LECTURE : 11
SHARED PREFERENCES IN ANDROID:

WHAT IS SHARED PREFERENCES:

- One of the most Interesting Data Storage options Android provides its users is Shared Preferences. Shared Preferences is the way in which one can **store and retrieve small amounts of primitive data as key/value pairs** to a file on the device storage such as String, int, float, Boolean that make up your preferences in an XML file inside the app on the device storage. Shared Preferences can be thought of as a dictionary or a key/value pair.

FEATURES OF SHARED PREFERENCES:

Simple Storage: You can store basic data types such as Boolean, float, int, long, and String.

Key-Value Pairs: Data is stored in pairs, where each value is accessed by a unique key.

Private: The data stored using Shared Preferences is private to the application, meaning no other app can directly access this data.

Persistent: Data saved in Shared Preferences persists across user sessions. Even if the application is killed or the device is restarted, this data remains saved.

COMMON USES:

- **User Settings**: For example, saving settings such as notifications turned on or off, sound settings, or display options.
- **Remembering User State**: Such as whether the user is logged in or has seen an introductory tutorial screen.
- **Storing Small Amounts of Data**: Like keeping track of the user's high score in a game or the user's last position in a list or document.

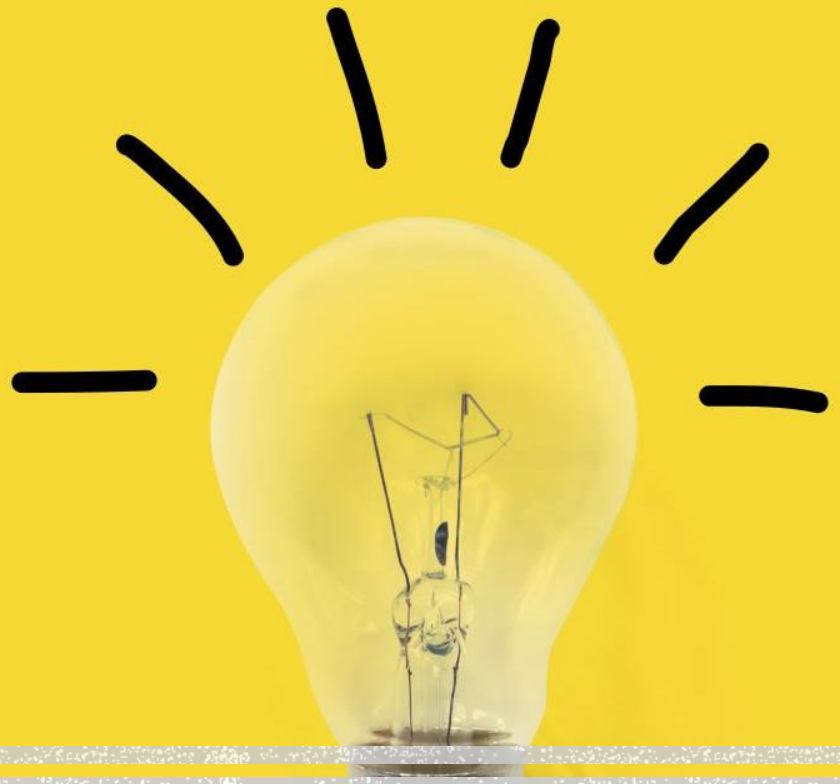
LIMITATIONS:

Not Suitable for Complex Data: It is not intended for storage of complex relational data or large datasets.

Security: The data stored is not encrypted by default, so it should not be used for storing sensitive information like passwords or personal identification numbers without proper encryption.

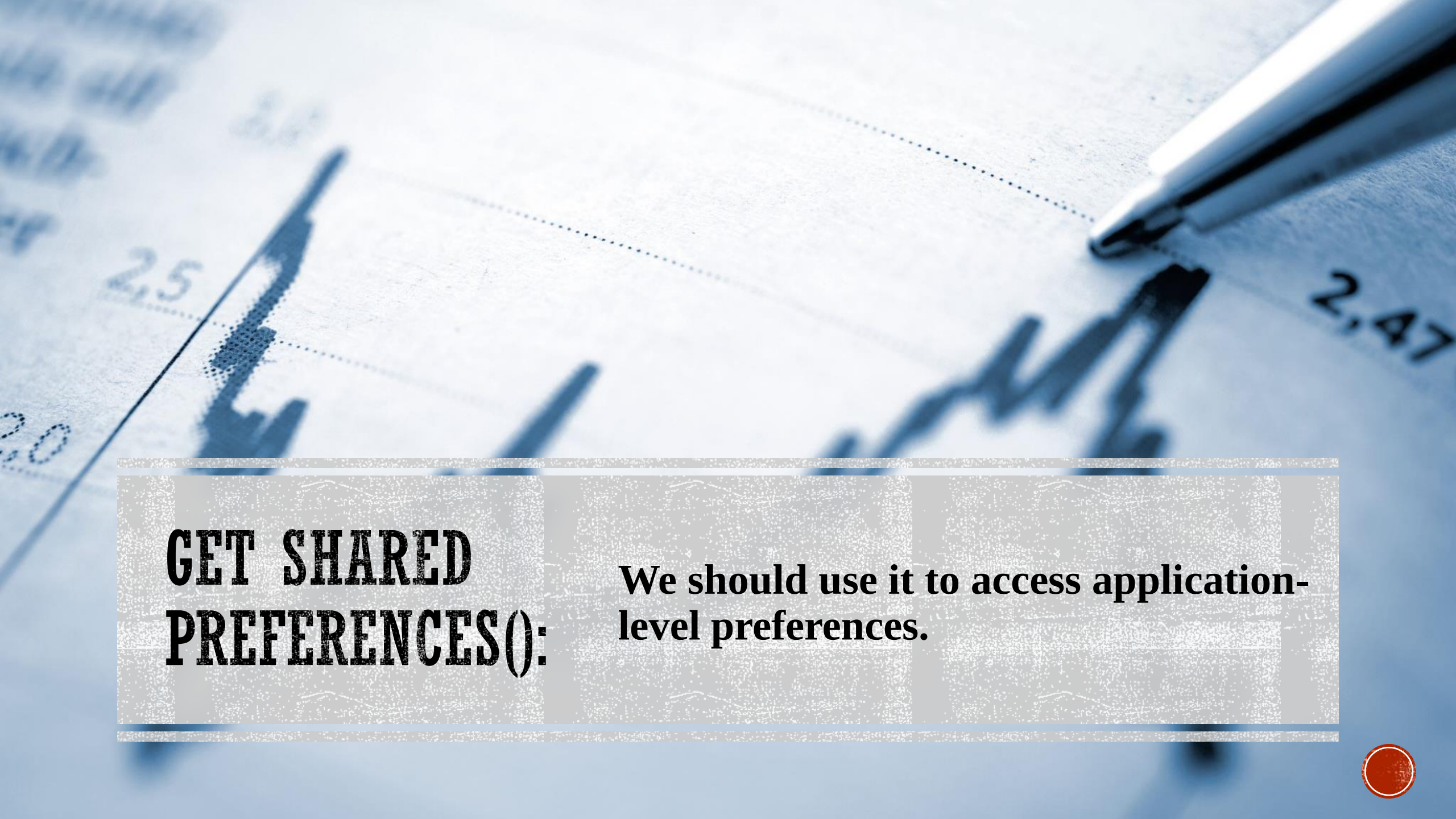
ACCESSING SHARED PREFERENCES:

Let’s see how to access shared preferences in our app. To get access to the preferences, there are three distinctive way.



GET PREFERENCE() :

We should use it to access activity-specific preferences.

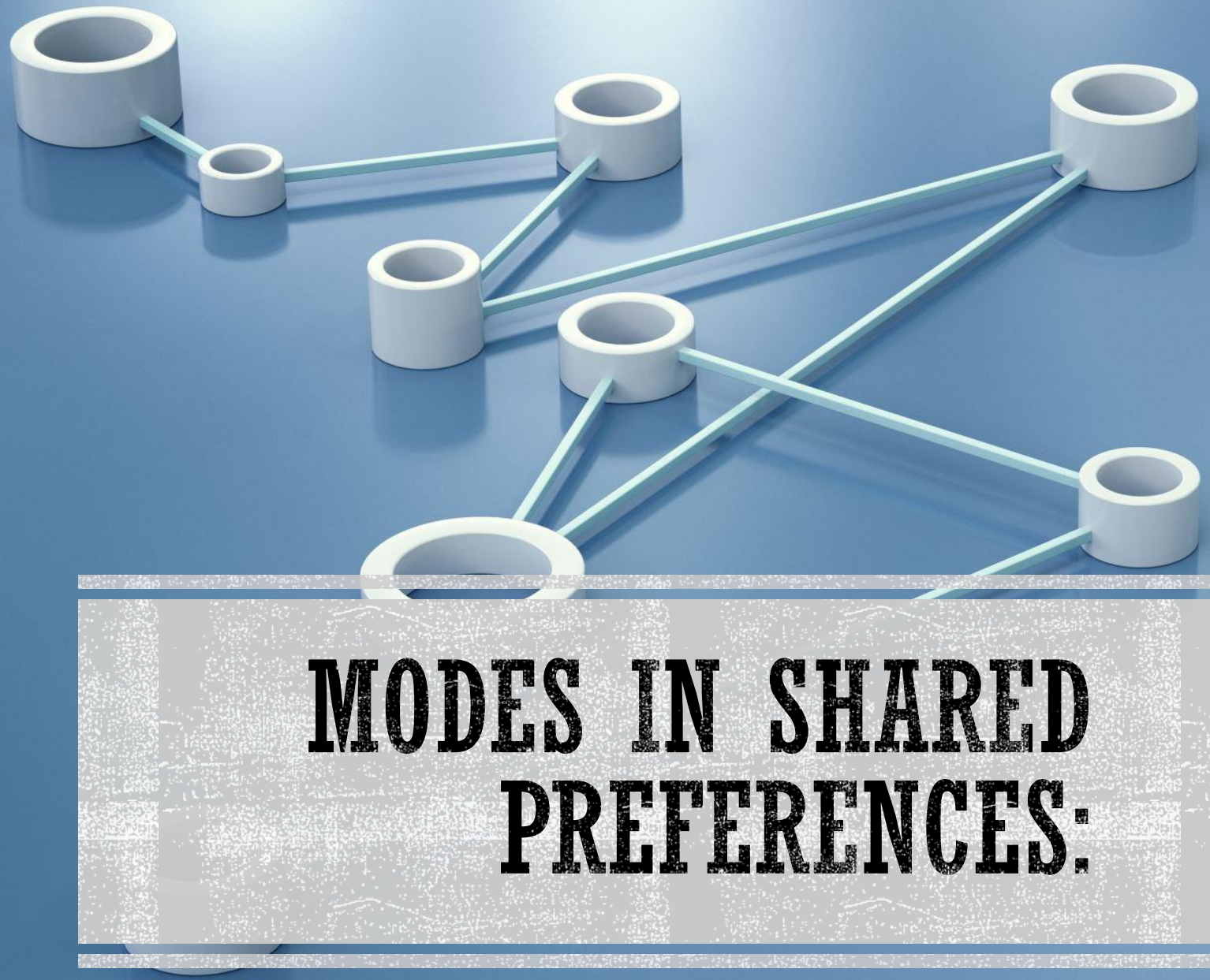


GET SHARED PREFERENCES():

We should use it to access application-level preferences.

GET DEFAULT SHARED PREFERENCES():

We should use it to get the shared preferences that Android's overall preference framework.



MODES IN SHARED PREFERENCES:

MODE_PRIVATE

MODE_PRIVATE

Description: This is the default mode. When set, the created file can only be accessed by the calling application, or all applications sharing the same user ID.

Usage: This mode is used when you want to ensure that the data saved in Shared Preferences is not accessible by any other application.

MODE_WORLD_READABLE

- **MODE_WORLD_READABLE (Deprecated)**
- **Description:** This mode allowed other applications to read the contents of the preferences file.
- **Important Note:** As of Android 7.0 (API level 24), this mode was deprecated due to security reasons. Using it can expose application data to other apps and lead to potential security breaches. It's no longer recommended or supported in newer versions of Android.

MODE_WORLD_WRITEABLE:

- **MODE_WORLD_WRITEABLE (Deprecated)**
- **Description:** This mode allowed other applications to write to the preferences file.
- **Important Note:** Like MODE_WORLD_READABLE, this mode was deprecated as of Android 7.0 (API level 24) for the same reasons. It poses a significant security risk and is not advised for use.

MODE_MULTI_PROCESS:

- **MODE_MULTI_PROCESS (Deprecated)**
- **Description:** This mode was intended for use in scenarios where multiple processes might be interacting with the same instance of SharedPreferences. It ensured that changes made in one process would immediately be visible in others.
- **Important Note:** This mode has been deprecated since API level 23 (Android M) because it was unreliable. Instead, it's recommended to use other forms of inter-process communication (IPC) or data storage.

CURRENT BEST PRACTICES:

For most applications, MODE_PRIVATE is sufficient and recommended. It ensures that your application's preferences are secure and not accessible to other applications.

```
sharedPreferences = getSharedPreferences( name: "LoginPrefs", MODE_PRIVATE);
```

CODE EXAMPLE:

LOGIN CLASS

```java
package org.hamza.shared_preferences;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;


public class LoginActivity extends AppCompatActivity {
    2 usages
    private EditText etUsername, etPassword;
    2 usages
    private Button btnLogin, btnClearPrefs;
    4 usages
    private SharedPreferences sharedPreferences;

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

```
etUsername = findViewById(R.id.et_username);
etPassword = findViewById(R.id.et_password);
btnLogin = findViewById(R.id.btn_login);
btnClearPrefs = findViewById(R.id.btn_clear_prefs);  // Make sure this button is in your layout

sharedPreferences = getSharedPreferences( name: "LoginPrefs", MODE_PRIVATE);

btnClearPrefs.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { clearPreferences(); }
});

btnLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String username = etUsername.getText().toString();
        String password = etPassword.getText().toString();

        if (validateLogin(username, password)) {
            storeLoginCredentials(username, password);
            goToMainActivity();
        } else {
            Toast.makeText( context: LoginActivity.this,  text: "Invalid login credentials", Toast.LENGTH_SHORT).show();
        }
```

```
        }
    });

    checkLoginStatus();
}

1 usage
private void clearPreferences() {
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.clear();
    editor.apply();
    Toast.makeText( context: this,  text: "Preferences cleared", Toast.LENGTH_SHORT).show();
}

1 usage
private void checkLoginStatus() {
    if (isUserLoggedIn()) {
        goToMainActivity();
    }
}

1 usage
private boolean isUserLoggedIn() { return sharedPreferences.getBoolean( key: "isLoggedIn",  defValue: false); }
```

```java
    1 usage
    private void storeLoginCredentials(String username, String password) {
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString("username", username);
        editor.putString("password", password);
        editor.putBoolean("isLoggedIn", true);
        editor.apply();
    }

    1 usage
    private boolean validateLogin(String username, String password) {
        // Simulate valid credentials
        return "admin".equals(username) && "password".equals(password);
    }

    2 usages
    private void goToMainActivity() {
        startActivity(new Intent( packageContext: LoginActivity.this, MainActivity.class));
        finish();
    }
}
```

```
        sharedPreferences = getSharedPreferences("LoginPrefs", MODE_PRIVATE);

        btnClearPrefs.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                clearPreferences();
            }
        });

        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = etUsername.getText().toString();
                String password = etPassword.getText().toString();

                if (validateLogin(username, password)) {
                    storeLoginCredentials(username, password);
                    goToMainActivity();
                } else {
                    Toast.makeText(LoginActivity.this, "Invalid login credentials", Toast.LENGTH_SHORT).show();
                }
            }
        });

        checkLoginStatus();
    }
```

In the provided code snippet, `getSharedPreferences("LoginPrefs", MODE_PRIVATE)` is used to get a reference to the shared preferences file named "LoginPrefs" with the specified mode, which in this case is `MODE_PRIVATE`. This allows the app to store and retrieve key-value pairs associated with this shared preferences file.

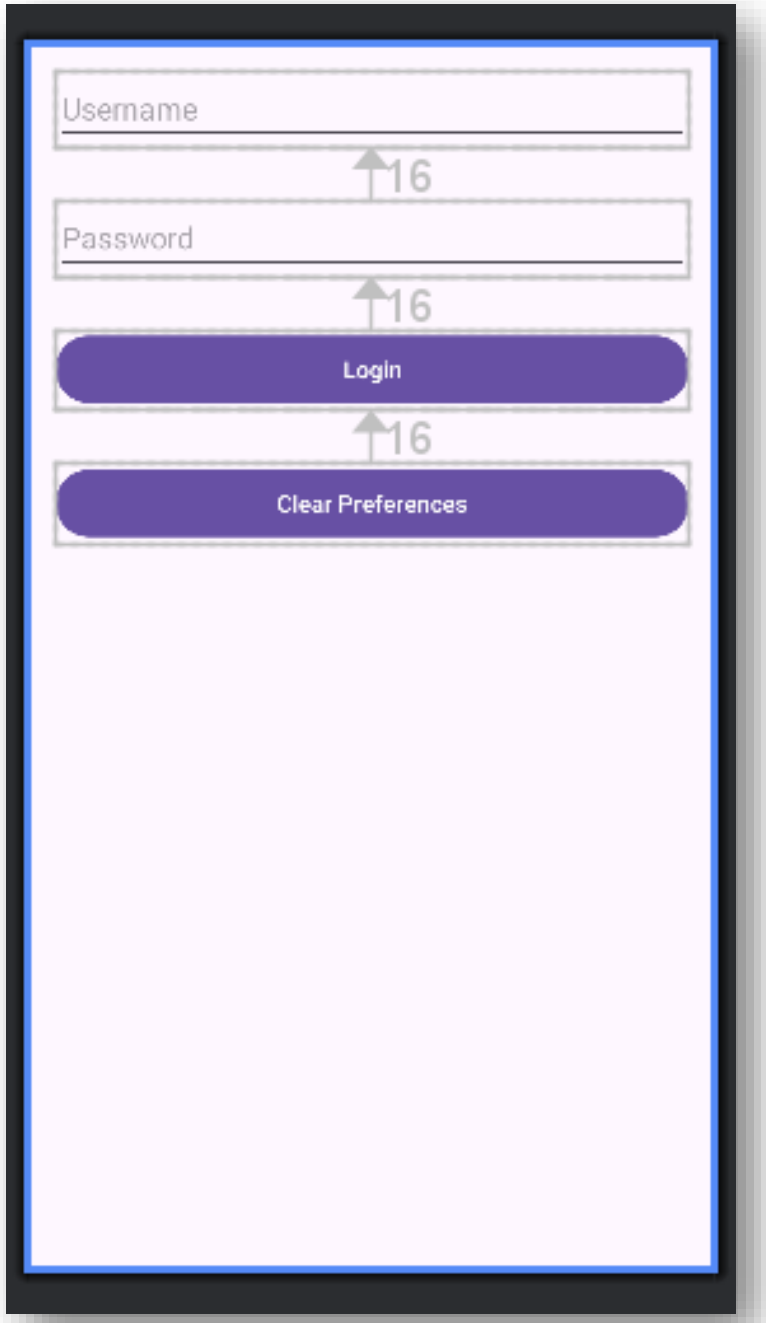
```
    private void clearPreferences() {
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.clear();
        editor.apply();
        Toast.makeText(this, "Preferences cleared", Toast.LENGTH_SHORT).show();
    }

    private void checkLoginStatus() {
        if (isUserLoggedIn()) {
            goToMainActivity();
        }
    }

    private boolean isUserLoggedIn() {
        return sharedPreferences.getBoolean("isLoggedIn", false);
    }

    private void storeLoginCredentials(String username, String password) {
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString("username", username);
        editor.putString("password", password);
        editor.putBoolean("isLoggedIn", true);
        editor.apply();
    }

    private boolean validateLogin(String username, String password) {
        // Simulate valid credentials
        return "admin".equals(username) && "password".equals(password);
    }
```

These methods are responsible for managing login credentials using SharedPreferences in an Android app. `clearPreferences()` clears stored preferences, `checkLoginStatus()` redirects to the main activity if the user is logged in, `isUserLoggedIn()` checks the login status, `storeLoginCredentials()` saves the username and password, and `validateLogin()` checks if the login credentials are valid.

LOGIN XML:

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

    <EditText
        android:id="@+id/et_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:hint="Username"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/et_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:layout_below="@id/et_username"
        android:hint="Password"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/btn_login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:layout_below="@id/et_password"
        android:text="Login" />

    <Button
        android:id="@+id/btn_clear_prefs"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:layout_below="@id/btn_login"
        android:text="Clear Preferences" />

</RelativeLayout>
```

LOGIN ACTIVITY XML:



ACTIVITY_MAIN.JAVA

```java
package org.hamza.shared_preferences;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    2 usages
    private TextView tvWelcome;
    2 usages
    private Button btnLogout;
    4 usages
    private SharedPreferences sharedPreferences;

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

MAIN.JAVA

```java
tvWelcome = findViewById(R.id.tv_welcome);
btnLogout = findViewById(R.id.btn_logout);

sharedPreferences = getSharedPreferences( name: "LoginPrefs", MODE_PRIVATE);

// Check if user is logged in
if (!sharedPreferences.getBoolean( key: "isLoggedIn",  defValue: false)) {
    // User is not logged in, redirect back to LoginActivity
    startActivity(new Intent( packageContext: this, LoginActivity.class));
    finish();
    return; // Prevent further execution of this method
}

String username = sharedPreferences.getString( key: "username",  defValue: "User");
tvWelcome.setText("Welcome, " + username + "!");

btnLogout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.clear();
        editor.apply();
        startActivity(new Intent( packageContext: MainActivity.this, LoginActivity.class));
        finish();
    }
});
```

hamza code

```java
package org.hamza.shared_preferences;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    private TextView tvWelcome;
    private Button btnLogout;
    private SharedPreferences sharedPreferences;

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

        tvWelcome = findViewById(R.id.tv_welcome);
        btnLogout = findViewById(R.id.btn_logout);

        sharedPreferences = getSharedPreferences("LoginPrefs", MODE_PRIVATE);

        // Check if user is logged in
        if (!sharedPreferences.getBoolean("isLoggedIn", false)) {
            // User is not logged in, redirect back to LoginActivity
            startActivity(new Intent(this, LoginActivity.class));
            finish();
            return; // Prevent further execution of this method
        }

        String username = sharedPreferences.getString("username", "User");
        tvWelcome.setText("Welcome, " + username + "!");

        btnLogout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.clear();
                editor.apply();
                startActivity(new Intent(MainActivity.this, LoginActivity.class));
                finish();
            }
        });
    }
}
```

This code snippet checks if the user is logged in by retrieving a boolean value from the SharedPreferences named "isLoggedIn". If the user is not logged in, it redirects the user to the LoginActivity and finishes the current MainActivity to prevent further execution.

The code snippet retrieves the username from SharedPreferences with a default value of "User" if no value is found, then sets a welcome message using the retrieved username. The OnClickListener for the logout button clears the SharedPreferences, starts the LoginActivity, and finishes the current MainActivity.

ACTIVITY_MAIN .XML

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

    <TextView
        android:id="@+id/tv_welcome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome, User!"
        android:textSize="24sp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="100dp"/>

    <Button
        android:id="@+id/btn_logout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Logout"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/tv_welcome"
        android:layout_marginTop="20dp"/>

</RelativeLayout>
```

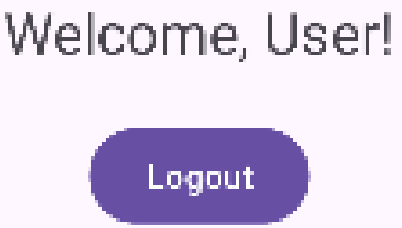
MANIFEST.XML

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Shared_Preferences"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">

        </activity>

        <activity android:name=".LoginActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

STYLES.XML

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">


    </style>
</resources>
```

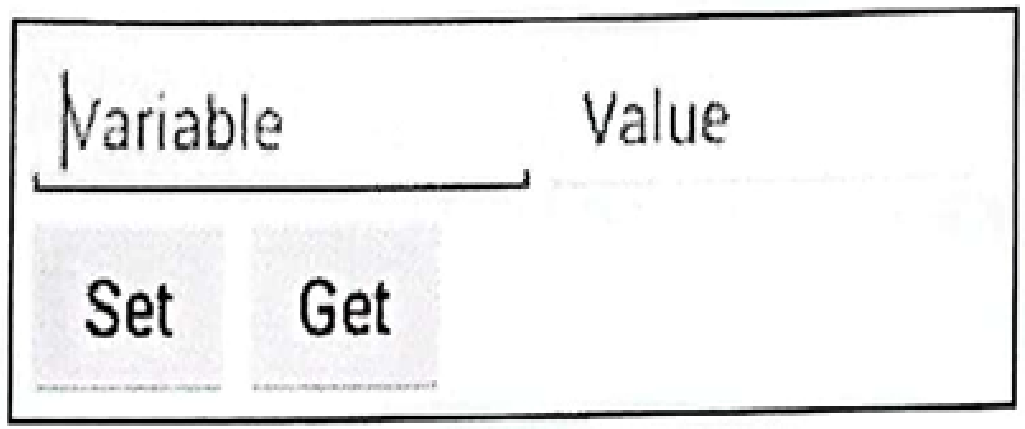
OUTPUT:



Data saved in Shared Preferences persists across user sessions. Even if the application is killed or the device is restarted, this data remains saved.

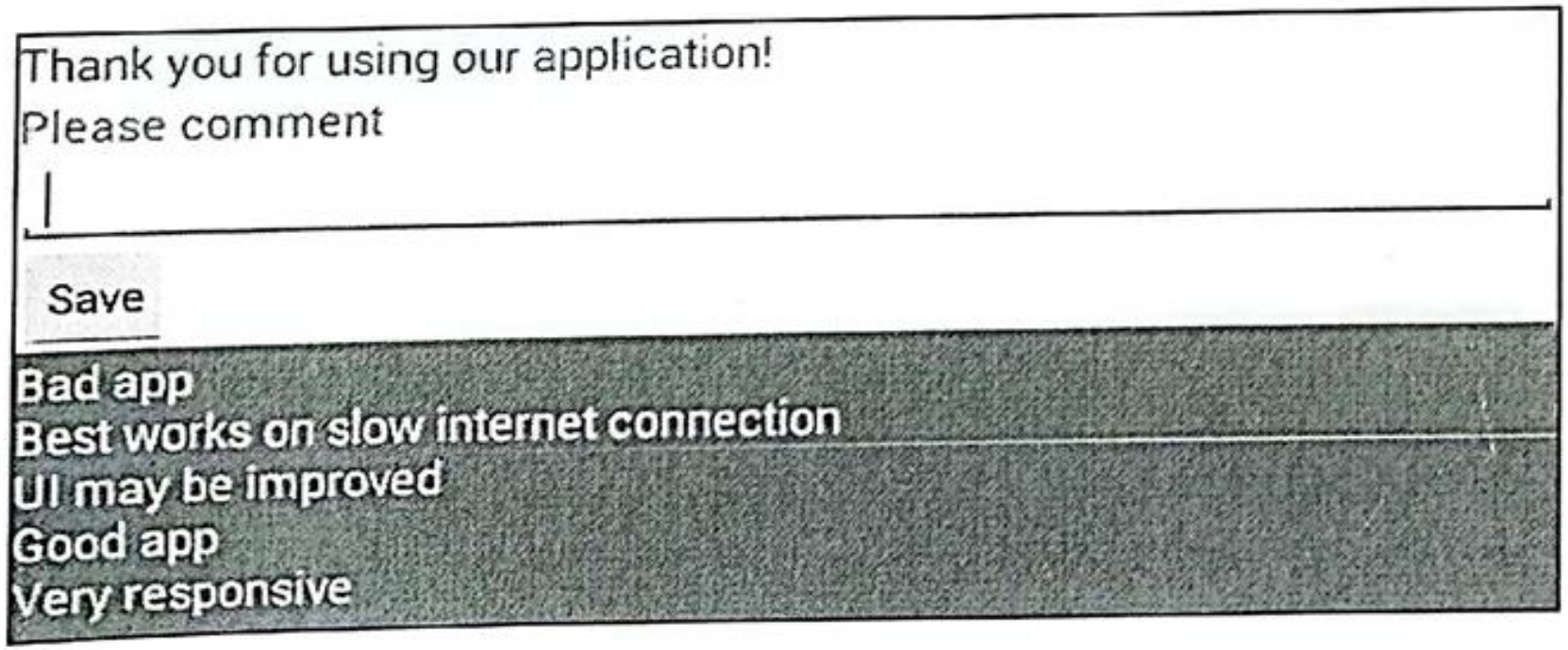
TASK 1:

1. Saving customized key-value pair: Create an app by which user can set and get customized key-value pair. You may follow the GUI given below:



TASK 2:

2. Create an app which can save user comments recorder by using SharedPreferences. User must be able to see previous comments chronologically.



TASK 3:

You are developing an Android application that requires user authentication. To ensure that the user experience is seamless, the application should remember the user's login state even after the app is closed and reopened. For simplicity and demonstration purposes, you will use SharedPreferences to store login credentials and session state.