Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 9**	
Title of Lab Assignment: Android program based on Rest API.		
DOP: 24-10-2024	**DOS: 25-10-2024**	
CO Mapped: **CO2, CO5**	**PO Mapped:** **PO2, PO3, PO5, PSO2**	**Signature:**

Practical No. 9

Aim: Android program based on Rest API.

1. Create a basic application that allows you to download HTML from a given web page using HttpURLConnection.
2. Create an application to parse the data using JSON Object methods and set it in the Text View's. (Employee name and salary stored in JSON format

Theory:

1. **HttpURLConnection Class in Android:** The HttpURLConnection class in Android facilitates communication with REST APIs over HTTP. This class handles network requests and responses, making it ideal for interacting with web services.

   Key methods of HttpURLConnection:

   1. openConnection(): Opens a connection to the specified URL.
   2. setRequestMethod(): Sets the HTTP method (GET, POST, etc.) for the request.
   3. getInputStream(): Retrieves the input stream from the server response.
   4. getOutputStream(): Writes data to the server (used in POST/PUT requests).
   5. getResponseCode(): Checks the HTTP response status (e.g., 200 for success).
   6. disconnect(): Closes the connection once the task is complete.

   Steps to Use HttpURLConnection:

   1. Establish the Connection: Create a URL object and call openConnection() to establish the connection.
   2. Configure the Request: Set properties like request method (GET or POST), headers, timeouts, etc.
   3. Send Data (if applicable): For POST requests, write data to the server using OutputStream.
   4. Read Response: Use InputStream to read the response from the server. For HTML data, it can be processed line by line using BufferedReader.
   5. Disconnect: Close the connection to free resources using disconnect().

2. **JSON (JavaScript Object Notation)**

   JSON is a lightweight data-interchange format that is easy to read and write for humans and easy to parse and generate for machines. It is widely used in REST APIs to exchange data between the client and server.

- Structure of JSON:
  - JSONObject: A collection of key/value pairs. The keys are strings, and the values can be strings, numbers, booleans, arrays, or other JSON objects.
  - JSONArray: An ordered list of values, which can be of any JSON type (e.g., strings, numbers, or objects).

Code:

MainActivity.java

```
package com.example.practical9;
import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class MainActivity extends AppCompatActivity {
    EditText urlInput;
    TextView htmlContent;
    Button fetchHtmlButton;
    ProgressDialog progressDialog;

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

        urlInput = findViewById(R.id.urlInput);
        htmlContent = findViewById(R.id.htmlContent);
```

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

    fetchHtmlButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String urlString = urlInput.getText().toString();
            new FetchHtmlTask().execute(urlString);
        }
    });
}

// AsyncTask to fetch HTML content
private class FetchHtmlTask extends AsyncTask<String, String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        progressDialog = new ProgressDialog(MainActivity.this);
        progressDialog.setMessage("Fetching HTML...");
        progressDialog.setCancelable(false);
        progressDialog.show();
    }

    @Override
    protected String doInBackground(String... params) {
        String urlString = params[0];
        String result = "";
        try {
            URL url = new URL(urlString);
            HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
            InputStream in = urlConnection.getInputStream();
            InputStreamReader isw = new InputStreamReader(in);

            int data = isw.read();
            while (data != -1) {
                result += (char) data;
                data = isw.read();
```

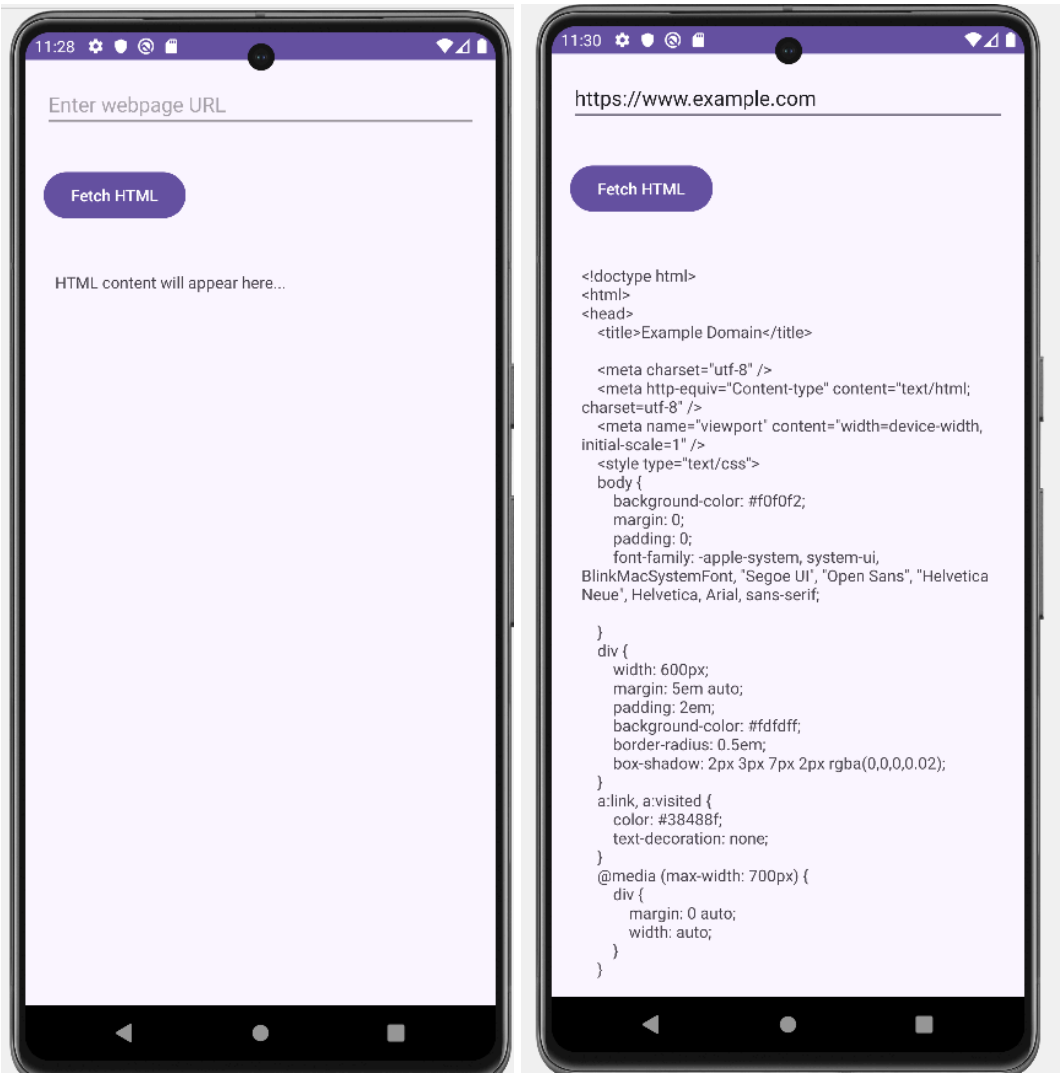
```
            }

            return result;
        } catch (Exception e) {
            e.printStackTrace();
            return "Error fetching HTML";
        }
    }

    @Override
    protected void onPostExecute(String result) {
        super.onPostExecute(result);
        progressDialog.dismiss();
        htmlContent.setText(result);
    }
  }
}
```

activity_main.xml

```
<?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">

    <!-- Input for URL -->
    <EditText
        android:id="@+id/urlInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter webpage URL"
        android:inputType="textUri"
        android:layout_margin="16dp" />

    <!-- Button to Fetch HTML -->
    <Button
        android:id="@+id/fetchHtmlButton"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Fetch HTML"
        android:layout_below="@id/urlInput"
        android:layout_margin="16dp" />

    <!-- TextView to display the fetched HTML content -->
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/fetchHtmlButton"
        android:layout_margin="16dp">

        <TextView
            android:id="@+id/htmlContent"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="HTML content will appear here..."
            android:textIsSelectable="true"
            android:padding="10dp" />
    </ScrollView>
</RelativeLayout>
```

Output:


11:28
Enter webpage URL
Fetch HTML
HTML content will appear here...
11:30
https://www.example.com
Fetch HTML
<!doctype html>
<html>
<head>
<title>Example Domain</title>
<meta charset="utf-8" />
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<style type="text/css">
body {
background-color: #f0f0f2;
margin: 0;
padding: 0;
font-family: -apple-system, system-ui, BlinkMacSystemFont, 'Segoe UI', "Open Sans", "Helvetica Neue', Helvetica, Arial, sans-serif;
}
div {
width: 600px;
margin: 5em auto;
padding: 2em;
background-color: #fdfdff;
border-radius: 0.5em;
box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
}
a:link, a:visited {
color: #38488f;
text-decoration: none;
}
@media (max-width: 700px) {
div {
margin: 0 auto;
width: auto;
}
}

3. **REST API Overview**

REST (Representational State Transfer) is a set of architectural principles used for designing web services that allow communication over the internet. REST APIs use standard HTTP methods to perform operations on resources, where each resource is identified by a URI (Uniform Resource Identifier). RESTful services are widely used because of their simplicity, flexibility, and ability to use lightweight data formats such as JSON or XML.

- HTTP Methods in REST API:
  - GET: Retrieves data from the server (e.g., fetching employee details).
  - POST: Sends data to the server to create a new resource (e.g., adding a new employee).
  - PUT: Updates an existing resource (e.g., updating employee salary).
  - DELETE: Removes a resource from the server (e.g., deleting an employee record).
- Advantages of REST API:
  - Stateless: Each request from the client to the server must contain all the information the server needs to fulfill the request, making it stateless and scalable.
  - Cacheable: Responses can be cached to improve performance.
  - Platform Independent: REST API can be used across different programming languages and platforms.
  - Lightweight: REST primarily uses JSON, which is less verbose than XML, making data transfer faster and more efficient.

Code:

MainActivity.java

```
package com.example.practical9;

import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import java.io.InputStream;
```

```
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import org.json.JSONArray;
import org.json.JSONObject;

public class MainActivity extends AppCompatActivity {
    String myUrl = "https://dummyjson.com/c/7bb9-58b5-4d5f-8584";
    TextView resultsTextView;
    Button displayData;
    ProgressDialog progressDialog;

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

        resultsTextView = findViewById(R.id.results);
        displayData = findViewById(R.id.displayData);

        displayData.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                new MyAsyncTasks().execute();
            }
        });
    }

    private class MyAsyncTasks extends AsyncTask<String, String, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            progressDialog = new ProgressDialog(MainActivity.this);
            progressDialog.setMessage("Fetching data...");
            progressDialog.setCancelable(false);
            progressDialog.show();
```

```
    }

    @Override
    protected String doInBackground(String... params) {
        String result = "";
        try {
            URL url = new URL(myUrl);
            HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
            InputStream in = urlConnection.getInputStream();
            InputStreamReader isw = new InputStreamReader(in);

            int data = isw.read();
            while (data != -1) {
                result += (char) data;
                data = isw.read();
            }

            return result;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @Override
    protected void onPostExecute(String result) {
        super.onPostExecute(result);
        progressDialog.dismiss();

        try {
            // Parse the JSON result
            JSONObject jsonObject = new JSONObject(result);

            // Extract the "Names" array
            JSONArray namesArray = jsonObject.getJSONArray("Names");
            // Build a string with the names
```

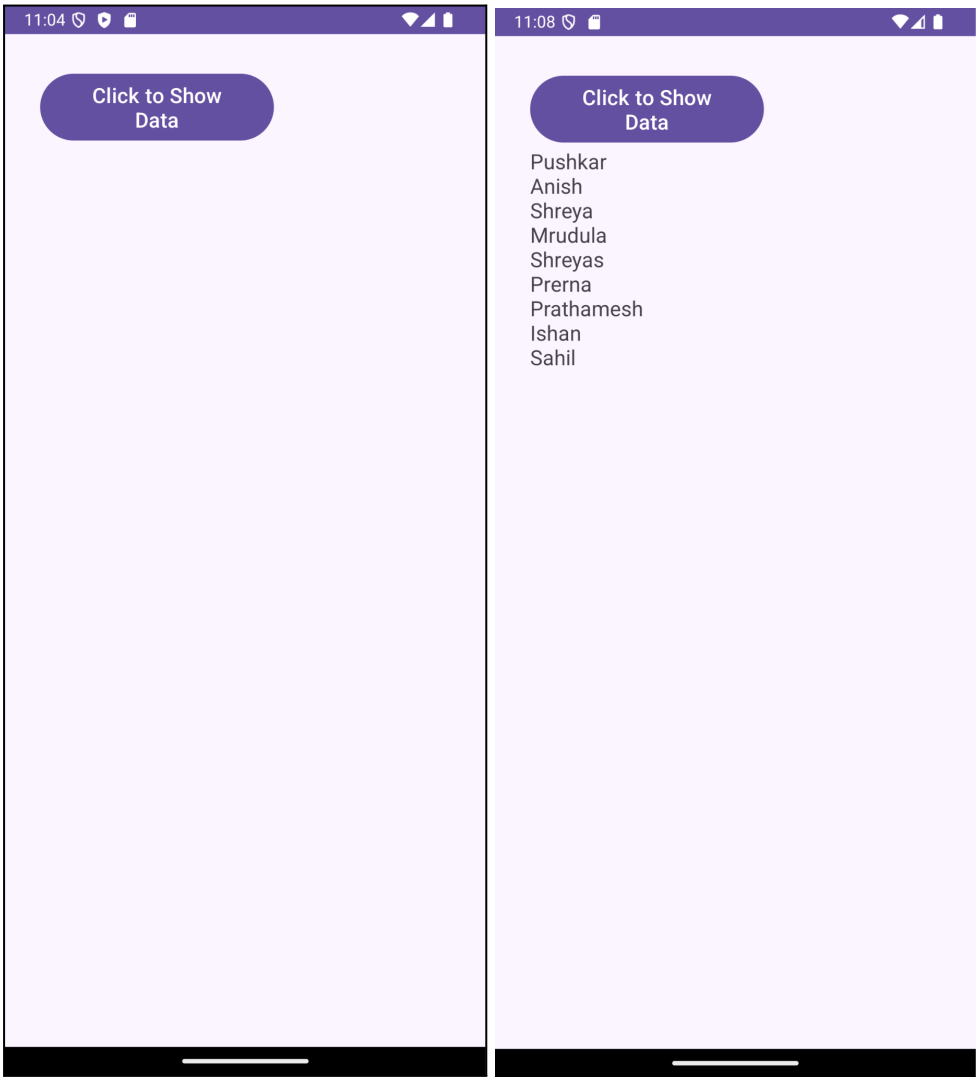
```
            StringBuilder names = new StringBuilder();
            for (int i = 0; i < namesArray.length(); i++) {
                names.append(namesArray.getString(i)).append("\n");
            }

            // Set the extracted names in the TextView
            resultsTextView.setVisibility(View.VISIBLE);
            resultsTextView.setText(names.toString());

        } catch (Exception e) {
            e.printStackTrace();
            resultsTextView.setText("Error parsing JSON.");
        }
    }
  }
}
```

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="30dp">

    <Button
        android:id="@+id/displayData"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:text="Click to Show Data"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/results"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"
```

```
    android:visibility="gone" />

</LinearLayout>
```

Output:



Conclusion:

REST APIs provide an efficient way to interact with web services using HTTP requests, with JSON as a preferred data exchange format. Using HttpURLConnection for network communication and JSON parsing techniques, developers can build Android applications that retrieve, process, and display data dynamically from remote servers. This practical introduces the foundation of working with APIs and JSON, essential for building modern Android apps.