“ IMAGE STEGANOGRAPHY”

A Report submitted under Project-Based Learning

In Partial Fulfillment of the Course Requirements for
“ MOBILE APPLICATION DEVELOPMENT(22CS104002)”

Submitted By

GHARSHITHA	**22102A040797**
GMAHESWARI	**22102A040806**
GNISHIKA	**22102A040804**
GVENKATANANDINI	**22102A040783**
GSATWIKA	**22102A040784**
JRACHANA	**22102A040827**

Under the Guidance of

MOHAMED MUZAMIL
Department of CSE

Department of Computer Science and Engineering

School of Computing

MOHAN BABU UNIVERSITY

Sree Sainath Nagar, Tirupati – 517102

2024-2025

MOHAN BABU UNIVERSITY

Vision

To be a globally respected institution with an innovative and entrepreneurial culture that offers transformative education to advance sustainability and societal good.

Mission

- ❖ Develop industry-focused professionals with a global perspective.
- ❖ Offer academic programs that provide transformative learning experience founded on the spirit of curiosity, innovation, and integrity.
- ❖ Create confluence of research, innovation, and ideation to bring about sustainable and socially relevant enterprises.
- ❖ Uphold high standards of professional ethics leading to harmonious relationship with environment and society.

SCHOOL OF COMPUTING

Vision

To lead the advancement of computer science research and education that has real-world impact and to push the frontiers of innovation in the field.

Mission

- ❖ Instil within our students fundamental computing knowledge, a broad set of skills, and an inquisitive attitude to create innovative solutions to serve industry and community.
- ❖ Provide an experience par excellence with our state-of-the-art research, innovation, and incubation ecosystem to realise our learners' fullest potential.
- ❖ Impart continued education and research support to working professionals in the computing domain to enhance their expertise in the cutting-edge technologies.

- ❖ Inculcate among the computing engineers of tomorrow with a spirit to solve societal challenges.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision

To become a Centre of Excellence in Computer Science and its emerging areas by imparting high quality education through teaching, training and research.

Mission

- ► Imparting quality education in Computer Science and Engineering and emerging areas of IT industry by disseminating knowledge through contemporary curriculum, competent faculty and effective teaching-learning methodologies.
- ► Nurture research, innovation and entrepreneurial skills among faculty and students to contribute to the needs of industry and society.
- ► Inculcate professional attitude, ethical and social responsibilities for prospective and promising engineering profession.
- ► Encourage students to engage in life-long learning by creating awareness of the contemporary developments in Computer Science and Engineering and its emerging areas.

B.Tech. Computer Science and Engineering

PROGRAM EDUCATIONAL OBJECTIVES

After few years of graduation, the graduates of B.Tech. CSE will be:

PEO1. Pursuing higher studies in core, specialized or allied areas of Computer Science, or Management.

PEO2. Employed in reputed Computer and I.T organizations or Government to have a globally competent professional career in Computer Science and Engineering domain or be successful Entrepreneurs.

PEO3. Able to demonstrate effective communication, engage in teamwork, exhibit leadership skills and ethical attitude, and achieve professional advancement through continuing education.

PROGRAM OUTCOMES

On successful completion of the Program, the graduates of B.Tech. CSE Program will be able to:

PO1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem Analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct Investigations of Complex Problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern Tool Usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **The Engineer and Society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and Sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project Management and Finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one' s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. **Life-long Learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

On successful completion of the Program, the graduates of B. Tech. (CSE) program will be able to:

PSO1. Apply knowledge of computer science engineering, Use modern tools, techniques and technologies for efficient design and development of computer-based systems for complex engineering problems.

PSO2. Design and deploy networked systems using standards and principles, evaluate security measures for complex networks, apply procedures and tools to solve networking issues.

PSO3. Develop intelligent systems by applying adaptive algorithms and methodologies for solving problems from inter-disciplinary domains.

PSO4. Apply suitable models, tools and techniques to perform data analytics for effective decision making.

PROGRAM ELECTIVE

Course Code	Course Title	L	T	P	S	C
22CS104002	MOBILE APPLICATION DEVELOPMENT	3	-	2	4	5

Pre-Requisite - Object Oriented Programming through Java

Anti-Requisite -

Co-Requisite -

COURSE DESCRIPTION: Mobile platforms; Mobile User Interface and tools; Introduction to Android; Activities; Views; Menus; Database Storage; SMS; e-mail; Displaying Maps; Building a Location Tracker Web Services Using HTTP; Sockets Programming; Communication between a Service and an Activity; Introduction to iOS.

COURSE OUTCOMES: *After successful completion of this course, the students will be able to:*

CO1. Demonstrate knowledge on mobile platforms, mobile user interface and user interface design requirements.

CO2. Design user interfaces by analyzing user requirements.

CO3. Develop mobile applications for Messaging, Location-Based Services, And Networking.

CO4. Develop mobile applications and publish in different mobile platforms.

CO-PO -PSO Mapping Table:

Course Outcome	Program Outcomes												Program Specific Outcomes			
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	3												3			
CO2	1	2	3	2									3			
CO3	1	2	2	2	3	2	2	1					3			2
CO4	1	2	3	2	3	2	2	1					3			
Course Correlation Mapping	**3**	**2**	**3**	**2**	**3**	**2**	**2**	**1**	-	**-**	**-**	**-**	**3**	-	**-**	2

Correlation Level: 3-High; 2 -Medium; 1 -Low

MOHAN BABU UNIVERSITY

Sree Sainath Nagar, Tirupati 517 102

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Project Entitled

" **IMAGE STEGANOGRAPHY** "

Submitted By

G HARSHITHA	22102A040797
G MAHESWARI	22102A040806
G NISHIKA	22102A040804
G VENKATA NANDINI	22102A040783
G SATWIKA	22102A040784
J RACHANA	22102A040827

is the work submitted under Project-Based Learning in Partial Fulfillment of the Course Requirements for " MOBILE APPLICATION DEVELOPMENT(22CS104002)" during 2024-2025.

Supervisor:

MOHAMED MUZAMIL
Department of CSE
School of Computing
Mohan Babu University
Tirupati.

Head:

Dr. G. Sunitha
Professor & Head
Department of CSE
School of Computing
Mohan Babu University
Tirupati.

ACKNOWLEDGEMENTS

First and foremost, I extend my sincere thanks to **Dr. M. Mohan Babu**, Chancellor, for his unwavering support and vision that fosters academic excellence within the institution.

My gratitude also goes to **Mr. Manchu Vishnu, Pro-Chancellor**, for creating an environment that promotes creativity and for his encouragement and commitment to student success.

I am deeply appreciative of **Prof. Nagaraj Ramrao**, Vice Chancellor, whose leadership has created an environment conducive to learning and innovation.

I would like to thank **Dr. K. Saradhi**, Registrar, for his support in creating an environment conducive to academic success.

I am also grateful to **Dr. G. Sunitha**, Head of the Department of Computer Science and Engineering, for her valuable insights and support.

Finally, I would like to express my deepest appreciation to my project supervisor, **MOHAMED MUZAMIL,** Department of Computer Science and Engineering for continuous guidance, encouragement, and expertise throughout this project.

Thank you all for your support and encouragement.

Table of Contents

ABSTRACT

Image Steganography is a technique used to conceal information— such as text, images, or video— within a cover image, ensuring secure and unobtrusive communication. This project introduces an Android-based Image Steganography application that enables users to hide secret text messages within images using a simple and user-friendly interface.

In this system, the sender uploads an image, enters the secret message, and provides a secret key to securely encode and embed the message into the image. The recipient, with access to the same secret key, can then decode and retrieve the hidden message from the modified image.

The primary advantage of this approach lies in its simplicity and effectiveness in maintaining *confidentiality* and *authentication* without raising suspicion. Unlike traditional encryption, which may attract attention, steganography conceals the very existence of communication.

The main objective of this application is to enable secure and private communication by leveraging image-based steganography techniques. When used appropriately, this method offers a powerful solution for safeguarding sensitive information, ensuring both security and discretion in data exchange.

1.INTRODUCTION

1.1 Problem Statement

In an era dominated by digital communication, ensuring the privacy and confidentiality of sensitive information is crucial. Traditional encryption methods, while effective, often signal the presence of secret communication, which may attract attention from adversaries. Steganography offers a unique solution by hiding the existence of the message itself. When applied to images, this approach ensures that confidential information can be embedded within seemingly harmless files, thereby preventing unauthorized detection. The challenge lies in developing a system that is not only secure but also efficient and user-friendly for real-world mobile applications.

1.2 Importance of the Problem

With the rapid rise of cyber threats, surveillance, and data breaches, it has become increasingly important to secure communications in a manner that avoids drawing attention. Traditional cryptographic methods, while robust, often alert adversaries to the presence of encrypted data. Steganography, on the other hand, embeds the message within another medium— in this case, images— without visibly altering the cover medium. This project emphasizes the significance of providing secure messaging in a format that is inconspicuous, especially on widely-used Android platforms.

1.3 Objective

- Develop an Android application capable of hiding and retrieving text messages within digital images.
- Ensure the stego-image appears visually unchanged to preserve secrecy.
- Implement the Least Significant Bit (LSB) algorithm for efficient and secure data hiding.
- Provide a user-friendly interface suitable for both technical and non-technical users.
- Ensure compatibility across a wide range of Android devices.
- Handle image storage and sharing securely within the Android ecosystem.

1.4 Scope of the project

The project focuses on image-based steganography for Android platforms. Specifically, it implements the LSB method to hide and retrieve text messages in PNG and JPEG image formats. The project does not cover audio or video steganography. Furthermore, it does not incorporate internet-based data exchange to prevent external interception. All operations are conducted locally on the user's device. Target users include individuals and organizations looking for secure and private message transmission without reliance on third-party encryption services.

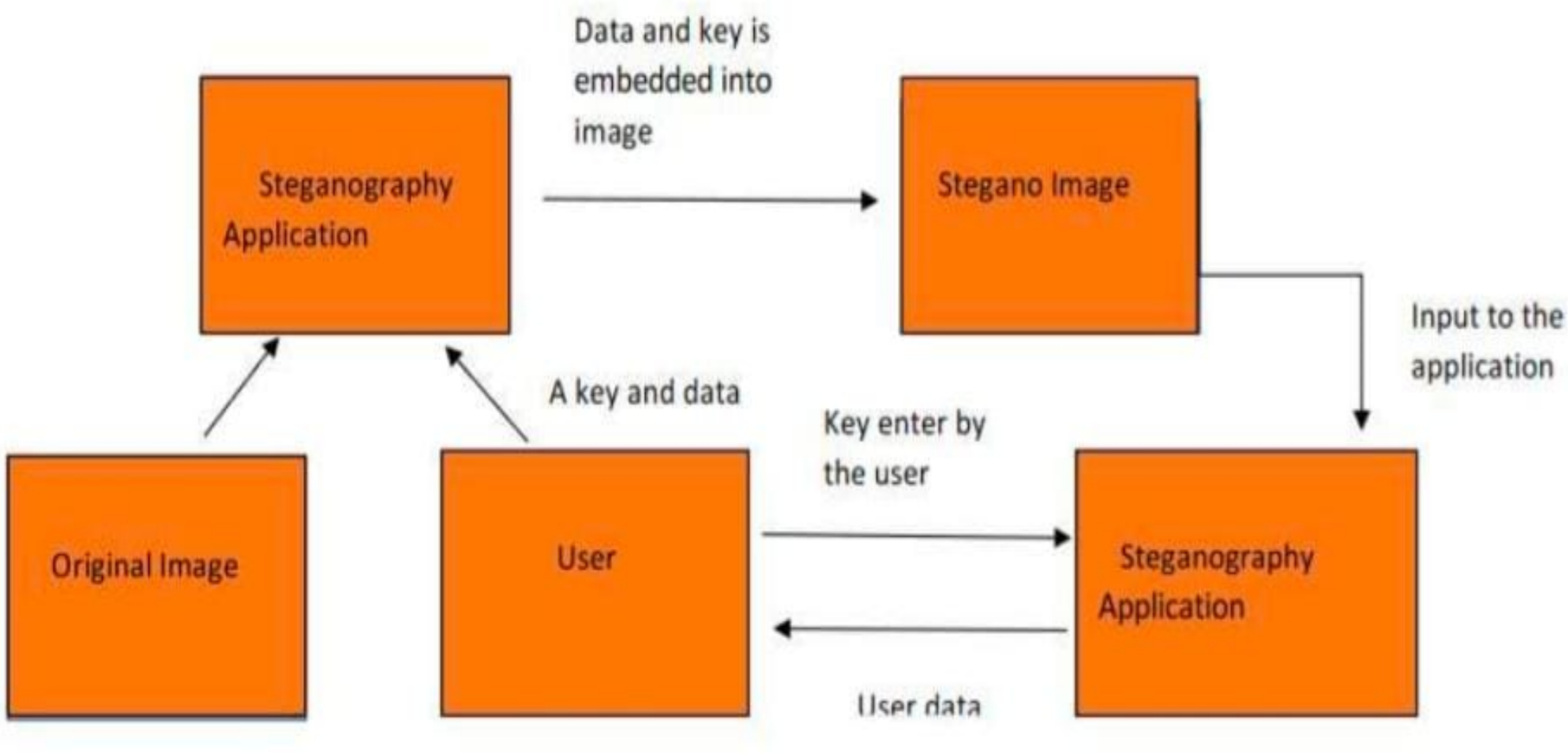
2. SYSTEM DESIGN

2.1 Architecture Diagram

The architecture of the system is modular, ensuring clear separation of concerns for each functional unit.
The major components include:

- User Interface Layer: Designed using Android's XML layout tools, this layer allows users to interact with the application, select images, and input messages.
- Steganography Core Layer: Contains the encoding and decoding logic, including the LSB algorithm used for hiding and extracting messages.
- Image Handling Layer: Manages loading, converting, and modifying bitmap images for use in steganography.
- Storage Layer: Handles saving and retrieving images from device storage, including permissions and path management.

SYSTEM ARCHITECTURE



2.2 Module Descriptions

- Image Selector Module: Facilitates choosing an image from the gallery or device storage. Ensures correct image format and size compatibility.
- Message Input Module: Provides a secure and intuitive input field for users to type or paste secret messages. Includes validation to prevent oversized messages.
- Encode Module: Implements the LSB algorithm to embed the message bit-by-bit into the image's pixel data.
- Decode Module: Extracts and reconstructs the hidden message from the stego-image using the reverse process.
- Save Module: Allows saving the encoded image to device storage.

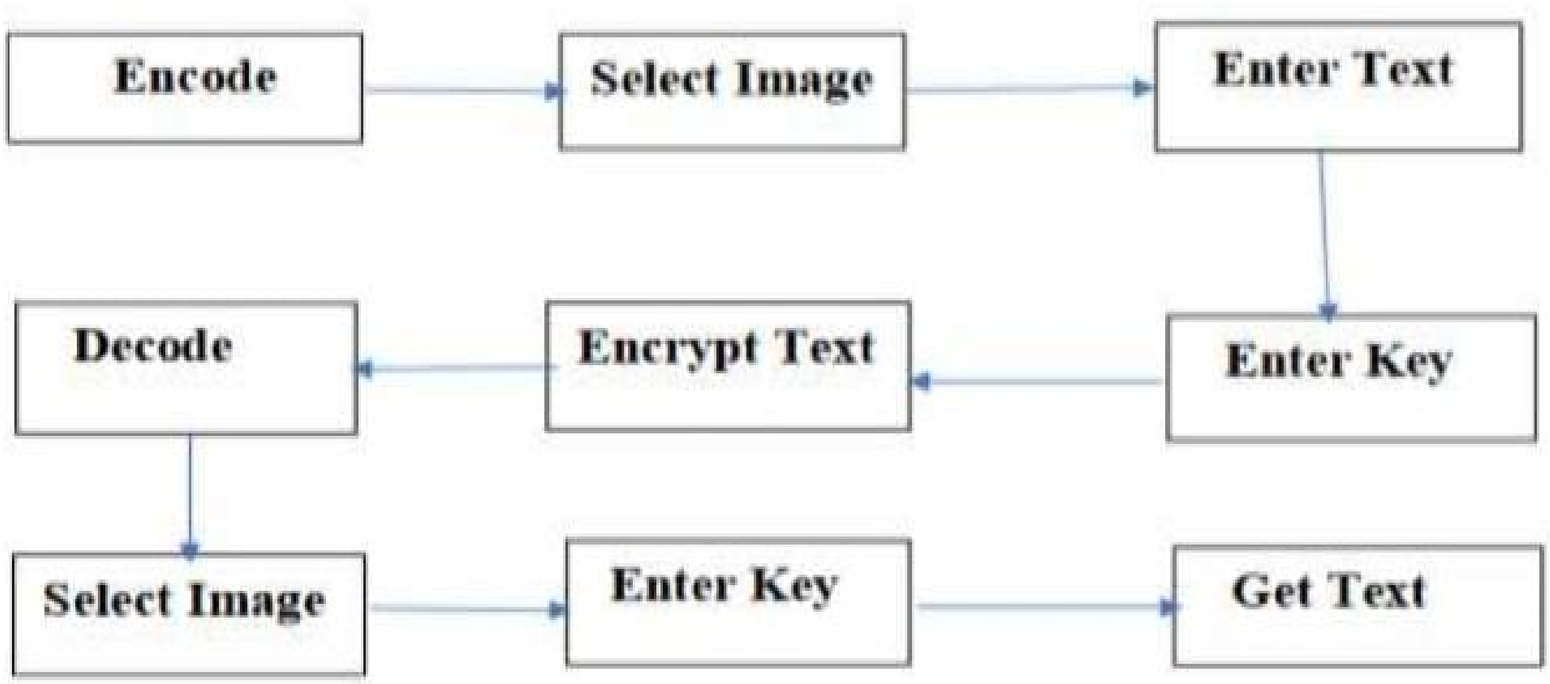


Figure: -Flow Diagram

2.3 Database Design

Since the application does not store user data on a remote server, a local storage strategy is used:

- Internal Storage: For temporary storage of stego-images.
- External Storage: For saving user-selected or processed images with appropriate file permissions.
- URI Management: Image access and manipulation are handled using Android's Content URI system to ensure security and compatibility.

3.IMPLEMENTATION

3.1 Tools and Technologies Used

- Programming Language: Java
- IDE: Android Studio
- Image Libraries: Glide for image rendering, Bitmap Factory for image manipulation
- Android SDK: API Level 29 and above
- Permissions: File access, external storage, and camera (if extended)

3.2 Front-End Development

User interfaces are developed using Android XML. The app contains multiple screens:

- Home Screen: Entry point offering options to encode or decode.
- Encoding Screen: Includes an image preview, text field for the message, and buttons for embedding.
- Decoding Screen: Allows users to choose an image and displays the extracted message after processing.
- Confirmation Dialogs: Inform the user of successful encoding/decoding.
- Error Handling: Prompt dialogs for invalid images, oversized messages, or decoding errors.

3.3 Back-End Development

- The LSB algorithm is implemented in Java, where each bit of the message is hidden in the least significant bits of image pixels.
- Pixel manipulation is done using Android' s Bitmap class.
- Message bits are distributed sequentially across the red component of pixels.
- Logic is added to track message length, ensuring complete decoding.

3.4 Integration

- UI components communicate with backend logic using event listeners and view models.
- Encoding results are previewed immediately, and saved stego-images are made available for download or sharing.
- Error handling is integrated throughout to maintain user-friendly flow.

4. TESTING, RESULT, AND DISCUSSION

4.1 Test Cases

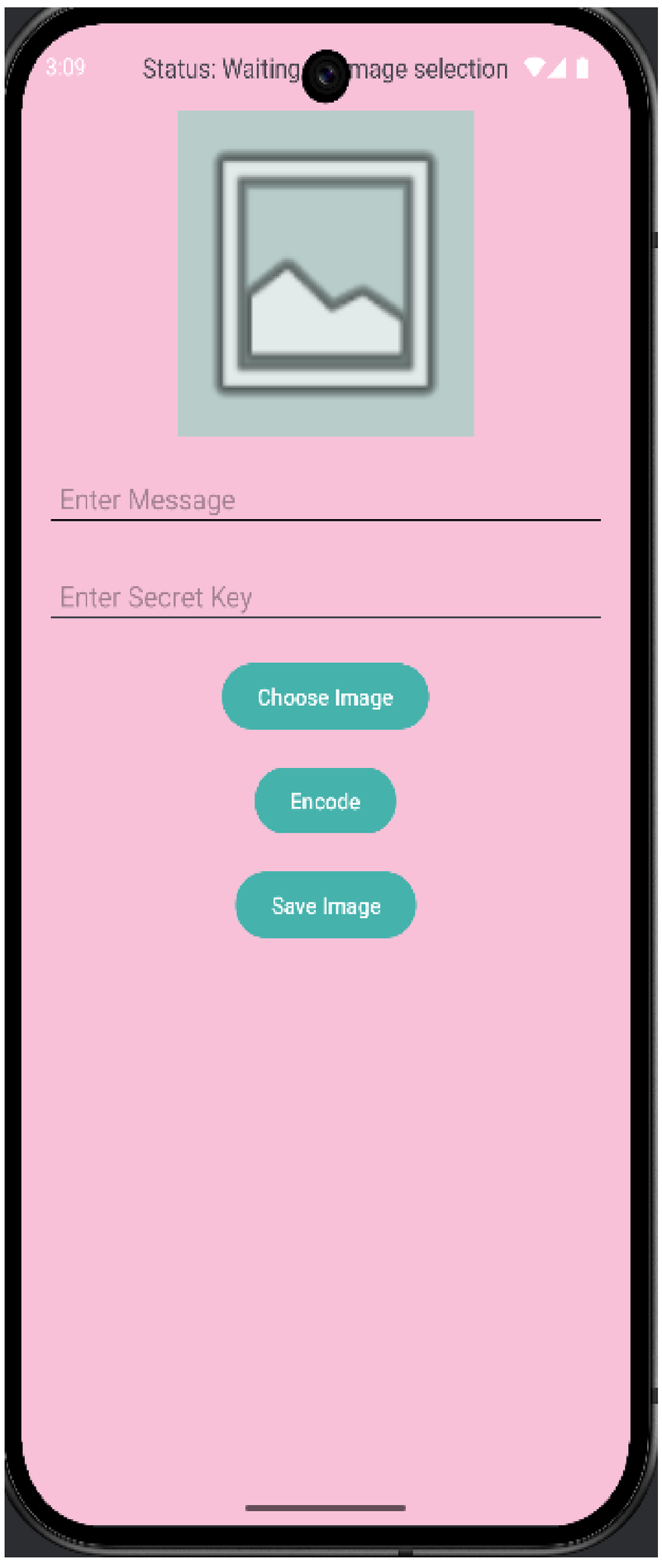
1. Encoding and decoding a short message in a high-resolution image.
2. Encoding and decoding a long message near the capacity limit of the image.
3. Attempting to encode a message that exceeds the image' s capacity.
4. Decoding an image that does not contain a hidden message.
5. Encoding and decoding with different image formats (PNG, JPEG).
6. Saving and retrieving stego-images from internal and external storage.
7. Encode & Decode Message (No Key)
   - Message: "hello"
   - Expect: Successfully decoded
8. Encode & Decode (With Key)
   - Message: "secret123", Key: "abc123"
   - Expect: Correct message with key
9. Wrong Key
   - Key used to decode is incorrect
   - Expect: Fail or incorrect output
10. No Hidden Message
    - Try decoding a clean image
    - Expect: "No message found"

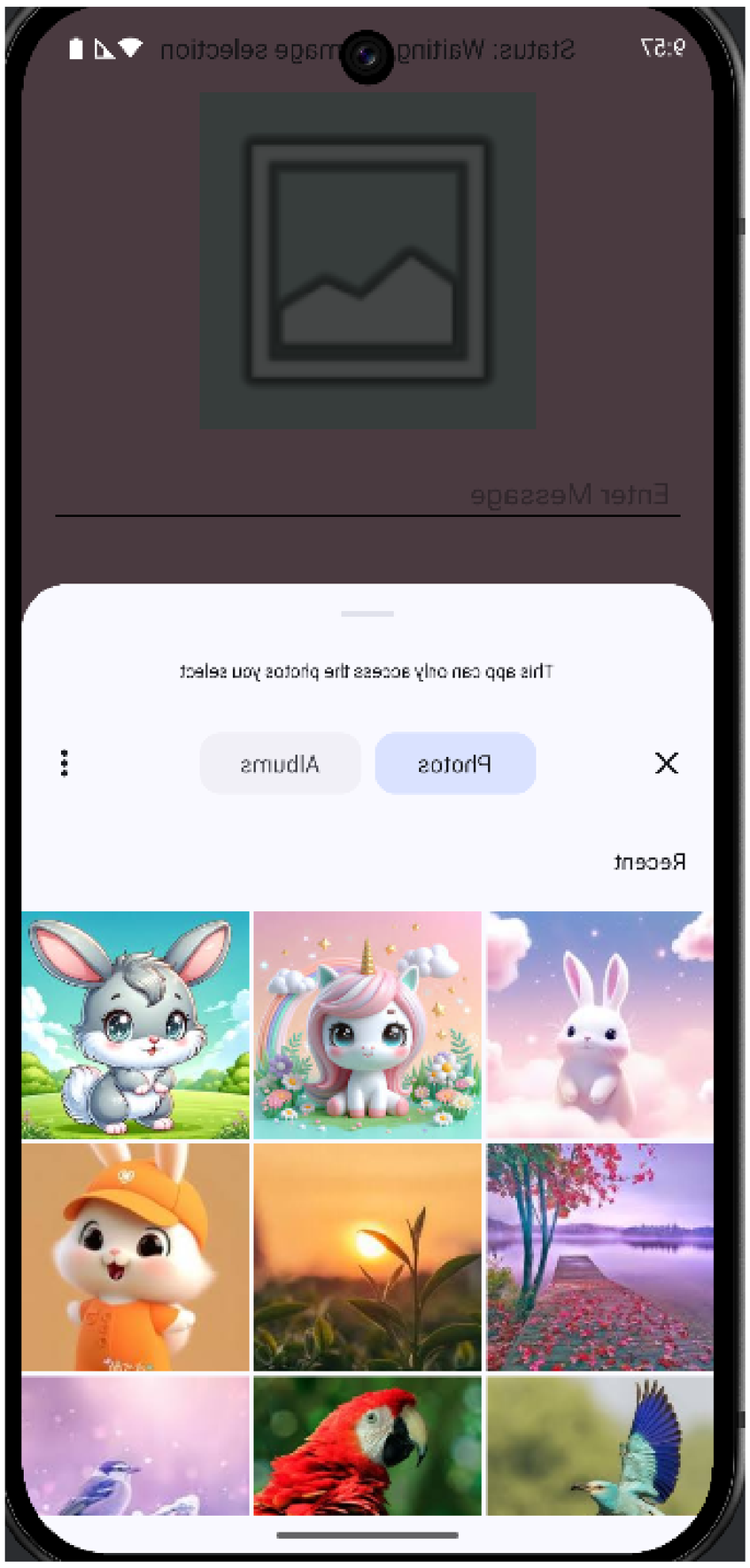
4.2 Testing Methods

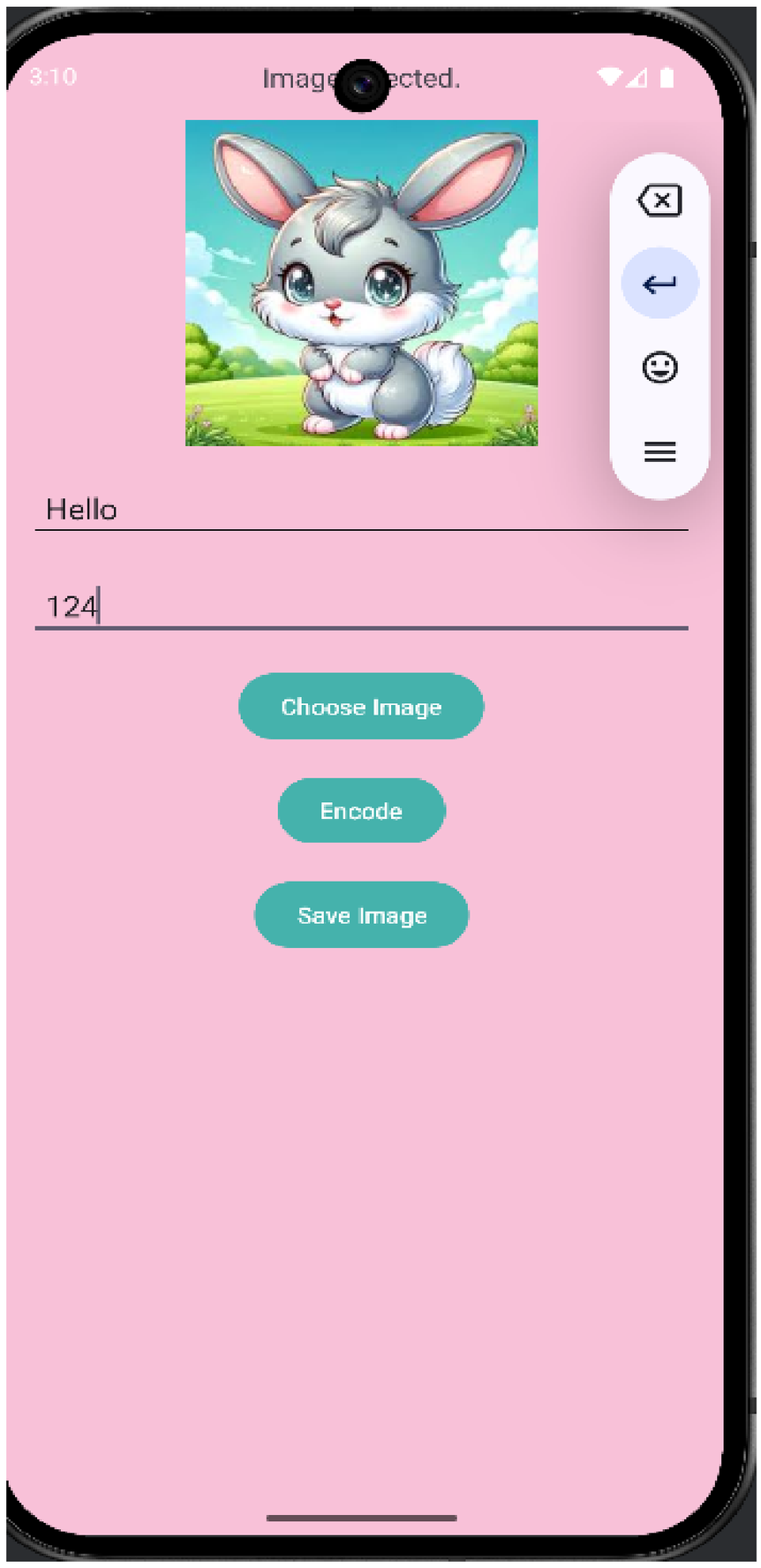
- Manual Testing: Checked user flow, UI response, and application crashes.
- Functional Testing: Verified core encoding/decoding accuracy.
- Performance Testing: Tested speed and memory usage during processing of large images.
- Error Testing: Ensured app displays appropriate messages on invalid input.

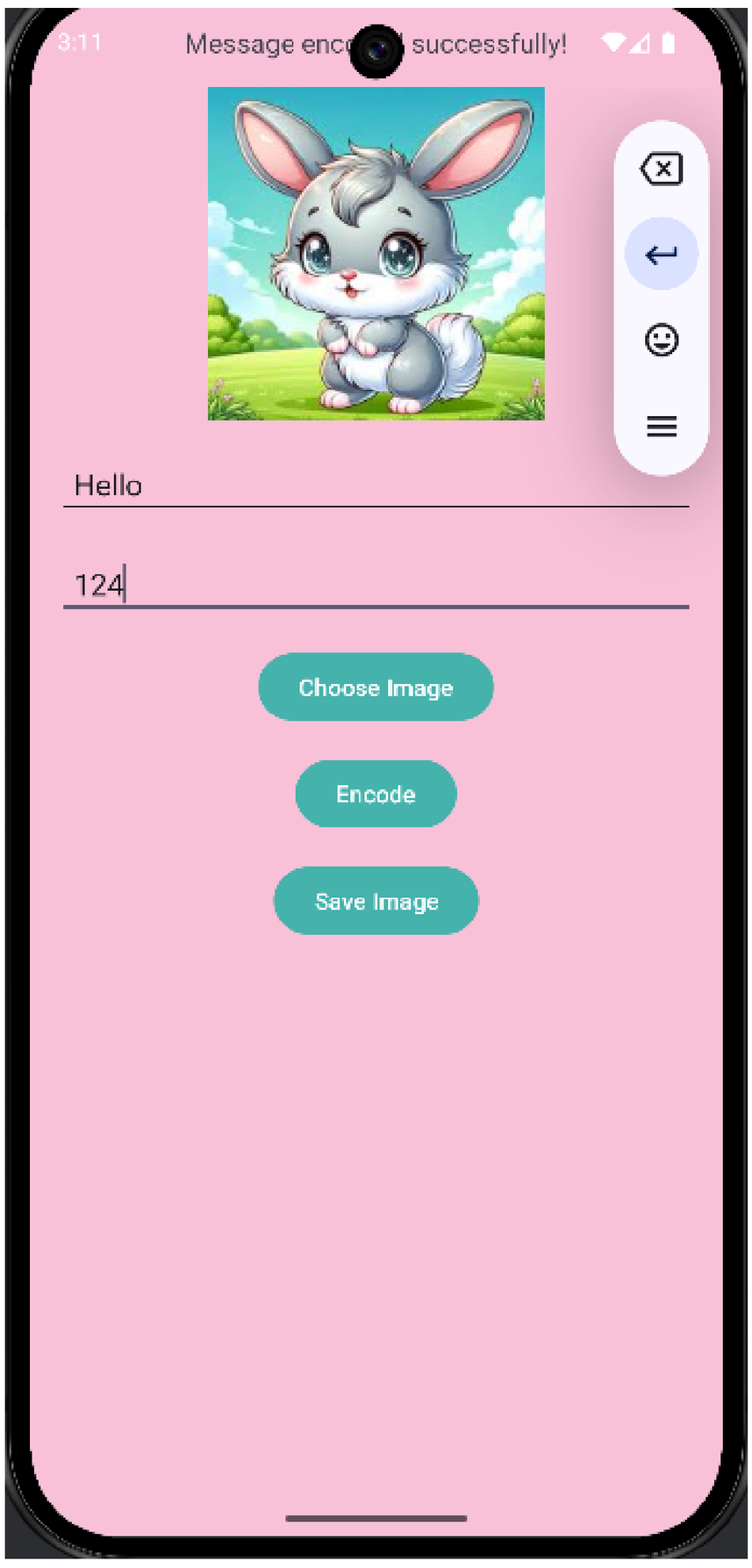
4.3 Output Screenshots

- Screenshots include:
- App launch
- Image selection interface
- Message input and confirmation
- Stego-image preview and save/share option
- Decoded message display

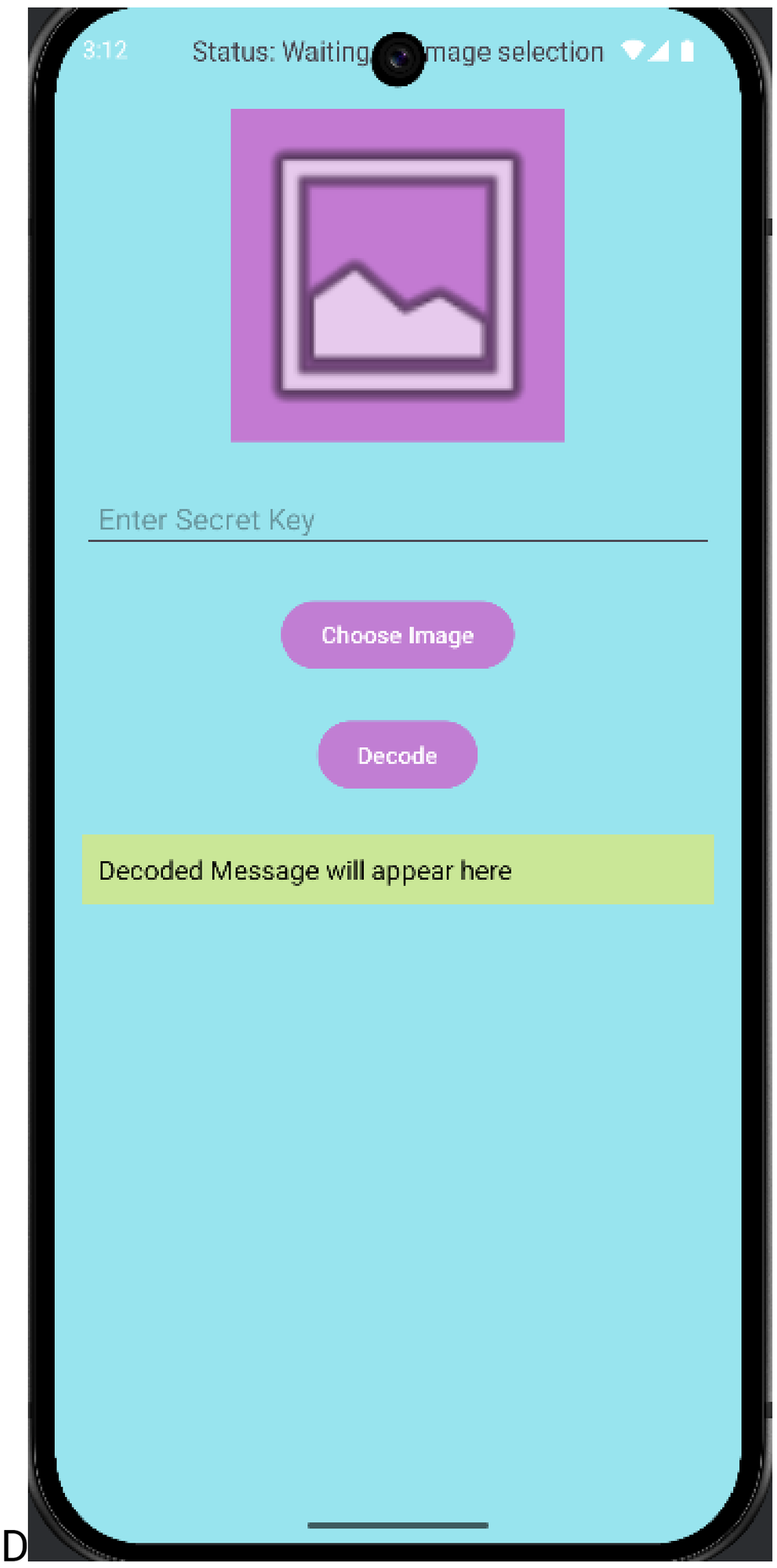


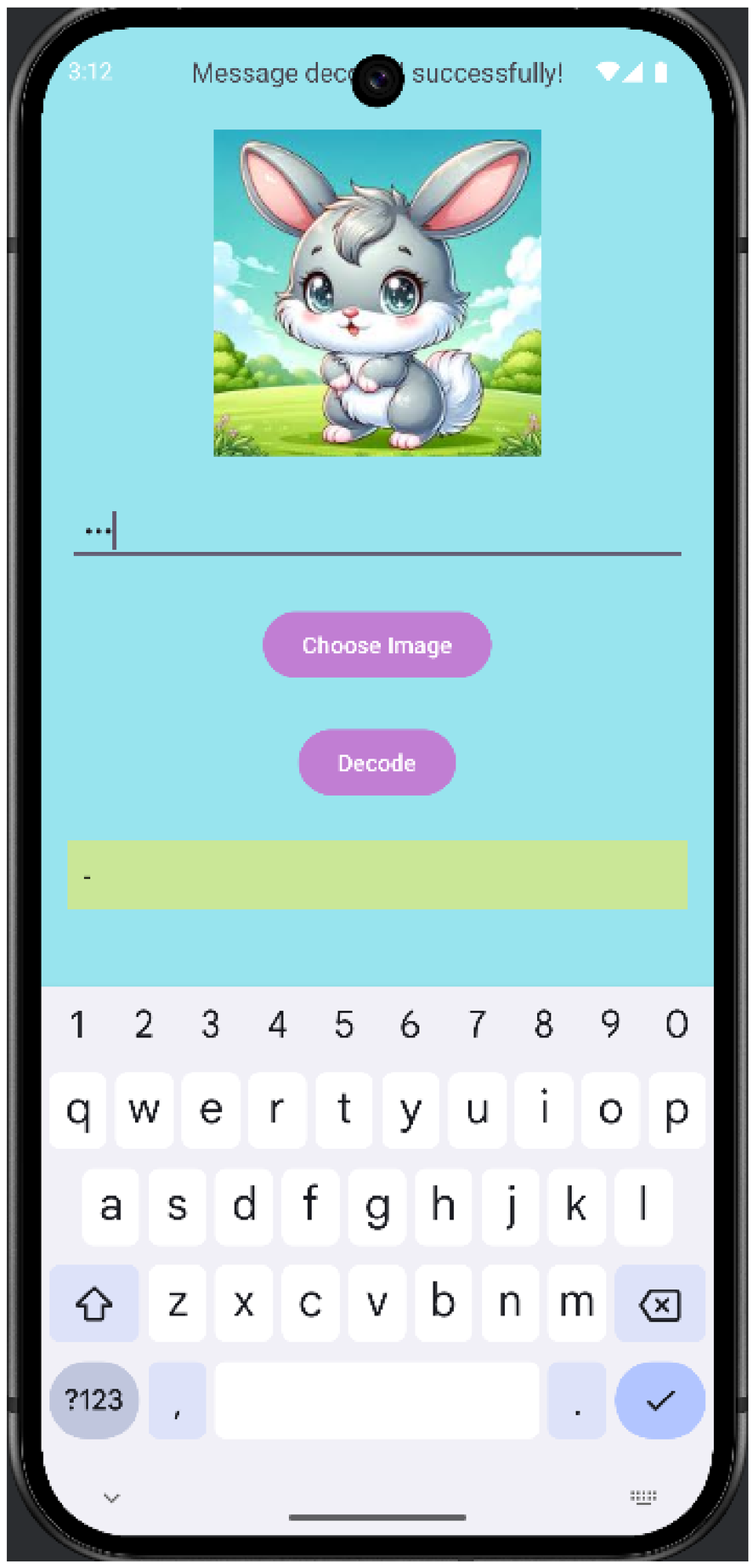

9:57
Enter Message
This app can only access the photos you select
Albums
Photos
Recent


3:10
Hello
124
Choose Image
Encode
Save Image


3:11
Message encoded successfully!
Hello
124
Choose Image
Encode
Save Image


3:08
Image Steganography
Encode
Decode





D

4.4 Analysis of Results

- The application performs efficiently on Android phones with 2GB RAM or higher.
- Stego-images retain original quality, with no visual difference detected.
- Extraction of messages is consistent and accurate.
- App handles edge cases gracefully, such as exceeding message capacity.
- No crashes or significant delays were encountered during testing.

5. CONCLUSION

5.1 Summary of Findings

The Android-based steganography app fulfills the primary objective of providing a secure and discrete method of communication. By using LSB steganography, it allows messages to be hidden within images without altering the visual integrity of the cover image. The user interface is intuitive, and the app performs reliably under various test scenarios. It offers a practical solution for everyday secure communication.

5.2 Future Enhancements

- Password Protection: Encrypt messages before embedding for double-layer security.
- Image Compression: Compress stego-images to reduce storage usage.
- Multi-format Support: Add support for BMP and GIF formats.
- Cloud Integration: Option to save/share encoded images securely via cloud.
- Video and Audio Steganography: Extend functionality to embed messages in other media types.
- Steganalysis Detection Prevention: Implement advanced techniques to avoid detection by steganalysis tools.

6. APPENDIX

6.1 Code Snippets

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/img1"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="16dp"
    app:layout_insetEdge="top">

    <!-- Button to go to Encode Activity -->

    <TextView
        android:id="@+id/title_text"
        android:layout_width="309dp"
        android:layout_height="198dp"
        android:layout_margin="0dp"
        android:layout_marginStart="20dp"
        android:layout_marginLeft="40dp"
        android:layout_marginTop="80dp"
        android:layout_marginEnd="100dp"
        android:background="#00FFFFFF"
        android:backgroundTint="#00FFFFFF"
        android:text="Image Steganography"
        android:textColor="#9C27B0"
        android:textSize="30sp"
        android:textStyle="bold" />

    <Button
        android:id="@+id/encode_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:backgroundTint="#8B3AB7"
        android:text="Encode" />

    <!-- Button to go to Decode Activity -->
```

```
    <Button
        android:id="@+id/decode_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:backgroundTint="#8B3AB7"
        android:text="Decode" />

</LinearLayout>
```

activity_encode.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:background="#E91EE6"
    android:backgroundTint="#40E91E63"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:padding="16dp">

    <!-- Status text showing the current status of encoding -->
    <TextView
        android:id="@+id/whether_encoded"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:text="Status: Waiting for image selection"
        android:textSize="16sp" />

    <!-- Image view to show the selected image -->
    <ImageView
        android:id="@+id/imageview"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_marginBottom="16dp"
        android:background="#AD1E36E9"
        android:backgroundTint="#6F1EE9A9"
        android:src="@android:drawable/ic_menu_gallery" />

    <!-- EditText for entering the message to encode -->
    <EditText
        android:id="@+id/message"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_marginBottom="16dp"
        android:backgroundTint="#000000"
        android:hint="Enter Message"
        android:padding="10dp" />

    <!-- EditText for entering the secret key for encoding -->
    <EditText
        android:id="@+id/secret_key"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:hint="Enter Secret Key"
        android:padding="10dp" />

    <!-- Button to choose the image -->
    <Button
        android:id="@+id/choose_image_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:backgroundTint="#9E27B0A5"
        android:text="Choose Image" />

    <!-- Button to trigger encoding process -->
    <Button
        android:id="@+id/encode_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:backgroundTint="#9E27B0A5"
        android:text="Encode" />

    <!-- Button to save the encoded image -->
    <Button
        android:id="@+id/save_image_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:backgroundTint="#9E27B0A5"
        android:text="Save Image" />

</LinearLayout>
```

main_decode.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:background="#6600C9D4"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:padding="16dp">

    <!-- Status text showing the current decoding status -->
    <TextView
        android:id="@+id/status_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        android:gravity="center"
        android:text="Status: Waiting for image selection"
        android:textSize="16sp" />

    <!-- Image view to show the selected image -->
    <ImageView
        android:id="@+id/image_view"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_marginBottom="24dp"
        android:background="@color/white"
        android:backgroundTint="#89E91EBA"
        android:scaleType="centerCrop"
        android:src="@android:drawable/ic_menu_gallery" />

    <!-- EditText for entering the secret key for decoding -->
    <EditText
        android:id="@+id/secret_key_field"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        android:hint="Enter Secret Key"
        android:inputType="textPassword"
        android:padding="10dp" />

    <!-- Button to choose the image to decode -->
    <Button
        android:id="@+id/choose_image_button"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:backgroundTint="#4DE91EBA"
    android:text="Choose Image" />

<!-- Button to trigger the decoding process -->
<Button
    android:id="@+id/decode_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:backgroundTint="#4DE91EBA"
    android:text="Decode" />

<!-- TextView to display the decoded message -->
<TextView
    android:id="@+id/message_field"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:background="#EEEEEE"
    android:backgroundTint="#7BFFEB3B"
    android:padding="10dp"
    android:text="Decoded Message will appear here"
    android:textColor="#000000"
    android:textSize="16sp" />
</LinearLayout>
```

MainActivity.java:

```
package com.example.project;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Ensure you are using the correct IDs from XML
        Button encodeButton = findViewById(R.id.encode_button);
        Button decodeButton = findViewById(R.id.decode_button);
```

```
        // Set click listener for encode button
        encodeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Start the Encode activity when the button is clicked
                Intent intent = new Intent(MainActivity.this, Encode.class);
                startActivity(intent);
            }
        });

        // Set click listener for decode button
        decodeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Start the Decode activity when the button is clicked
                Intent intent = new Intent(MainActivity.this, Decode.class);
                startActivity(intent);
            }
        });
    }
}
```

Encode.java :

```
package com.example.project;
import android.Manifest;
import android.app.ProgressDialog;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.ArrayList;
```

```
import java.util.List;
public class Encode extends AppCompatActivity {
    private static final int SELECT_PICTURE = 100;
    private static final String TAG = "EncodeActivity";
    private TextView statusText;
    private ImageView;
    private EditText messageInput, secretKeyInput;
    private ProgressDialog saveDialog;
    private Uri selectedImageUri;
    private Bitmap selectedBitmap;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_encode);
        statusText = findViewById(R.id.whether_encoded);
        imageView = findViewById(R.id.imageview);
        messageInput = findViewById(R.id.message);
        secretKeyInput = findViewById(R.id.secret_key);
        Button chooseImageBtn = findViewById(R.id.choose_image_button);
        Button encodeBtn = findViewById(R.id.encode_button);
        Button saveImageBtn = findViewById(R.id.save_image_button);

        checkAndRequestPermissions();
    chooseImageBtn.setOnClickListener(view -> chooseImage());
    encodeBtn.setOnClickListener(view -> {
        if (selectedBitmap == null) {
            statusText.setText("Please select an image.");
            return;
        }
        String message = messageInput.getText().toString();
        String key = secretKeyInput.getText().toString();
        if (message.isEmpty() || key.isEmpty()) {
            statusText.setText("Enter both message and secret key.");
            return;
        }
        Bitmap encodedBitmap = encodeMessage(selectedBitmap, message, key);
        if (encodedBitmap != null) {
            selectedBitmap = encodedBitmap;
            imageView.setImageBitmap(encodedBitmap);
            statusText.setText("Message encoded successfully!");
        } else {
            statusText.setText("Encoding failed. Message too large?");
        }
    });
    saveImageBtn.setOnClickListener(view -> {
        if (selectedBitmap == null) {
```

```
        statusText.setText("No image selected.");
        return;
    }
    saveDialog = new ProgressDialog(this);
    saveDialog.setMessage("Saving, Please Wait...");
    saveDialog.setTitle("Saving Image");
    saveDialog.setIndeterminate(false);
    saveDialog.setCancelable(false);
    saveDialog.show();
            new Thread(() -> saveImageToStorage(selectedBitmap)).start();
        });
    }
    private void chooseImage() {
        Intent = new Intent(Intent.ACTION_GET_CONTENT);
        intent.setType("image/*");
        startActivityForResult(Intent.createChooser(intent, "Select Picture"), SELECT_PICTURE);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == SELECT_PICTURE && resultCode == RESULT_OK && data != null &&
data.getData() != null) {
            selectedImageUri = data.getData();
            try {
                selectedBitmap = MediaStore.Images.Media.getBitmap(getContentResolver(),
selectedImageUri);
                imageView.setImageBitmap(selectedBitmap);
                statusText.setText("Image selected.");
            } catch (IOException e) {
                Log.e(TAG, "Error loading image", e);
                statusText.setText("Error loading image.");
            }
        }
    }
    private Bitmap encodeMessage(Bitmap original, String message, String key) {
        Bitmap image = original.copy(Bitmap.Config.ARGB_8888, true);
String encrypted = xorWithKey(message, key);
encrypted += "\0"; // Null termination
byte[] messageBytes = encrypted.getBytes();
StringBuilder binary = new StringBuilder();
for (byte b : messageBytes) {
    binary.append(String.format("%8s", Integer.toBinaryString(b & 0xFF)).replace(' ', '0'));
}
int width = image.getWidth();
int height = image.getHeight();
```

```
int bitIndex = 0;
int totalBits = binary.length();
if (totalBits > width * height) {
    return null; // Not enough pixels to store message
}
outerLoop:
for (int y = 0; y < height; y++) {
    for (int x = 0; x < width; x++) {
        if (bitIndex >= totalBits) break outerLoop;
        int pixel = image.getPixel(x, y);
        int alpha = (pixel >> 24) & 0xFF;
        int red = (pixel >> 16) & 0xFF;
        int green = (pixel >> 8) & 0xFF;
        int blue = pixel & 0xFF;
        int bit = binary.charAt(bitIndex) - '0';
        blue = (blue & 0xFE) | bit;
        int newPixel = (alpha << 24) | (red << 16) | (green << 8) | blue;
        image.setPixel(x, y, newPixel);

        bitIndex++;
    }
}
return image;
}
private String xorWithKey(String data, String key) {
    StringBuilder result = new StringBuilder();
    for (int i = 0; i < data.length(); i++) {
        result.append((char) (data.charAt(i) ^ key.charAt(i % key.length())));
    }
    return result.toString();
}
private void saveImageToStorage(Bitmap bitmap) {
    File = new File(getExternalFilesDir(null), "Image_" + System.currentTimeMillis() + ".png");
    try (OutputStream out = new FileOutputStream(file)) {
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, out);
        runOnUiThread(() -> {
            saveDialog.dismiss();
            statusText.setText("Saved to:\n" + file.getAbsolutePath());
        });
    } catch (IOException e) {
        Log.e(TAG, "Saving failed", e);
        runOnUiThread(() -> {
            saveDialog.dismiss();
            statusText.setText("Saving failed.");
        });
}
```

```
    }
    private void checkAndRequestPermissions(){
        List<String> neededPermissions = new ArrayList<>();
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED){
            neededPermissions.add(Manifest.permission.READ_EXTERNAL_STORAGE);
        }
        if (!neededPermissions.isEmpty()){
            ActivityCompat.requestPermissions(this,
                    neededPermissions.toArray(new String[0]),1);
        }
    }
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           @NonNull String[] permissions,
                                           @NonNull int[] grantResults){
        super.onRequestPermissionsResult(requestCode,permissions,grantResults);
    }
}
```

Decode.java:

```
package com.example.project;

import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.activity.result.ActivityResult;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;

import java.io.IOException;

public class Decode extends AppCompatActivity {
```

```
private static final String TAG = "DecodeActivity";

private TextView statusText;
private ImageView;
private TextView messageField;
private EditText secretKeyField;
private Bitmap selectedImage;
private Uri filepath;

    private final ActivityResultLauncher<Intent> imagePickerLauncher =
            registerForActivityResult(new ActivityResultContracts.StartActivityForResult(),
                    new ActivityResultCallback<ActivityResult>() {
                        @Override
                        public void onActivityResult(ActivityResult result) {
                            if (result.getResultCode() == RESULT_OK && result.getData() != null &&
result.getData().getData() != null) {
                                filepath = result.getData().getData();
                                try {
                                    selectedImage =
MediaStore.Images.Media.getBitmap(getContentResolver(), filepath);
                                    imageView.setImageBitmap(selectedImage);
                                    statusText.setText("Image loaded successfully.");
                                } catch (IOException e) {
                                    Log.e(TAG, "Image loading failed", e);
                                    statusText.setText("Failed to load image.");
                                }
                            } else {
                                statusText.setText("No image selected.");
                            }
                        }
                    });

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

    statusText = findViewById(R.id.status_text);
    imageView = findViewById(R.id.image_view);
    messageField = findViewById(R.id.message_field);
    secretKeyField = findViewById(R.id.secret_key_field);

    Button chooseImageButton = findViewById(R.id.choose_image_button);
    Button decodeButton = findViewById(R.id.decode_button);
```

```
chooseImageButton.setOnClickListener(v -> openImageChooser());

decodeButton.setOnClickListener(v -> {
    if (selectedImage != null) {
        String key = secretKeyField.getText().toString();
        if (key.isEmpty()) {
            statusText.setText("Please enter the secret key.");
            return;
        }

        String decoded = decodeMessage(selectedImage, key);
        if (decoded != null && !decoded.isEmpty()) {
            messageField.setText(decoded);
            statusText.setText("Message decoded successfully!");
        } else {
            statusText.setText("No hidden message found.");
            messageField.setText("");
        }
    } else {
        statusText.setText("Please select an image first.");
    }
});
}

private void openImageChooser() {
    Intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("image/*");
    imagePickerLauncher.launch(Intent.createChooser(intent, "Select an Image"));
}

private String decodeMessage(Bitmap image, String key) {
    StringBuilder binary = new StringBuilder();
    int width = image.getWidth();
    int height = image.getHeight();

    outerLoop:
    for (int y = 0; y < height; y++) {
for (int x = 0; x < width; x++) {
    int pixel = image.getPixel(x, y);
    int blue = pixel & 0xFF;
    int lsb = blue & 1;
    binary.append(lsb);

    // Check for null terminator (00000000)
    if (binary.length() % 8 == 0) {
        String byteStr = binary.substring(binary.length() - 8);
```

```
                if (Integer.parseInt(byteStr, 2) == 0) {
                    break outerLoop;
                }
            }
        }
    }
```

```
// Convert binary to encrypted string
        StringBuilder encryptedMessage = new StringBuilder();
        for (int i = 0; i + 8 <= binary.length(); i += 8) {
            String byteStr = binary.substring(i, i + 8);
            int ascii = Integer.parseInt(byteStr, 2);
            if (ascii == 0) break;
            encryptedMessage.append((char) ascii);
        }

        return xorWithKey(encryptedMessage.toString(), key); // Final decoded message
    }
```

```
    private String xorWithKey(String data, String key) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < data.length(); i++) {
            result.append((char) (data.charAt(i) ^ key.charAt(i % key.length())));
        }
        return result.toString();
    }
}
```