# **1. Watermark Image Generator: Project Report**

## **Abstract**

This application is designed to provide users with the functionality to add a custom watermark to their images. Users can select an image from their device's gallery or capture a new one using the camera. After selection or capture, the app allows the user to apply a watermark (which could be text or another image) and then save the watermarked image into a specific folder within the device's gallery. This tool is ideal for protecting intellectual property or branding images before sharing.

## **Key-Words**

* Image Processing
* Watermark
* Gallery Integration
* Camera Integration
* Image Saving
* File Management

## **Modules**

The application is structured into the following key modules:

* **Main Activity:** The central activity managing UI and navigation.
* **Image Selection Module:** Handles user interaction for selecting images from the gallery or capturing from the camera.
* **Image Processing Module:** Contains the core logic for applying watermarks (text or image overlay) onto the selected image.
* **File Management Module:** Responsible for saving the processed, watermarked image to a designated folder in the device's gallery.

## **Tech Stack**

The Watermark Image Generator application is developed using the following technologies:

* **Platform:** Android
* **Programming Language:** Java
* **User Interface:** XML (for Android UI design)
* **Database:** Not applicable (data is processed and saved locally as files)
* **API:** Android Camera API, Android MediaStore API
* **Build Tool:** Gradle
* **Integrated Development Environment (IDE):** Android Studio

**Front-End:** XML **Back-End:** Java (image processing logic) **Database:** Not applicable

## **Development Process**

### **Overview**

This application enables users to add watermarks to images. The process involves selecting an image, applying the watermark, and then saving the modified image.

### **Key Components and Operations**

**UI Components (XML Example):**

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:gravity="center\_horizontal">

<Button

android:id="@+id/btn\_select\_image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Select Image from Gallery" />

<Button

android:id="@+id/btn\_capture\_image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Capture Image via Camera" />

<ImageView

android:id="@+id/image\_preview"

android:layout\_width="300dp"

android:layout\_height="300dp"

android:scaleType="fitCenter"

android:layout\_marginTop="20dp"

android:background="#EEE" />

<EditText

android:id="@+id/et\_watermark\_text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter watermark text"

android:layout\_margin="16dp" />

<Button

android:id="@+id/btn\_add\_watermark"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Add Watermark" />

<Button

android:id="@+id/btn\_save\_image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Save Watermarked Image" />

</LinearLayout>

**Code Snippets (Illustrative Java):**

Java

// In your Activity/Fragment

private static final int PICK\_IMAGE\_REQUEST = 1;

private static final int CAPTURE\_IMAGE\_REQUEST = 2;

private Uri imageUri; // Uri of the selected/captured image

private Bitmap watermarkedBitmap;

// --- Image Selection ---

// Select Image from Gallery

findViewById(R.id.btn\_select\_image).setOnClickListener(v -> {

Intent intent = new Intent(Intent.ACTION\_PICK, MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI);

startActivityForResult(intent, PICK\_IMAGE\_REQUEST);

});

// Capture Image via Camera

findViewById(R.id.btn\_capture\_image).setOnClickListener(v -> {

Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

if (takePictureIntent.resolveActivity(getPackageManager()) != null) {

// You would typically use FileProvider for saving camera output to a specific URI

// For simplicity, showing direct capture to result for now.

startActivityForResult(takePictureIntent, CAPTURE\_IMAGE\_REQUEST);

}

});

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (resultCode == RESULT\_OK && data != null) {

if (requestCode == PICK\_IMAGE\_REQUEST) {

imageUri = data.getData();

try {

Bitmap bitmap = MediaStore.Images.Media.getBitmap(this.getContentResolver(), imageUri);

((ImageView) findViewById(R.id.image\_preview)).setImageBitmap(bitmap);

// Store original bitmap for watermarking

watermarkedBitmap = bitmap.copy(Bitmap.Config.ARGB\_8888, true);

} catch (IOException e) {

e.printStackTrace();

}

} else if (requestCode == CAPTURE\_IMAGE\_REQUEST) {

Bundle extras = data.getExtras();

Bitmap imageBitmap = (Bitmap) extras.get("data");

((ImageView) findViewById(R.id.image\_preview)).setImageBitmap(imageBitmap);

watermarkedBitmap = imageBitmap.copy(Bitmap.Config.ARGB\_8888, true);

}

}

}

// --- Image Watermarking (simple text watermark) ---

findViewById(R.id.btn\_add\_watermark).setOnClickListener(v -> {

if (watermarkedBitmap != null) {

String watermarkText = ((EditText) findViewById(R.id.et\_watermark\_text)).getText().toString();

if (watermarkText.isEmpty()) {

Toast.makeText(this, "Please enter watermark text", Toast.LENGTH\_SHORT).show();

return;

}

Canvas canvas = new Canvas(watermarkedBitmap);

Paint paint = new Paint();

paint.setColor(Color.WHITE);

paint.setTextSize(50);

paint.setAntiAlias(true);

paint.setAlpha(150); // Semi-transparent

// Draw text at a position (e.g., bottom right)

float textWidth = paint.measureText(watermarkText);

float x = watermarkedBitmap.getWidth() - textWidth - 20;

float y = watermarkedBitmap.getHeight() - 20;

canvas.drawText(watermarkText, x, y, paint);

((ImageView) findViewById(R.id.image\_preview)).setImageBitmap(watermarkedBitmap);

Toast.makeText(this, "Watermark added!", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(this, "Please select an image first", Toast.LENGTH\_SHORT).show();

}

});

// --- Image Saving ---

findViewById(R.id.btn\_save\_image).setOnClickListener(v -> {

if (watermarkedBitmap != null) {

String imageName = "Watermarked\_Image\_" + System.currentTimeMillis() + ".png";

OutputStream fos;

try {

// Using MediaStore for Android Q (API 29) and above

ContentResolver resolver = getContentResolver();

ContentValues contentValues = new ContentValues();

contentValues.put(MediaStore.MediaColumns.DISPLAY\_NAME, imageName);

contentValues.put(MediaStore.MediaColumns.MIME\_TYPE, "image/png");

contentValues.put(MediaStore.MediaColumns.RELATIVE\_PATH, Environment.DIRECTORY\_PICTURES + "/WatermarkedImages"); // Custom folder

imageUri = resolver.insert(MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI, contentValues);

fos = resolver.openOutputStream(imageUri);

watermarkedBitmap.compress(Bitmap.CompressFormat.PNG, 100, fos);

if (fos != null) {

fos.close();

}

Toast.makeText(this, "Image saved to Gallery/WatermarkedImages!", Toast.LENGTH\_LONG).show();

} catch (IOException e) {

e.printStackTrace();

Toast.makeText(this, "Error saving image: " + e.getMessage(), Toast.LENGTH\_LONG).show();

}

} else {

Toast.makeText(this, "No watermarked image to save.", Toast.LENGTH\_SHORT).show();

}

});

**Permissions:**

* READ\_EXTERNAL\_STORAGE: To read images from the gallery.
* WRITE\_EXTERNAL\_STORAGE: To save the watermarked image (for API < 29). For API 29+, rely on MediaStore.
* CAMERA: To capture images using the camera.

**Enhancements:**

* Allow users to customize watermark properties: text content, font, size, color, opacity, position (top-left, center, bottom-right, etc.), and rotation.
* Enable selection of a watermark image from the gallery.
* Provide a preview of the watermarked image before saving.
* Implement a progress indicator during image processing and saving.
* Offer options to share the watermarked image directly from the app.

# **2. Custom Video Player: Project Report**

## **Abstract**

This application provides a custom video player that allows users to play video files stored internally on their device. It features essential playback controls including play, pause, next, and previous functionalities, offering a tailored media consumption experience directly from local storage.

## **Key-Words**

* Video Player
* Custom Controls
* Internal Storage
* Playback
* Media Control
* Android Media

## **Modules**

The application is structured into the following key modules:

* **Video Player Activity:** The main activity hosting the video playback and custom controls.
* **File Browser Module:** Enables users to select video files from their device's internal storage.
* **Media Control Module:** Implements the logic for play, pause, next, and previous video functionalities.

## **Tech Stack**

The Custom Video Player application is developed using the following technologies:

* **Platform:** Android
* **Programming Language:** Java
* **User Interface:** XML (for Android UI design)
* **API:** Android MediaPlayer or ExoPlayer (recommended for robust features), Android MediaStore API
* **Build Tool:** Gradle
* **Integrated Development Environment (IDE):** Android Studio

**Front-End:** XML **Back-End:** Java (media playback logic) **Database:** Not applicable

## **Development Process**

### **Overview**

This application focuses on providing a user-friendly custom video player for local video files.

### **Key Components and Operations**

**UI Components (XML Example):**

XML

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<VideoView

android:id="@+id/video\_view"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_centerInParent="true" />

<LinearLayout

android:id="@+id/controls\_layout"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:gravity="center"

android:orientation="horizontal"

android:padding="16dp"

android:background="#80000000"> <Button

android:id="@+id/btn\_previous"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Previous" />

<Button

android:id="@+id/btn\_play\_pause"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Play"

android:layout\_marginStart="16dp"

android:layout\_marginEnd="16dp"/>

<Button

android:id="@+id/btn\_next"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Next" />

</LinearLayout>

</RelativeLayout>

**Code Snippets (Illustrative Java):**

Java

// In your Activity/Fragment

private static final int SELECT\_VIDEO\_REQUEST = 3;

private VideoView videoView;

private Button playPauseButton;

private MediaController mediaController;

private List<Uri> videoPlaylist; // Assume this is populated with video URIs

private int currentVideoIndex = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_video\_player);

videoView = findViewById(R.id.video\_view);

playPauseButton = findViewById(R.id.btn\_play\_pause);

Button prevButton = findViewById(R.id.btn\_previous);

Button nextButton = findViewById(R.id.btn\_next);

// Initialize MediaController (can be customized further)

mediaController = new MediaController(this);

mediaController.setAnchorView(videoView);

videoView.setMediaController(mediaController);

// Populate dummy playlist for demonstration

videoPlaylist = new ArrayList<>();

// Add dummy URIs - in a real app, these would come from user selection or database

// videoPlaylist.add(Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.sample\_video\_1));

// videoPlaylist.add(Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.sample\_video\_2));

// --- Video Selection ---

// Example: Trigger video selection on app start or via a button

if (videoPlaylist.isEmpty()) { // If no default playlist, prompt user to select

Intent intent = new Intent(Intent.ACTION\_GET\_CONTENT);

intent.setType("video/\*");

startActivityForResult(intent, SELECT\_VIDEO\_REQUEST);

} else {

playVideo(videoPlaylist.get(currentVideoIndex));

}

// --- Play/Pause Control ---

playPauseButton.setOnClickListener(v -> {

if (videoView.isPlaying()) {

videoView.pause();

playPauseButton.setText("Play");

} else {

videoView.start();

playPauseButton.setText("Pause");

}

});

// --- Next/Previous Controls ---

nextButton.setOnClickListener(v -> playNextVideo());

prevButton.setOnClickListener(v -> playPreviousVideo());

// Listener for video completion

videoView.setOnCompletionListener(mp -> {

Toast.makeText(this, "Video finished", Toast.LENGTH\_SHORT).show();

playNextVideo(); // Play next video automatically

});

}

private void playVideo(Uri videoUri) {

videoView.setVideoURI(videoUri);

videoView.start();

playPauseButton.setText("Pause");

}

private void playNextVideo() {

if (!videoPlaylist.isEmpty()) {

currentVideoIndex = (currentVideoIndex + 1) % videoPlaylist.size();

playVideo(videoPlaylist.get(currentVideoIndex));

} else {

Toast.makeText(this, "No more videos in playlist", Toast.LENGTH\_SHORT).show();

}

}

private void playPreviousVideo() {

if (!videoPlaylist.isEmpty()) {

currentVideoIndex = (currentVideoIndex - 1 + videoPlaylist.size()) % videoPlaylist.size();

playVideo(videoPlaylist.get(currentVideoIndex));

} else {

Toast.makeText(this, "No previous videos in playlist", Toast.LENGTH\_SHORT).show();

}

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (resultCode == RESULT\_OK && data != null && requestCode == SELECT\_VIDEO\_REQUEST) {

Uri selectedVideoUri = data.getData();

if (selectedVideoUri != null) {

videoPlaylist.clear(); // Clear existing playlist

videoPlaylist.add(selectedVideoUri);

currentVideoIndex = 0;

playVideo(selectedVideoUri);

}

}

}

**Permissions:**

* READ\_EXTERNAL\_STORAGE: To access video files stored on the device.

**Enhancements:**

* Implement a custom progress bar and time display.
* Add volume control (e.g., using a SeekBar).
* Support full-screen toggle.
* Create a playlist functionality to easily switch between multiple videos.
* Implement gesture controls (e.g., swipe for seek, double-tap for play/pause).
* Consider using ExoPlayer for better performance, wider format support, and advanced features.

# **3. Signup With Validation: Project Report**

## **Abstract**

This application features a comprehensive user signup page designed to collect various user details through a rich set of Android widgets. It incorporates robust validation rules to ensure data integrity and user input correctness. Upon successful validation, the application seamlessly redirects the user to a subsequent screen, displaying all the data they have entered during the signup process, providing immediate feedback and confirmation.

## **Key-Words**

* Signup Page
* Form Validation
* EditText
* AutoCompleteTextView
* Radio Button
* Checkbox
* Spinner
* Button
* Data Display
* User Onboarding

## **Modules**

The application is structured into the following key modules:

* **Signup Activity/Fragment:** The primary UI component for user registration, containing all input widgets.
* **Validation Logic Module:** Implements all the rules and checks for validating user input across various fields.
* **Data Display Activity:** A subsequent screen responsible for receiving and displaying the validated user data.

## **Tech Stack**

The Signup With Validation application is developed using the following technologies:

* **Platform:** Android
* **Programming Language:** Java
* **User Interface:** XML (for Android UI design)
* **Database:** Not explicitly required for this scope (data passed via Intent), but could be extended to use SQLite for persistence.
* **Build Tool:** Gradle
* **Integrated Development Environment (IDE):** Android Studio

**Front-End:** XML **Back-End:** Java (validation logic, data handling) **Database:** Not applicable (data passed directly between activities)

## **Development Process**

### **Overview**

This application focuses on a robust user signup process with extensive input validation and immediate data display.

### **Key Components and Operations**

**UI Components (XML Example):**

XML

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:padding="16dp">

<EditText

android:id="@+id/et\_name"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Full Name"

android:inputType="textPersonName" />

<EditText

android:id="@+id/et\_email"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Email"

android:inputType="textEmailAddress" />

<EditText

android:id="@+id/et\_password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Password"

android:inputType="textPassword" />

<EditText

android:id="@+id/et\_confirm\_password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Confirm Password"

android:inputType="textPassword" />

<AutoCompleteTextView

android:id="@+id/actv\_city"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="City" />

<RadioGroup

android:id="@+id/rg\_gender"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginTop="10dp">

<RadioButton android:id="@+id/radio\_male" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Male" />

<RadioButton android:id="@+id/radio\_female" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Female" />

</RadioGroup>

<Spinner

android:id="@+id/spinner\_country"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dp" />

<CheckBox

android:id="@+id/cb\_terms"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="I agree to the Terms and Conditions"

android:layout\_marginTop="10dp" />

<Button

android:id="@+id/btn\_signup"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Sign Up"

android:layout\_marginTop="20dp" />

</LinearLayout>

</ScrollView>

**Code Snippets (Illustrative Java):**

Java

// In your SignupActivity

private EditText etName, etEmail, etPassword, etConfirmPassword;

private AutoCompleteTextView actvCity;

private RadioGroup rgGender;

private Spinner spinnerCountry;

private CheckBox cbTerms;

private Button btnSignup;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_signup);

etName = findViewById(R.id.et\_name);

etEmail = findViewById(R.id.et\_email);

etPassword = findViewById(R.id.et\_password);

etConfirmPassword = findViewById(R.id.et\_confirm\_password);

actvCity = findViewById(R.id.actv\_city);

rgGender = findViewById(R.id.rg\_gender);

spinnerCountry = findViewById(R.id.spinner\_country);

cbTerms = findViewById(R.id.cb\_terms);

btnSignup = findViewById(R.id.btn\_signup);

// Setup AutoCompleteTextView (Example with ArrayAdapter)

String[] cities = {"New York", "Los Angeles", "Chicago", "Houston", "Phoenix"};

ArrayAdapter<String> cityAdapter = new ArrayAdapter<>(this, android.R.layout.simple\_dropdown\_item\_1line, cities);

actvCity.setAdapter(cityAdapter);

// Setup Spinner (Example with ArrayAdapter)

String[] countries = {"USA", "Canada", "UK", "Australia", "India"};

ArrayAdapter<String> countryAdapter = new ArrayAdapter<>(this, android.R.layout.simple\_spinner\_dropdown\_item, countries);

spinnerCountry.setAdapter(countryAdapter);

btnSignup.setOnClickListener(v -> {

if (validateForm()) {

// Data is valid, proceed to next screen and display data

Intent intent = new Intent(SignupActivity.this, DisplayDataActivity.class);

intent.putExtra("name", etName.getText().toString());

intent.putExtra("email", etEmail.getText().toString());

intent.putExtra("city", actvCity.getText().toString());

// Get selected radio button text

int selectedGenderId = rgGender.getCheckedRadioButtonId();

if (selectedGenderId != -1) {

RadioButton selectedGender = findViewById(selectedGenderId);

intent.putExtra("gender", selectedGender.getText().toString());

}

intent.putExtra("country", spinnerCountry.getSelectedItem().toString());

// Pass other data as needed

startActivity(intent);

}

});

}

private boolean validateForm() {

boolean isValid = true;

// --- Empty Field Check ---

if (etName.getText().toString().trim().isEmpty()) {

etName.setError("Name is required");

isValid = false;

} else {

etName.setError(null);

}

if (etEmail.getText().toString().trim().isEmpty()) {

etEmail.setError("Email is required");

isValid = false;

} else {

etEmail.setError(null);

}

if (etPassword.getText().toString().trim().isEmpty()) {

etPassword.setError("Password is required");

isValid = false;

} else {

etPassword.setError(null);

}

if (etConfirmPassword.getText().toString().trim().isEmpty()) {

etConfirmPassword.setError("Confirm Password is required");

isValid = false;

} else {

etConfirmPassword.setError(null);

}

// --- Email Format Validation ---

String emailPattern = "[a-zA-Z0-9.\_-]+@[a-z]+\\.+[a-z]+";

if (!etEmail.getText().toString().matches(emailPattern)) {

etEmail.setError("Invalid email format");

isValid = false;

}

// --- Password Matching ---

if (!etPassword.getText().toString().equals(etConfirmPassword.getText().toString())) {

etConfirmPassword.setError("Passwords do not match");

isValid = false;

}

// --- Checkbox Validation ---

if (!cbTerms.isChecked()) {

Toast.makeText(this, "Please agree to the terms and conditions", Toast.LENGTH\_SHORT).show();

isValid = false;

}

// --- Radio Button Validation ---

if (rgGender.getCheckedRadioButtonId() == -1) {

Toast.makeText(this, "Please select a gender", Toast.LENGTH\_SHORT).show();

isValid = false;

}

return isValid;

}

**Permissions:**

* No specific permissions are typically required for a local signup form and data display.

**Enhancements:**

* Implement real-time validation feedback (e.g., error messages appearing as the user types).
* Provide clear and user-friendly error messages for each validation failure.
* Add a password strength indicator.
* Use a DatePickerDialog for date input fields.
* Implement a custom dialog for displaying validation errors instead of just toasts or setError().
* Encrypt sensitive data (like password) before passing it via Intent or storing it, even temporarily.

# **4. Session Management: Project Report**

## **Abstract**

This application demonstrates a fundamental session management system across three core screens: Splash, Login, and Home. It ensures a seamless user experience by intelligently directing users based on their login status. The system utilizes Shared Preferences for persistent storage of session data, allowing the application to remember logged-in users and manage their access, including a clear logout mechanism.

## **Key-Words**

* Session Management
* Splash Screen
* Login Screen
* Home Screen
* Shared Preferences
* User Authentication
* Logout

## **Modules**

The application is structured into the following key modules:

* **Splash Activity:** The entry point of the application, responsible for initial session check.
* **Login Activity:** Handles user authentication and stores session data.
* **Home Activity:** The main content screen accessible after successful login, providing a logout option.
* **Session Manager (Conceptual):** A utility class or set of methods to manage Shared Preferences for session data.

## **Tech Stack**

The Session Management application is developed using the following technologies:

* **Platform:** Android
* **Programming Language:** Java
* **User Interface:** XML (for Android UI design)
* **Storage:** Android Shared Preferences
* **Build Tool:** Gradle
* **Integrated Development Environment (IDE):** Android Studio

**Front-End:** XML **Back-End:** Java (session logic) **Database:** Not applicable (Shared Preferences used for session data)

## **Development Process**

### **Overview**

This application implements a common mobile app flow: Splash -> Login/Home based on session, with robust session management using Shared Preferences.

### **Key Components and Operations**

**UI Components (XML Examples):**

XML

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#FF5722"> <ImageView

android:id="@+id/splash\_logo"

android:layout\_width="150dp"

android:layout\_height="150dp"

android:layout\_centerInParent="true"

android:src="@mipmap/ic\_launcher\_round" /> <ProgressBar

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/splash\_logo"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="20dp"

android:indeterminateTint="@android:color/white" />

</RelativeLayout>

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:gravity="center"

android:padding="24dp">

<EditText

android:id="@+id/et\_email"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Email"

android:inputType="textEmailAddress" />

<EditText

android:id="@+id/et\_password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Password"

android:inputType="textPassword"

android:layout\_marginTop="16dp" />

<CheckBox

android:id="@+id/cb\_terms\_conditions"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Agree to Terms &amp; Conditions"

android:layout\_marginTop="16dp" />

<Button

android:id="@+id/btn\_login"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login"

android:layout\_marginTop="24dp" />

</LinearLayout>

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:gravity="center\_horizontal"

android:padding="16dp">

<TextView

android:id="@+id/tv\_welcome\_message"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Welcome!"

android:textSize="24sp"

android:textStyle="bold"

android:layout\_marginBottom="20dp" />

<TextView

android:id="@+id/tv\_session\_email"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Logged in as: "

android:textSize="18sp" />

<Button

android:id="@+id/btn\_logout"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Logout"

android:layout\_marginTop="40dp" />

</LinearLayout>

**Code Snippets (Illustrative Java):**

Java

// --- SessionManager Helper Class ---

public class SessionManager {

private SharedPreferences pref;

private SharedPreferences.Editor editor;

private static final String PREF\_NAME = "MyAppSession";

private static final String IS\_LOGIN = "IsLoggedIn";

public static final String KEY\_EMAIL = "email";

public SessionManager(Context context) {

pref = context.getSharedPreferences(PREF\_NAME, Context.MODE\_PRIVATE);

editor = pref.edit();

}

public void createLoginSession(String email) {

editor.putBoolean(IS\_LOGIN, true);

editor.putString(KEY\_EMAIL, email);

editor.commit(); // Use apply() for async save

}

public boolean isLoggedIn() {

return pref.getBoolean(IS\_LOGIN, false);

}

public String getUserEmail() {

return pref.getString(KEY\_EMAIL, null);

}

public void logoutUser() {

editor.clear();

editor.commit(); // Use apply() for async clear

}

}

// --- SplashActivity.java ---

public class SplashActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_splash);

new Handler().postDelayed(() -> {

SessionManager sessionManager = new SessionManager(SplashActivity.this);

if (sessionManager.isLoggedIn()) {

startActivity(new Intent(SplashActivity.this, HomeActivity.class));

} else {

startActivity(new Intent(SplashActivity.this, LoginActivity.class));

}

finish();

}, 3000); // 3 seconds delay

}

}

// --- LoginActivity.java ---

public class LoginActivity extends AppCompatActivity {

private EditText etEmail, etPassword;

private CheckBox cbTerms;

private Button btnLogin;

private SessionManager sessionManager;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

sessionManager = new SessionManager(this);

etEmail = findViewById(R.id.et\_email);

etPassword = findViewById(R.id.et\_password);

cbTerms = findViewById(R.id.cb\_terms\_conditions);

btnLogin = findViewById(R.id.btn\_login);

btnLogin.setOnClickListener(v -> {

String email = etEmail.getText().toString().trim();

String password = etPassword.getText().toString().trim();

if (email.isEmpty() || password.isEmpty()) {

Toast.makeText(this, "Please enter email and password", Toast.LENGTH\_SHORT).show();

} else if (!cbTerms.isChecked()) {

Toast.makeText(this, "Please agree to terms and conditions", Toast.LENGTH\_SHORT).show();

} else {

// Simulate login success (replace with actual authentication logic)

if (email.equals("test@example.com") && password.equals("password123")) {

sessionManager.createLoginSession(email);

startActivity(new Intent(LoginActivity.this, HomeActivity.class));

finish();

} else {

Toast.makeText(this, "Invalid credentials", Toast.LENGTH\_SHORT).show();

}

}

});

}

}

// --- HomeActivity.java ---

public class HomeActivity extends AppCompatActivity {

private SessionManager sessionManager;

private TextView tvSessionEmail;

private Button btnLogout;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_home);

sessionManager = new SessionManager(this);

tvSessionEmail = findViewById(R.id.tv\_session\_email);

btnLogout = findViewById(R.id.btn\_logout);

tvSessionEmail.setText("Logged in as: " + sessionManager.getUserEmail());

btnLogout.setOnClickListener(v -> {

sessionManager.logoutUser();

// Redirect to LoginActivity and clear back stack

Intent intent = new Intent(HomeActivity.this, LoginActivity.class);

intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP | Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intent);

finish();

});

}

}

**Permissions:**

* No specific permissions are typically required for this core session management functionality.

**Enhancements:**

* Implement proper user authentication against a backend API instead of just local simulation.
* Encrypt sensitive data stored in SharedPreferences.
* Add a "Remember Me" checkbox that persists login credentials (or a token) for a longer duration.
* Provide visual feedback (e.g., a loading spinner) during the splash screen's session check.
* Handle network connectivity issues for actual login.

# **5. External Share Image Functionality: Project Report**

## **Abstract**

This application provides a straightforward utility for users to select an image from their device's gallery, display it within the app, and then seamlessly share it to popular social media platforms such as WhatsApp, Facebook, or Instagram, based on user selection. It simplifies the process of sharing visual content directly from the application.

## **Key-Words**

* Image Sharing
* Gallery Selection
* WhatsApp
* Facebook
* Instagram
* Android Intent
* External Sharing

## **Modules**

The application is structured into the following key modules:

* **Image Selector Activity:** The main activity responsible for allowing users to select an image and displaying it.
* **Share Functionality Module:** Contains the logic for constructing and executing the appropriate Android Intents for sharing to external applications.

## **Tech Stack**

The External Share Image Functionality application is developed using the following technologies:

* **Platform:** Android
* **Programming Language:** Java
* **User Interface:** XML (for Android UI design)
* **API:** Android Intent system, Android MediaStore API
* **Build Tool:** Gradle
* **Integrated Development Environment (IDE):** Android Studio

**Front-End:** XML **Back-End:** Java (sharing logic) **Database:** Not applicable

## **Development Process**

### **Overview**

This application focuses on integrating external image sharing capabilities, allowing users to select an image and share it to various social media platforms.

### **Key Components and Operations**

**UI Components (XML Example):**

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:gravity="center\_horizontal"

android:padding="16dp">

<Button

android:id="@+id/btn\_select\_image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Select Image from Gallery" />

<ImageView

android:id="@+id/image\_to\_share\_preview"

android:layout\_width="300dp"

android:layout\_height="300dp"

android:scaleType="fitCenter"

android:layout\_marginTop="20dp"

android:background="#EEE"

android:src="@android:drawable/ic\_menu\_gallery" /> <Button

android:id="@+id/btn\_share\_image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Share Image"

android:layout\_marginTop="30dp" />

</LinearLayout>

**Code Snippets (Illustrative Java):**

Java

// In your Activity/Fragment

private static final int PICK\_IMAGE\_REQUEST = 4;

private Uri selectedImageUri;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_share\_image);

findViewById(R.id.btn\_select\_image).setOnClickListener(v -> {

Intent intent = new Intent(Intent.ACTION\_PICK, MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI);

startActivityForResult(intent, PICK\_IMAGE\_REQUEST);

});

findViewById(R.id.btn\_share\_image).setOnClickListener(v -> {

if (selectedImageUri != null) {

showShareOptionsDialog();

} else {

Toast.makeText(this, "Please select an image first", Toast.LENGTH\_SHORT).show();

}

});

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (resultCode == RESULT\_OK && data != null && requestCode == PICK\_IMAGE\_REQUEST) {

selectedImageUri = data.getData();

if (selectedImageUri != null) {

((ImageView) findViewById(R.id.image\_to\_share\_preview)).setImageURI(selectedImageUri);

}

}

}

private void showShareOptionsDialog() {

final CharSequence[] options = {"WhatsApp", "Facebook", "Instagram", "Other Apps"};

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Share Image To:");

builder.setItems(options, (dialog, item) -> {

Intent shareIntent = new Intent(Intent.ACTION\_SEND);

shareIntent.setType("image/\*");

shareIntent.putExtra(Intent.EXTRA\_STREAM, selectedImageUri);

shareIntent.addFlags(Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION); // Grant temporary permission

String packageName = null;

switch (item) {

case 0: // WhatsApp

packageName = "com.whatsapp";

break;

case 1: // Facebook

packageName = "com.facebook.katana"; // Or com.facebook.lite for Facebook Lite

break;

case 2: // Instagram

packageName = "com.instagram.android";

break;

case 3: // Other Apps (Chooser)

startActivity(Intent.createChooser(shareIntent, "Share Image Using..."));

return; // Don't proceed with specific package

}

if (packageName != null) {

shareIntent.setPackage(packageName);

try {

startActivity(shareIntent);

} catch (ActivityNotFoundException e) {

Toast.makeText(this, options[item] + " app not installed.", Toast.LENGTH\_SHORT).show();

}

}

});

builder.show();

}