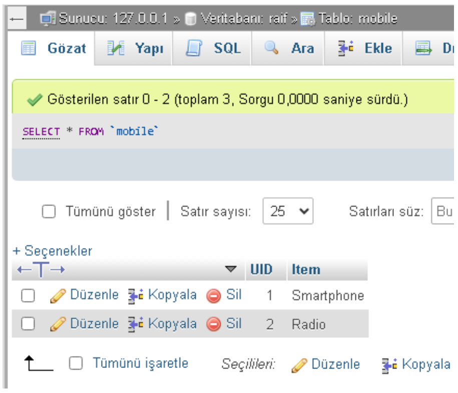
EREN KIZILIRMAK 210704025

Android Client Flask Server

XAMPP is a free, open-source cross-platform web server solution stack package that includes Apache, MySQL, PHP, and Perl for local web development. In this project, we will use XAMPP for managing the MYSQL database used by the android client.

We already set up XAMPP with a database and a table in that database with name mobile.

Let’s add two entries to the table.



Gösterilen satır 0 - 2 (toplam 3, Sorgu 0,0000 saniye sürdü.)

```
SELECT * FROM `mobile`
```

Tümünü göster | Satır sayısı: 25 Satırları süz: Bu

+ Seçenekler

	UID	Item
Düzenle Kopyala Sil	1	Smartphone
Düzenle Kopyala Sil	2	Radio

Tümünü işaretle *Seçilileri:* Düzenle Kopyala

We will first implement a Flask Server to read and write to the table

Prompt: I have an XAMPP and use MySQL. The database name is raif the table information is as follows host='localhost', user='root', password='', # Empty.. no password database='mobile', # Replace with your database name. The two column names are UID and Item. Can you implement a flask server that will list all items in the table.

Open a cmd window

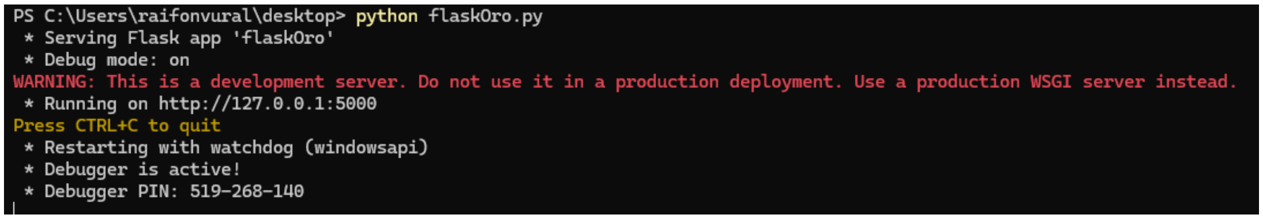
Install required packages.

```
pip install Flask

pip install mysql-connector-python
```

This is the flask server code in python

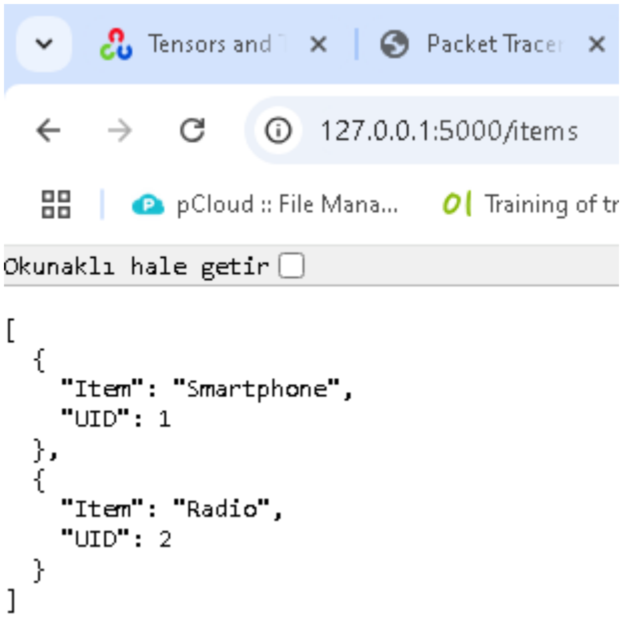
```
from flask import Flask, jsonify
import mysql.connector

app = Flask(__name__)

# MySQL configuration
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'raif'
}

@app.route('/items', methods=['GET'])
def get_items():
    conn = None
    cursor = None
    try:
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT UID, Item FROM mobile")
        results = cursor.fetchall()
        return jsonify(results)
    except mysql.connector.Error as err:
        return jsonify({'error': str(err)})
    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()

if __name__ == '__main__':
    app.run(debug=True)
```

run your flask server program from the command line, you should see



```
PS C:\Users\raifonvural\desktop> python flaskOro.py
 * Serving Flask app 'flaskOro'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 519-268-140
```

Now enter http://127.0.0.1:5000/items in your browser



```
[
  {
    "Item": "Smartphone",
    "UID": 1
  },
  {
    "Item": "Radio",
    "UID": 2
  }
]
```

Your server code and XAMPP are working

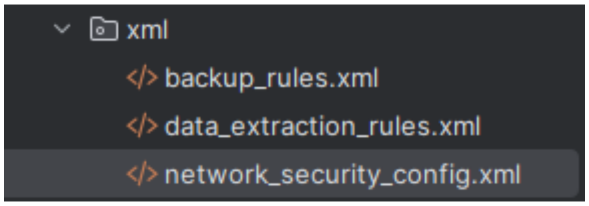
Android App

Create a new project with Empty Views Activity

To allow android app to access local server with http (since we cannot implement secure server with https) we need to add network_security_config.xml file.

xml > New > XML Resource File and name it network_security_config.xml

To be able to access local server with http (not https) add network_security_config.xml



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">192.168.1.107</domain>
        <base-config cleartextTrafficPermitted="true" />
    </domain-config>
</network-security-config>
```

You need to modify the IP address to your IP address. (use ipconfig in cmd window)

Modify AndroidManifest.xml.

```
<?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:networkSecurityConfig="@xml/network_security_config"
```

activity_main.xml has a single button

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

    <Button
        android:id="@+id/buttonMp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open Item Display"
        android:layout_centerInParent="true" />
</RelativeLayout>
```

Create activity_item_display.xml

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

    <Button
        android:id="@+id/btnRetrieveItems"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Retrieve Items" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:importantForAccessibility="no">

        <LinearLayout
            android:id="@+id/layoutItemsContainer"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
        </LinearLayout>
    </ScrollView>

    <TextView
        android:id="@+id/tvItemDetails"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Item details here"
        android:visibility="gone" />

</LinearLayout>
```

The button in activity_main.xml should set an onClickListener, when clicked it should start another activity called ItemDisplay

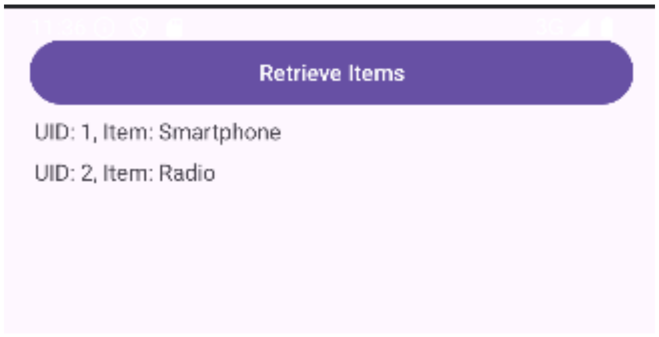
```kotlin
import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity

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

        val button = findViewById<Button>(R.id.buttonMp)
        button.setOnClickListener {
            startActivity(Intent(this, ItemDisplay::class.java))
        }
    }
}
```

ItemDisplay.kt

```kotlin
import android.os.Bundle
import android.widget.Button
import android.widget.LinearLayout
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import kotlinx.coroutines.*
import org.json.JSONArray
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL

class ItemDisplay : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_item_display)

        val btnRetrieveItems = findViewById<Button>(R.id.btnRetrieveItems)
        val layoutItemsContainer =
findViewById<LinearLayout>(R.id.layoutItemsContainer)

        btnRetrieveItems.setOnClickListener {
            CoroutineScope(Dispatchers.IO).launch {
                val items = retrieveItemsFromServer()
                withContext(Dispatchers.Main) {
                    layoutItemsContainer.removeAllViews()
                    items?.forEach { item ->
                        val tvItem = TextView(this@ItemDisplay).apply {
                            text = "UID: ${item.UID}, Item: ${item.item}"
                            layoutParams = LinearLayout.LayoutParams(
                                LinearLayout.LayoutParams.MATCH_PARENT,
                                LinearLayout.LayoutParams.WRAP_CONTENT
                            )
                            setPadding(8, 8, 8, 8)
                        }
                        layoutItemsContainer.addView(tvItem)
                    }
                }
            }
        }
    }

    suspend fun retrieveItemsFromServer(): List<ItemData>? {
        val url = URL("http://192.168.1.12:5000/items")
        val connection = url.openConnection() as HttpURLConnection
        connection.requestMethod = "GET"
        connection.connectTimeout = 10000
        connection.readTimeout = 10000

        return if (connection.responseCode == HttpURLConnection.HTTP_OK) {
            val inputStream = connection.inputStream
            val reader = BufferedReader(InputStreamReader(inputStream))
            val response = reader.use { it.readText() }
            parseResponseList(response)
        } else {
            null
        }
    }

    fun parseResponseList(response: String?): List<ItemData>? {
        return if (response != null) {
            try {
                val jsonArray = JSONArray(response)
                val items = mutableListOf<ItemData>()
                for (i in 0 until jsonArray.length()) {
                    val itemObject = jsonArray.getJSONObject(i)
                    val itemData = ItemData(
                        UID = itemObject.getInt("UID"),
                        item = itemObject.getString("Item")
                    )
                    items.add(itemData)
                }
                items
            } catch (e: Exception) {
                null
            }
        } else {
            null
        }
    }
}

data class ItemData(val UID: Int, val item: String)
```

You need to modify the IP address to your IP address.

Run the code, the emulator should display the contents of your table



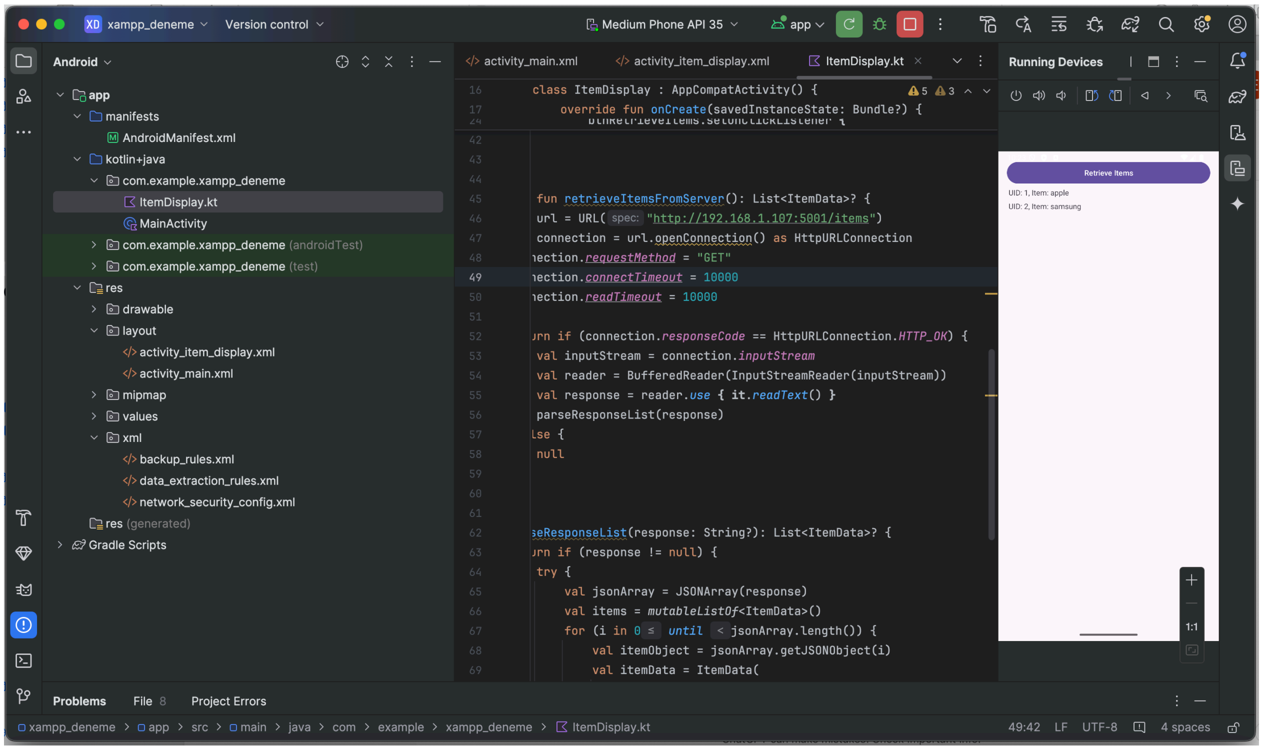
```
UID: 1, Item: Smartphone
UID: 2, Item: Radio
```

Describe the code in ItemDisplay.kt (3 points)

ItemDisplay retrieves a list of items from a Flask API. (my mac's system features uses 5000 port on the local so i used 5001 instead.) and displays them in a vertical list. It uses coroutines to network operations and then updates the UI on the main thread. The retrieved data is in in JSON format, which is parsed into a list of ItemData objects. and them viewed in the list of items with their UIDs and item names.

It sets a binding to the retrieve object when the user clicks the retrieve button, it runs the function which connects the http addrress on the net and retrieves the flask output.

Paste the screen capture of emulator displaying the table content (2 points)



```
class ItemDisplay : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        btnRetrieveItems.setOnClickListener {


 fun retrieveItemsFromServer(): List<ItemData>? {
 url = URL( spec: "http://192.168.1.107:5001/items")
 connection = url.openConnection() as HttpURLConnection
nection.requestMethod = "GET"
nection.connectTimeout = 10000
nection.readTimeout = 10000

urn if (connection.responseCode == HttpURLConnection.HTTP_OK) {
 val inputStream = connection.inputStream
 val reader = BufferedReader(InputStreamReader(inputStream))
 val response = reader.use { it.readText() }
 parseResponseList(response)
lse {
 null



seResponseList(response: String?): List<ItemData>? {
urn if (response != null) {
 try {
     val jsonArray = JSONArray(response)
     val items = mutableListOf<ItemData>()
     for (i in 0 ≤ until < jsonArray.length()) {
         val itemObject = jsonArray.getJSONObject(i)
         val itemData = ItemData(
```

```
UID: 1, Item: apple
UID: 2, Item: samsung
```

```
xampp_deneme > app > src > main > java > com > example > xampp_deneme > ItemDisplay.kt        49:42  LF  UTF-8  4 spaces
```

(20 points if working, no points if not working)
Modify flask server code to support adding an item, paste the code here (use chat gpt as desired)

```
from flask import Flask, jsonify, request
import mysql.connector

app = Flask(__name__)

# MySQL configuration
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'mobil'
}

@app.route('/items', methods=['GET'])
def get_items():
    conn = None
    cursor = None
    try:
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT UID, Item FROM mobile")
        results = cursor.fetchall()
        return jsonify(results)
    except mysql.connector.Error as err:
        return jsonify({'error': str(err)})
    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()

@app.route('/items', methods=['POST'])
def add_item():
    conn = None
    cursor = None
    try:
        data = request.get_json()
        if not data or 'Item' not in data:
            return jsonify({'error': 'Item field is required'}), 400

        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor(dictionary=True)
        query = "INSERT INTO mobile (Item) VALUES (%s)"
        cursor.execute(query, (data['Item'],))
        conn.commit()

        return jsonify({
            'message': 'Item added successfully',
            'id': cursor.lastrowid
        }), 201

    except mysql.connector.Error as err:
        return jsonify({'error': str(err)}), 500
    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0',port=5001)
```

Modify ItemDisplay.kt and activity_item_display.xml to support adding a new item (use chat gpt as desired)

itemdisplay.kt code : package com.example.xampp_deneme

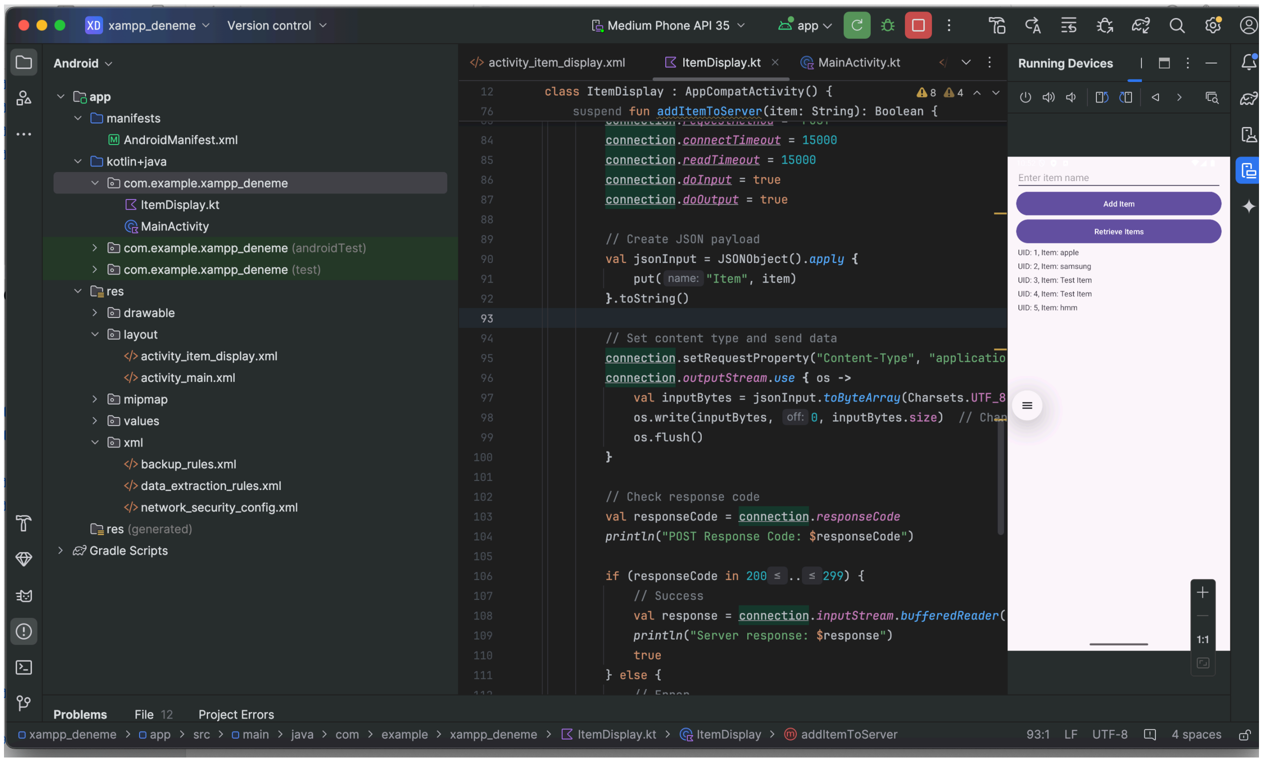
```
import android.os.Bundle
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import kotlinx.coroutines.*
import org.json.JSONObject
import java.io.OutputStreamWriter
import java.net.HttpURLConnection
import java.net.URL

class ItemDisplay : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_item_display)

        val btnRetrieveItems = findViewById<Button>(R.id.btnRetrieveItems)
        val btnAddItem = findViewById<Button>(R.id.btnAddItem)
        val etItemInput = findViewById<EditText>(R.id.etItemInput)
        val layoutItemsContainer = findViewById<LinearLayout>(R.id.layoutItemsContainer)

        btnRetrieveItems.setOnClickListener {
            CoroutineScope(Dispatchers.IO).launch {
                val items = retrieveItemsFromServer()
                withContext(Dispatchers.Main) {
                    layoutItemsContainer.removeAllViews()
                    items?.forEach { item ->
                        val tvItem = TextView(this@ItemDisplay).apply {
                            text = "UID: ${item.UID}, Item: ${item.item}"
                            layoutParams = LinearLayout.LayoutParams(
                                LinearLayout.LayoutParams.MATCH_PARENT,
                                LinearLayout.LayoutParams.WRAP_CONTENT
                            )
                            setPadding(8, 8, 8, 8)
                        }
                        layoutItemsContainer.addView(tvItem)
                    }
                }
            }
        }

        btnAddItem.setOnClickListener {
            val itemText = etItemInput.text.toString()
            if (itemText.isNotEmpty()) {
                CoroutineScope(Dispatchers.IO).launch {
                    val success = addItemToServer(itemText)
                    withContext(Dispatchers.Main) {
                        if (success) {
                            Toast.makeText(this@ItemDisplay, "Item added successfully",
Toast.LENGTH_SHORT).show()
                            etItemInput.text.clear()
                        } else {
                            Toast.makeText(this@ItemDisplay, "Failed to add item", Toast.LENGTH_SHORT).show()
                        }
                    }
                }
            } else {
                Toast.makeText(this, "Please enter an item", Toast.LENGTH_SHORT).show()
            }
        }
    }

    suspend fun retrieveItemsFromServer(): List<ItemData>? {
        val url = URL("http://192.168.1.107:5001/items")
        val connection = url.openConnection() as HttpURLConnection
        connection.requestMethod = "GET"
        connection.connectTimeout = 10000
        connection.readTimeout = 10000

        return if (connection.responseCode == HttpURLConnection.HTTP_OK) {
            val response = connection.inputStream.bufferedReader().use { it.readText() }
            parseResponseList(response)
        } else {
            null
        }
    }
    suspend fun addItemToServer(item: String): Boolean {
        var connection: HttpURLConnection? = null

        return try {
            // Setup connection
            val url = URL("http://192.168.1.107:5001/add_item")
            connection = url.openConnection() as HttpURLConnection
            connection.requestMethod = "POST"
            connection.connectTimeout = 15000
            connection.readTimeout = 15000
            connection.doInput = true
            connection.doOutput = true

            // Create JSON payload
            val jsonInput = JSONObject().apply {
                put("Item", item)
            }.toString()

            // Set content type and send data
            connection.setRequestProperty("Content-Type", "application/json; charset=UTF-8")
            connection.outputStream.use { os ->
                val inputBytes = jsonInput.toByteArray(Charsets.UTF_8)
                os.write(inputBytes, 0, inputBytes.size)  // Changed 'length' to 'size'
                os.flush()
            }

            // Check response code
            val responseCode = connection.responseCode
            println("POST Response Code: $responseCode")

            if (responseCode in 200..299) {
                // Success
                val response = connection.inputStream.bufferedReader().use { it.readText() }
                println("Server response: $response")
                true
            } else {
                // Error
                val errorResponse = connection.errorStream?.bufferedReader()?.use { it.readText() } ?: "No error
details"
                println("Error response code: $responseCode")
                println("Error response: $errorResponse")
                false
            }
        } catch (e: Exception) {
            e.printStackTrace()
            println("Exception while adding item: ${e.message}")
            false
        } finally {
            connection?.disconnect()
        }
    }


    fun parseResponseList(response: String?): List<ItemData>? {
        return response?.let {
            try {
                val jsonArray = org.json.JSONArray(it)
                List(jsonArray.length()) { i ->
                    val itemObject = jsonArray.getJSONObject(i)
                    ItemData(
                        UID = itemObject.getInt("UID"),
                        item = itemObject.getString("Item")
                    )
                }
            } catch (e: Exception) {
                null
            }
        }
    }
}

data class ItemData(val UID: Int, val item: String)
```

activity_item dsiplay code:

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

    <EditText
        android:id="@+id/etItemInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter item name" />

    <Button
        android:id="@+id/btnAddItem"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:minHeight="48dp"
        android:padding="12dp"
        android:text="Add Item" />

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

```
      android:minHeight="48dp"
      android:padding="12dp"
      android:text="Retrieve Items" />

    <ScrollView
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight="1"
      android:importantForAccessibility="no">

      <LinearLayout
        android:id="@+id/layoutItemsContainer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
      </LinearLayout>
    </ScrollView>

</LinearLayout>
```

Take a screen capture of android emulator for adding an item and the database table with item added.



```
class ItemDisplay : AppCompatActivity() {
    suspend fun addItemToServer(item: String): Boolean {
            connection.connectTimeout = 15000
            connection.readTimeout = 15000
            connection.doInput = true
            connection.doOutput = true

            // Create JSON payload
            val jsonInput = JSONObject().apply {
                put( name: "Item", item)
            }.toString()

            // Set content type and send data
            connection.setRequestProperty("Content-Type", "applicatio
            connection.outputStream.use { os ->
                val inputBytes = jsonInput.toByteArray(Charsets.UTF_8
                os.write(inputBytes, off: 0, inputBytes.size)  // Chan
                os.flush()
            }

            // Check response code
            val responseCode = connection.responseCode
            println("POST Response Code: $responseCode")

            if (responseCode in 200 ≤.. ≤ 299) {
                // Success
                val response = connection.inputStream.bufferedReader(
                println("Server response: $response")
                true
            } else {
```

```
UID: 1, Item: apple
UID: 2, Item: samsung
UID: 3, Item: Test Item
UID: 4, Item: Test Item
UID: 5, Item: hmm
```