun onPortChanged() {
525        startListener()
526    }
527
528    private fun onMessageReceived(message: String) {
529        etMessage.append(message)
530    }
531
532    private fun onConnectionEstablished() {
533        etListenPort.isEnabled = true
534        etMessage.isEnabled = true
535    }
536
537    private fun onConnectionLost() {
538        etListenPort.isEnabled = false
539        etMessage.isEnabled = false
540    }
541
542    private fun onPortChanged() {
543        startListener()
544    }
545
546    private fun onMessageReceived(message: String) {
547        etMessage.append(message)
548    }
549
550    private fun onConnectionEstablished() {
551        etListenPort.isEnabled = true
552        etMessage.isEnabled = true
553    }
554
555    private fun onConnectionLost() {
556        etListenPort.isEnabled = false
557        etMessage.isEnabled = false
558    }
559
560    private fun onPortChanged() {
561        startListener()
562    }
563
564    private fun onMessageReceived(message: String) {
565        etMessage.append(message)
566    }
567
568    private fun onConnectionEstablished() {
569        etListenPort.isEnabled = true
570        etMessage.isEnabled = true
571    }
572
573    private fun onConnectionLost() {
574        etListenPort.isEnabled = false
575        etMessage.isEnabled = false
576    }
577
578    private fun onPortChanged() {
579        startListener()
580    }
581
582    private fun onMessageReceived(message: String) {
583        etMessage.append(message)
584    }
585
586    private fun onConnectionEstablished() {
587        etListenPort.isEnabled = true
588        etMessage.isEnabled = true
589    }
590
591    private fun onConnectionLost() {
592        etListenPort.isEnabled = false
593        etMessage.isEnabled = false
594    }
595
596    private fun onPortChanged() {
597        startListener()
598    }
599
600    private fun onMessageReceived(message: String) {
601        etMessage.append(message)
602    }
603
604    private fun onConnectionEstablished() {
605        etListenPort.isEnabled = true
606        etMessage.isEnabled = true
607    }
608
609    private fun onConnectionLost() {
610        etListenPort.isEnabled = false
611        etMessage.isEnabled = false
612    }
613
614    private fun onPortChanged() {
615        startListener()
616    }
617
618    private fun onMessageReceived(message: String) {
619        etMessage.append(message)
620    }
621
622    private fun onConnectionEstablished() {
623        etListenPort.isEnabled = true
624        etMessage.isEnabled = true
625    }
626
627    private fun onConnectionLost() {
628        etListenPort.isEnabled = false
629        etMessage.isEnabled = false
630    }
631
632    private fun onPortChanged() {
633        startListener()
634    }
635
636    private fun onMessageReceived(message: String) {
637        etMessage.append(message)
638    }
639
640    private fun onConnectionEstablished() {
641        etListenPort.isEnabled = true
642        etMessage.isEnabled = true
643    }
644
645    private fun onConnectionLost() {
646        etListenPort.isEnabled = false
647        etMessage.isEnabled = false
648    }
649
650    private fun onPortChanged() {
651        startListener()
652    }
653
654    private fun onMessageReceived(message: String) {
655        etMessage.append(message)
656    }
657
658    private fun onConnectionEstablished() {
659        etListenPort.isEnabled = true
660        etMessage.isEnabled = true
661    }
662
663    private fun onConnectionLost() {
664        etListenPort.isEnabled = false
665        etMessage.isEnabled = false
666    }
667
668    private fun onPortChanged() {
669        startListener()
670    }
671
672    private fun onMessageReceived(message: String) {
673        etMessage.append(message)
674    }
675
676    private fun onConnectionEstablished() {
677        etListenPort.isEnabled = true
678        etMessage.isEnabled = true
679    }
680
681    private fun onConnectionLost() {
682        etListenPort.isEnabled = false
683        etMessage.isEnabled = false
684    }
685
686    private fun onPortChanged() {
687        startListener()
688    }
689
690    private fun onMessageReceived(message: String) {
691        etMessage.append(message)
692    }
693
694    private fun onConnectionEstablished() {
695        etListenPort.isEnabled = true
696        etMessage.isEnabled = true
697    }
698
699    private fun onConnectionLost() {
700        etListenPort.isEnabled = false
701        etMessage.isEnabled = false
702    }
703
704    private fun onPortChanged() {
705        startListener()
706    }
707
708    private fun onMessageReceived(message: String) {
709        etMessage.append(message)
710    }
711
712    private fun onConnectionEstablished() {
713        etListenPort.isEnabled = true
714        etMessage.isEnabled = true
715    }
716
717    private fun onConnectionLost() {
718        etListenPort.isEnabled = false
719        etMessage.isEnabled = false
720    }
721
722    private fun onPortChanged() {
723        startListener()
724    }
725
726    private fun onMessageReceived(message: String) {
727        etMessage.append(message)
728    }
729
730    private fun onConnectionEstablished() {
731        etListenPort.isEnabled = true
732        etMessage.isEnabled = true
733    }
734
735    private fun onConnectionLost() {
736        etListenPort.isEnabled = false
737        etMessage.isEnabled = false
738    }
739
740    private fun onPortChanged() {
741        startListener()
742    }
743
744    private fun onMessageReceived(message: String) {
745        etMessage.append(message)
746    }
747
748    private fun onConnectionEstablished() {
749        etListenPort.isEnabled = true
750        etMessage.isEnabled = true
751    }
752
753    private fun onConnectionLost() {
754        etListenPort.isEnabled = false
755        etMessage.isEnabled = false
756    }
757
758    private fun onPortChanged() {
759        startListener()
760    }
761
762    private fun onMessageReceived(message: String) {
763        etMessage.append(message)
764    }
765
766    private fun onConnectionEstablished() {
767        etListenPort.isEnabled = true
768        etMessage.isEnabled = true
769    }
770
771    private fun onConnectionLost() {
772        etListenPort.isEnabled = false
773        etMessage.isEnabled = false
774    }
775
776    private fun onPortChanged() {
777        startListener()
778    }
779
780    private fun onMessageReceived(message: String) {
781        etMessage.append(message)
782    }
783
784    private fun onConnectionEstablished() {
785        etListenPort.isEnabled = true
786        etMessage.isEnabled = true
787    }
788
789    private fun onConnectionLost() {
790        etListenPort.isEnabled = false
791        etMessage.isEnabled = false
792    }
793
794    private fun onPortChanged() {
795        startListener()
796    }
797
798    private fun onMessageReceived(message: String) {
799        etMessage.append(message)
800    }
801
802    private fun onConnectionEstablished() {
803        etListenPort.isEnabled = true
804        etMessage.isEnabled = true
805    }
806
807    private fun onConnectionLost() {
808        etListenPort.isEnabled = false
809        etMessage.isEnabled = false
810    }
811
812    private fun onPortChanged() {
813        startListener()
814    }
815
816    private fun onMessageReceived(message: String) {
817        etMessage.append(message)
818    }
819
820    private fun onConnectionEstablished() {
821        etListenPort.isEnabled = true
822        etMessage.isEnabled = true
823    }
824
825    private fun onConnectionLost() {
826        etListenPort.isEnabled = false
827        etMessage.isEnabled = false
828    }
829
830    private fun onPortChanged() {
831        startListener()
832    }
833
834    private fun onMessageReceived(message: String) {
835        etMessage.append(message)
836    }
837
838    private fun onConnectionEstablished() {
839        etListenPort.isEnabled = true
840        etMessage.isEnabled = true
841    }
842
843    private fun onConnectionLost() {
844        etListenPort.isEnabled = false
845        etMessage.isEnabled = false
846    }
847
848    private fun onPortChanged() {
849        startListener()
850    }
851
852    private fun onMessageReceived(message: String) {
853        etMessage.append(message)
854    }
855
856    private fun onConnectionEstablished() {
857        etListenPort.isEnabled = true
858        etMessage.isEnabled = true
859    }
860
861    private fun onConnectionLost() {
862        etListenPort.isEnabled = false
863        etMessage.isEnabled = false
864    }
865
866    private fun onPortChanged() {
867        startListener()
868    }
869
870    private fun onMessageReceived(message: String) {
871        etMessage.append(message)
872    }
873
874    private fun onConnectionEstablished() {
875        etListenPort.isEnabled = true
876        etMessage.isEnabled = true
877    }
878
879    private fun onConnectionLost() {
880        etListenPort.isEnabled = false
881        etMessage.isEnabled = false
882    }
883
884    private fun onPortChanged() {
885        startListener()
886    }
887
888    private fun onMessageReceived(message: String) {
889        etMessage.append(message)
890    }
891
892    private fun onConnectionEstablished() {
893        etListenPort.isEnabled = true
894        etMessage.isEnabled = true
895    }
896
897    private fun onConnectionLost() {
898        etListenPort.isEnabled = false
899        etMessage.isEnabled = false
900    }
901
902    private fun onPortChanged() {
903        startListener()
904    }
905
906    private fun onMessageReceived(message: String) {
907        etMessage.append(message)
908    }
909
910    private fun onConnectionEstablished() {
911        etListenPort.isEnabled = true
912        etMessage.isEnabled = true
913    }
914
915    private fun onConnectionLost() {
916        etListenPort.isEnabled = false
917        etMessage.isEnabled = false
918    }
919
920    private fun onPortChanged() {
921        startListener()
922    }
923
924    private fun onMessageReceived(message: String) {
925        etMessage.append(message)
926    }
927
928    private fun onConnectionEstablished() {
929        etListenPort.isEnabled = true
930        etMessage.isEnabled = true
931    }
932
933    private fun onConnectionLost() {
934        etListenPort.isEnabled = false
935        etMessage.isEnabled = false
936    }
937
938    private fun onPortChanged() {
939        startListener()
940    }
941
942    private fun onMessageReceived(message: String) {
943        etMessage.append(message)
944    }
945
946    private fun onConnectionEstablished() {
947        etListenPort.isEnabled = true
948        etMessage.isEnabled = true
949    }
950
951    private fun onConnectionLost() {
952        etListenPort.isEnabled = false
953        etMessage.isEnabled = false
954    }
955
956    private fun onPortChanged() {
957        startListener()
958    }
959
960    private fun onMessageReceived(message: String) {
961        etMessage.append(message)
962    }
963
964    private fun onConnectionEstablished() {
965        etListenPort.isEnabled = true
966        etMessage.isEnabled = true
967    }
968
969    private fun onConnectionLost() {
970        etListenPort.isEnabled = false
971        etMessage.isEnabled = false
972    }
973
974    private fun onPortChanged() {
975        startListener()
976    }
977
978    private fun onMessageReceived(message: String) {
979        etMessage.append(message)
980    }
981
982    private fun onConnectionEstablished() {
983        etListenPort.isEnabled = true
984        etMessage.isEnabled = true
985    }
986
987    private fun onConnectionLost() {
988        etListenPort.isEnabled = false
989        etMessage.isEnabled = false
990    }
991
992    private fun onPortChanged() {
993        startListener()
994    }
995
996    private fun onMessageReceived(message: String) {
997        etMessage.append(message)
998    }
999
1000   private fun onConnectionEstablished() {
1001        etListenPort.isEnabled = true
1002        etMessage.isEnabled = true
1003    }
1004
1005   private fun onConnectionLost() {
1006        etListenPort.isEnabled = false
1007        etMessage.isEnabled = false
1008    }
1009
1010  private fun onPortChanged() {
1011        startListener()
1012    }
1013
1014  private fun onMessageReceived(message: String) {
1015        etMessage.append(message)
1016    }
1017
1018  private fun onConnectionEstablished() {
1019        etListenPort.isEnabled = true
1020        etMessage.isEnabled = true
1021    }
1022
1023  private fun onConnectionLost() {
1024        etListenPort.isEnabled = false
1025        etMessage.isEnabled = false
1026    }
1027
1028  private fun onPortChanged() {
1029        startListener()
1030    }
1031
1032  private fun onMessageReceived(message: String) {
1033        etMessage.append(message)
1034    }
1035
1036  private fun onConnectionEstablished() {
1037        etListenPort.isEnabled = true
1038        etMessage.isEnabled = true
1039    }
1040
1041  private fun onConnectionLost() {
1042        etListenPort.isEnabled = false
1043        etMessage.isEnabled = false
1044    }
1045
1046  private fun onPortChanged() {
1047        startListener()
1048    }
1049
1050  private fun onMessageReceived(message: String) {
1051        etMessage.append(message)
1052    }
1053
1054  private fun onConnectionEstablished() {
1055        etListenPort.isEnabled = true
1056        etMessage.isEnabled = true
1057    }
1058
1059  private fun onConnectionLost() {
1060        etListenPort.isEnabled = false
1061        etMessage.isEnabled = false
1062    }
1063
1064  private fun onPortChanged() {
1065        startListener()
1066    }
1067
1068  private fun onMessageReceived(message: String) {
1069        etMessage.append(message)
1070    }
1071
1072  private fun onConnectionEstablished() {
1073        etListenPort.isEnabled = true
1074        etMessage.isEnabled = true
1075    }
1076
1077  private fun onConnectionLost() {
1078        etListenPort.isEnabled = false
1079        etMessage.isEnabled = false
1080    }
1081
1082  private fun onPortChanged() {
1083        startListener()
1084    }
1085
1086  private fun onMessageReceived(message: String) {
1087        etMessage.append(message)
1088    }
1089
1090  private fun onConnectionEstablished() {
1091        etListenPort.isEnabled = true
1092        etMessage.isEnabled = true
1093    }
1094
1095  private fun onConnectionLost() {
1096        etListenPort.isEnabled = false
1097        etMessage.isEnabled = false
1098    }
1099
1100 private fun onPortChanged() {
1101        startListener()
1102    }
1103
1104 private fun onMessageReceived(message: String) {
1105        etMessage.append(message)
1106    }
1107
1108 private fun onConnectionEstablished() {
1109        etListenPort.isEnabled = true
1110        etMessage.isEnabled = true
1111    }
1112
1113 private fun onConnectionLost() {
1114        etListenPort.isEnabled = false
1115        etMessage.isEnabled = false
1116    }
1117
1118 private fun onPortChanged() {
1119        startListener()
1120    }
1121
1122 private fun onMessageReceived(message: String) {
1123        etMessage.append(message)
1124    }
1125
1126 private fun onConnectionEstablished() {
1127        etListenPort.isEnabled = true
1128        etMessage.isEnabled = true
1129    }
1130
1131 private fun onConnectionLost() {
1132        etListenPort.isEnabled = false
1133        etMessage.isEnabled = false
1134    }
1135
1136 private fun onPortChanged() {
1137        startListener()
1138    }
1139
1140 private fun onMessageReceived(message: String) {
1141        etMessage.append(message)
1142    }
1143
1144 private fun onConnectionEstablished() {
1145        etListenPort.isEnabled = true
1146        etMessage.isEnabled = true
1147    }
1148
1149 private fun onConnectionLost() {
1150        etListenPort.isEnabled = false
1151        etMessage.isEnabled = false
1152    }
1153
1154 private fun onPortChanged() {
1155        startListener()
1156    }
1157
1158 private fun onMessageReceived(message: String) {
1159        etMessage.append(message)
1160    }
1161
1162 private fun onConnectionEstablished() {
1163        etListenPort.isEnabled = true
1164        etMessage.isEnabled = true
1165    }
1166
1167 private fun onConnectionLost() {
1168        etListenPort.isEnabled = false
1169        etMessage.isEnabled = false
1170    }
1171
1172 private fun onPortChanged() {
1173        startListener()
1174    }
1175
1176 private fun onMessageReceived(message: String) {
1177        etMessage.append(message)
1178    }
1179
1180 private fun onConnectionEstablished() {
1181        etListenPort.isEnabled = true
1182        etMessage.isEnabled = true
1183    }
1184
1185 private fun onConnectionLost() {
1186        etListenPort.isEnabled = false
1187        etMessage.isEnabled = false
1188    }
1189
1190 private fun onPortChanged() {
1191        startListener()
1192    }
1193
1194 private fun onMessageReceived(message: String) {
1195        etMessage.append(message)
1196    }
1197
1198 private fun onConnectionEstablished() {
1199        etListenPort.isEnabled = true
1200        etMessage.isEnabled = true
1201    }
1202
1203 private fun onConnectionLost() {
1204        etListenPort.isEnabled = false
1205        etMessage.isEnabled = false
1206    }
1207
1208 private fun onPortChanged() {
1209        startListener()
1210    }
1211
1212 private fun onMessageReceived(message: String) {
1213        etMessage.append(message)
1214    }
1215
1216 private fun onConnectionEstablished() {
1217        etListenPort.isEnabled = true
1218        etMessage.isEnabled = true
1219    }
1220
1221 private fun onConnectionLost() {
1222        etListenPort.isEnabled = false
1223        etMessage.isEnabled = false
1224    }
1225
1226 private fun onPortChanged() {
1227        startListener()
1228    }
1229
1230 private fun onMessageReceived(message: String) {
1231        etMessage.append(message)
1232    }
1233
1234 private fun onConnectionEstablished() {
1235        etListenPort.isEnabled = true
1236        etMessage.isEnabled = true
1237    }
1238
1239 private fun onConnectionLost() {
1240        etListenPort.isEnabled = false
1241        etMessage.isEnabled = false
1242    }
1243
1244 private fun onPortChanged() {
1245        startListener()
1246    }
1247
1248 private fun onMessageReceived(message: String) {
1249        etMessage.append(message)
1250    }
1251
1252 private fun onConnectionEstablished() {
1253        etListenPort.isEnabled = true
1254        etMessage.isEnabled = true
1255    }
1256
1257 private fun onConnectionLost() {
1258        etListenPort.isEnabled = false
1259        etMessage.isEnabled = false
1260    }
1261
1262 private fun onPortChanged() {
1263        startListener()
1264    }
1265
1266 private fun onMessageReceived(message: String) {
1267        etMessage.append(message)
1268    }
1269
1270 private fun onConnectionEstablished() {
1271        etListenPort.isEnabled = true
1272        etMessage.isEnabled = true
1273    }
1274
1275 private fun onConnectionLost() {
1276        etListenPort.isEnabled = false
1277        etMessage.isEnabled = false
1278    }
1279
1280 private fun onPortChanged() {
1281        startListener()
1282    }
1283
1284 private fun onMessageReceived(message: String) {
1285        etMessage.append(message)
1286    }
1287
1288 private fun onConnectionEstablished() {
1289        etListenPort.isEnabled = true
1290        etMessage.isEnabled = true
1291    }
1292
1293 private fun onConnectionLost() {
1294        etListenPort.isEnabled = false
1295        etMessage.isEnabled = false
1296    }
1297
1298 private fun onPortChanged() {
1299        startListener()
1300    }
1301
1302 private fun onMessageReceived(message: String) {
1303        etMessage.append(message)
1304    }
1305
1306 private fun onConnectionEstablished() {
1307        etListenPort.isEnabled = true
1308        etMessage.isEnabled = true
1309    }
1310
1311 private fun onConnectionLost() {
1312        etListenPort.isEnabled = false
1313        etMessage.isEnabled = false
1314    }
1315
1316 private fun onPortChanged() {
1317        startListener()
1318    }
1319
1320 private fun onMessageReceived(message: String) {
1321        etMessage.append(message)
1322    }
1323
1324 private fun onConnectionEstablished() {
1325        etListenPort.isEnabled = true
1326        etMessage.isEnabled = true
1327    }
1328
1329 private fun onConnectionLost() {
1330        etListenPort.isEnabled = false
1331        etMessage.isEnabled = false
1332    }
1333
1334 private fun onPortChanged() {
1335        startListener()
1336    }
1337
1338 private fun onMessageReceived(message: String) {
1339        etMessage.append(message)
1340    }
1341
1342 private fun onConnectionEstablished() {
1343        etListenPort.isEnabled = true
1344        etMessage.isEnabled = true
1345    }
1346
1347 private fun onConnectionLost() {
1348        etListenPort.isEnabled = false
1349        etMessage.isEnabled = false
1350    }
1351
1352 private fun onPortChanged() {
1353        startListener()
1354    }
1355
1356 private fun onMessageReceived(message: String) {
1357        etMessage.append(message)
1358    }
1359
1360 private fun onConnectionEstablished() {
1361        etListenPort.isEnabled = true
1362        etMessage.isEnabled = true
1363    }
1364
1365 private fun onConnectionLost() {
1366        etListenPort.isEnabled = false
1367        etMessage.isEnabled = false
1368    }
1369
1370 private fun onPortChanged() {
1371        startListener()
1372    }
1373
1374 private fun onMessageReceived(message: String) {
1375        etMessage.append(message)
1376    }
1377
1378 private fun onConnectionEstablished() {
1379        etListenPort.isEnabled = true
1380        etMessage.isEnabled = true
1381    }
1382
1383 private fun onConnectionLost() {
1384        etListenPort.isEnabled = false
1385        etMessage.isEnabled = false
1386    }
1387
1388 private fun onPortChanged() {
1389        startListener()
1390    }
1391
1392 private fun onMessageReceived(message: String) {
1393        etMessage.append(message)
1394    }
1395
1396 private fun onConnectionEstablished() {
1397        etListenPort.isEnabled = true
1398        etMessage.isEnabled = true
1399    }
1400
1401 private fun onConnectionLost() {
1402        etListenPort.isEnabled = false
1403        etMessage.isEnabled = false
1404    }
1405
1406 private fun onPortChanged() {
1407        startListener()
1408    }
1409
1410 private fun onMessageReceived(message: String) {
1411        etMessage.append(message)
1412    }
1413
1414 private fun onConnectionEstablished() {
1415        etListenPort.isEnabled = true
1416        etMessage.isEnabled = true
1417    }
1418
1419 private fun onConnectionLost() {
1420        etListenPort.isEnabled = false
1421        etMessage.isEnabled = false
1422    }
1423
1424 private fun onPortChanged() {
1425        startListener()
1426    }
1427
1428 private fun onMessageReceived(message: String) {
1429        etMessage.append(message)
1430    }
1431
1432 private fun onConnectionEstablished() {
1433        etListenPort.isEnabled = true
1434        etMessage.isEnabled = true
1435    }
1436
1437 private fun onConnectionLost() {
1438        etListenPort.isEnabled = false
1439        etMessage.isEnabled = false
1440    }
1441

```

```

135
136     private fun listenForMessages() {
137         try {
138             // Create a buffer for incoming packets
139             val buffer = ByteArray( size: 1024)
140
141             while (isListening && listenerSocket != null && !listenerSocket!!.isClosed) {
142                 // Create a DatagramPacket to receive data
143                 val packet = DatagramPacket(buffer, buffer.size)
144
145                 try {
146                     // Wait for a packet to arrive
147                     listenerSocket?.receive(packet)
148
149                     // Extract the message from the packet
150                     val message = String(packet.data, offset: 0, packet.length, StandardCharsets.UTF_8)
151                     val sender = packet.address.hostAddress
152
153                     Log.d(TAG, msg: "Nhận tin nhắn từ $sender: $message")
154
155                     // Update UI on the main thread
156                     handler.post {
157                         addMessage("Nhận từ $sender: $message")
158                     }
159                 } catch (e: Exception) {
160                     if (isListening) {
161                         Log.e(TAG, msg: "Lỗi khi nhận tin nhắn: ${e.message}", e)
162                     }
163                 }
164             }
165         }
166
167         private fun stopListener() {
168             if (isListening) {
169                 isListening = false
170
171                 executorService.execute {
172                     try {
173                         listenerSocket?.close()
174                         listenerThread?.join( millis: 1000) // Wait for the listener thread to finish
175
176                         handler.post {
177                             tvStatus.text = "Trạng thái: Đã dừng lắng nghe"
178                             addMessage("Đã dừng lắng nghe")
179                             btnStartStopListener.text = "Bắt đầu lắng nghe"
180                         }
181                     } catch (e: Exception) {
182                         Log.e(TAG, msg: "Lỗi khi dừng lắng nghe: ${e.message}", e)
183
184                         handler.post {
185                             Toast.makeText( context: this, text: "Lỗi khi dừng lắng nghe: ${e.message}", Toast.LENGTH_SHORT).show
186                         }
187                     }
188                 }
189             }
190         }
191     }

```

```
210
211     private fun sendMessage() {
212         val message = etMessage.text.toString().trim()
213         if (message.isEmpty()) {
214             Toast.makeText(context: this, text: "Vui lòng nhập tin nhắn", Toast.LENGTH_SHORT).show()
215             return
216         }
217
218         val targetIP = etTargetIP.text.toString().trim()
219         if (targetIP.isEmpty()) {
220             Toast.makeText(context: this, text: "Vui lòng nhập địa chỉ IP đích", Toast.LENGTH_SHORT).show()
221             return
222         }
223
224         val targetPortStr = etTargetPort.text.toString().trim()
225         if (targetPortStr.isEmpty()) {
226             Toast.makeText(context: this, text: "Vui lòng nhập cổng đích", Toast.LENGTH_SHORT).show()
227             return
228         }
229
230         val targetPort = targetPortStr.toIntOrNull()
231         if (targetPort == null || targetPort <= 0 || targetPort > 65535) {
232             Toast.makeText(context: this, text: "Cổng đích không hợp lệ", Toast.LENGTH_SHORT).show()
233             return
234         }
235
236         executorService.execute {
237             var socket: DatagramSocket? = null
238
239             try {
240                 ...
241
242
243                 executorService.execute {
244                     var socket: DatagramSocket? = null
245
246                     try {
247                         // Create a DatagramSocket for sending
248                         socket = DatagramSocket()
249
250                         // Convert the message to bytes
251                         val messageBytes = message.toByteArray(StandardCharsets.UTF_8)
252
253                         // Create the destination address
254                         val address = InetAddress.getByName(targetIP)
255
256                         // Create a DatagramPacket with the message data
257                         val packet = DatagramPacket(messageBytes, messageBytes.size, address, targetPort)
258
259                         // Send the packet
260                         socket.send(packet)
261
262                         Log.d(TAG, msg: "Đã gửi tin nhắn đến $targetIP:$targetPort: $message")
263
264                         handler.post {
265                             addMessage("Gửi đến $targetIP:$targetPort: $message")
266                             etMessage.text.clear()
267                         }
268
269             } catch (e: Exception) {
270                 Log.e(TAG, msg: "Lỗi khi gửi tin nhắn: ${e.message}", e)
271             }
272         }
273     }
274 }
```

```
272
273     }
274 }
275
276     private fun addMessage(message: String) {
277         val time = dateFormat.format(Date())
278         tvMessages.append("[${time}] $message\n\n")
279
280         // Scroll to the bottom to show the latest message
281         scrollView.post {
282             scrollView.fullScroll(ScrollView.FOCUS_DOWN)
283         }
284     }
285
286     override fun onDestroy() {
287         super.onDestroy()
288
289         // Clean up resources
290         stopListener()
291         executorService.shutdown()
292     }
293
294 }
```

```
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />
    ! <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
```