A PROJECT REPORT

On

Morse Aid App

(SOS App)

Submitted by

Mr. Abheek Shailesh Shah

In partial fulfilment for the award of the degree

Of

BACHELOR OF SCIENCE

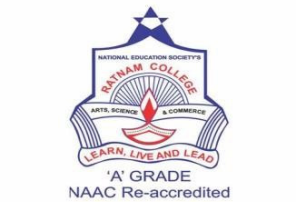
In

COMPUTER SCIENCE

Under the Guidance of

Mrs. Vinita Gupta

Department of Computer Science



NES Ratnam College of Arts, Science & Commerce

(Sem V)

(2024-2025)

'A' GRADE
NAAC Re-accredited

NATIONAL EDUCATION SOCIETY'S

Ratnam College of Arts, Science & Commerce
Bhandup, Mumbai – 400 078.

Department of Computer Science

CERTIFICATE

Class: ***Roll / Seat No.:*** ***Year:***

This is to certify that Mr./Ms. Abheek Shailesh Shah has satisfactorily completed the project Morse Aid App to be submitted in the partial fulfilment of the 3 years Degree Course Bachelor in Computer Science for the University of Mumbai for the year 2024 to2025.

*Place:*_____

*Date:*_____

_____

Project Guide

_____

In-charge,
Dept. of Computer Science

_____

Signature of Examiner with Date

DECLARATION

I, Abheek Shailesh Shah, hereby declare that the project entitled" Morse Aid App" submitted in the partial fulfilment for the award of **Bachelor of Science** in **Computer Science** during the academic year **2024 – 2025** is my original work and the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Signature of the Student:

Place:

Date:

Acknowledgement

To list who all have helped me is difficult because they are so numerous and the depth is so enormous.

I would like to acknowledge the following as being idealistic channels and fresh dimensions in the completion of this project.

I take this opportunity to thank the **University of Mumbai** for giving me a chance to do this project.

I would like to thank my **Principal, Dr. Vinita Dhulia** for providing the necessary facilities required for completion of this project.

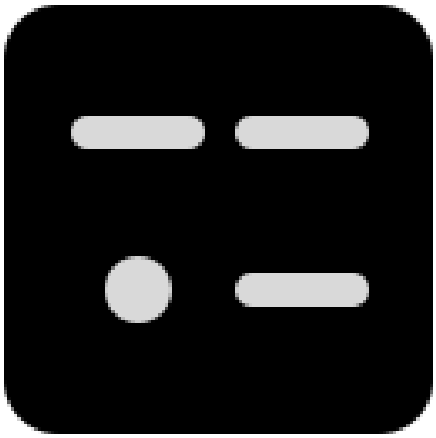
I would also like to express my sincere gratitude towards my **Project guide Mrs. Vinita Gupta** whose guidance and care made the project successful.

I would like to thank my college, for having provided various reference books and magazines related to my project.

Lastly, I would like to thank each and every person who directly or indirectly helped me in the completion of the project, especially M**y Parents and Peers** who supported me throughout my project.

Date:

Mr. Abheek Shailesh Shah



Morse Aid App

SOS Application

Abheek Shah | TYCS

INDEX

SR NO	COMPONENTS	PAGE NO	SIGN
1	**Preliminary Investigation** • Project Introduction • Project Overview • Project Objectives • Project Scope • Project Features • Project Resources • System Requirements	**8 - 14**	
2	**System Analysis & Design** • Flow Chart • Er Diagram • Data Flow Diagram • Use Case Diagram • Class Diagram • Sequence Diagram • Component Diagram • Gannt Chart	**15 - 23**	
3	**App Structure & Programming** • Source Code • Output	**23 - 67**	
4	**References**	**68**	

Preliminary Investigation

Project Introduction

Morse Aid: Revolutionizing Emergency Communication through Morse Code

In a world where quick and effective communication can mean the difference between life and death, Morse Aid stands as an innovative solution designed to harness the power of Morse code for emergency situations. The increasing need for a reliable communication tool has never been more critical, especially as we continue to face challenges in public safety and rapid response. With its unique features, Morse Aid transforms traditional emergency communication into a seamless and efficient process.

The development of Morse Aid was driven by the realization that during emergencies, every second counts. Many people are unaware of the Morse code system, which can effectively convey urgent messages. By leveraging this age-old method of communication, Morse Aid aims to educate and empower users, allowing them to signal for help or relay essential information quickly. The app is designed for both novices and experienced users, featuring a user-friendly interface that simplifies the process of sending Morse code messages.

Morse Aid conveys distress through sound and light, ensuring users can effectively signal their need for help. The app includes features for emergency calling, allowing users to reach out to the appropriate authorities without delay. Users can tap buttons representing various emergency situations, each corresponding to a specific helpline. This design makes it easy to access help without fumbling through contacts during high-stress moments, ensuring effective communication even in chaotic scenarios.

Additionally, Morse Aid simulates an ambulance siren with synchronized lights and sound effects, mimicking real-life emergency scenarios. This feature serves as a powerful reminder of the urgency of the situation while keeping the interface engaging and informative. Users can easily control the siren and light settings, ensuring they have command over their emergency signaling.

Morse Aid is built using Android Studio, Google’s official integrated development environment (IDE) for Android app development, which allows for robust coding, debugging, and testing processes. The design aspects of the app were crafted using Figma, a cloud-based design tool that enhances the overall user interface and experience.

By prioritizing user experience and integrating essential features, Morse Aid emerges as a comprehensive tool for emergency communication. The app's source code is hosted on GitHub, fostering an open-source environment for collaboration and improvement.

In conclusion, Morse Aid is not just an app; it is a lifeline for those seeking immediate help in emergencies. With a focus on accessibility and efficiency, Morse Aid stands ready to redefine communication during critical moments, empowering users to take action when it matters most. Available for download on F-Droid, Morse Aid prioritizes user privacy and offers a free, open-source solution for Android devices.

Project Overview

Morse Aid is a mobile application designed to enhance emergency communication using Morse code. Built with Android Studio and Java, it features intuitive controls that allow users to signal distress through sound, light, and emergency calling. Each button represents a specific emergency situation, ensuring quick access to help. The app also simulates an ambulance siren with synchronized lights and sounds to convey urgency. Prioritizing user privacy, Morse Aid is a free, open-source solution available on F-Droid, with its source code hosted on GitHub, fostering collaboration and continuous improvement.

Key Features:

1. Morse to Text / Text to Morse

The app enables users to convert Morse code into text and vice versa, facilitating easy communication and learning for both novices and experienced users.

2. Morse to Sound

Users can send distress signals through sound, allowing the app to convey messages audibly, enhancing awareness in emergency situations.

3. Morse to Flashing Light

The app can utilize light signals to communicate Morse code visually, providing an alternative method for signaling help when sound is not feasible.

4. Morse to Screen Flash

The screen can flash to represent Morse code messages, ensuring users can effectively convey distress signals in a visually striking manner.

5. Emergency Color

The app offers color-coded signals for different emergencies, allowing users to quickly identify and respond to various situations.

6. Emergency Siren

Users can activate an ambulance siren effect, mimicking real-life emergency sounds to draw attention during critical moments.

7. Emergency Numbers

The app includes quick access to essential emergency contact numbers, ensuring users can reach the appropriate authorities without delay.

Project Objectives

- **Enable Morse Code Communication:**
  - Provide tools for converting Morse code to text and vice versa for effective communication in emergency or non-verbal situations.

- **Facilitate Emergency Signaling:**
  - Offer emergency color-coded buttons that initiate calls to relevant helplines or display visual signals to attract attention.

- **Incorporate Sound and Visual Alerts:**
  - Implement a siren feature that mimic ambulance lights (alternating blue/red) along with synchronized sound to indicate distress or emergency.

- **Support Multiple Output Modes:**
  - Allow Morse code to be conveyed via sound, flashing light, or screen flashes for accessibility in various scenarios.

- **Create an Intuitive User Experience:**
  - Ensure a simple, user-friendly interface with quick access to emergency features, allowing users to operate under stress easily.

- **Promote Multilingual Access:**
  - Support multiple languages to make the app accessible to users across different regions.

- **Enhance Personal Safety:**
  - Provide customizable settings for distress signals, ensuring users can alert others effectively in various emergency situations.

- **Offer Offline Functionality:**
  - Ensure key features like Morse code translation and visual/sound alerts work without requiring an internet connection.

These objectives will guide the app's development to ensure it serves as a reliable tool for emergency situations and Morse code communication.

Project Scope

The Morse Aid app aims to develop, deploy, and test a comprehensive tool for effective communication through Morse code. Designed for a wide range of users, including individuals, emergency responders, and educational institutions, the app emphasizes accessibility, reliability, and performance.
The project encompasses the following key aspects:

1. Core Features Development

The scope includes implementing essential features to enhance functionality and usability:

- Morse Code Translation: Convert text to Morse code and vice versa for easy communication.
- Emergency Communication: Quick access buttons for emergency helplines enables direct dialling.
- Visual and Auditory Signals: Display Morse code as flashing lights and sound signals for multiple communication methods.
- Color-Coded Alerts: Users can select emergency situations through color-coded buttons that trigger corresponding signals.
- Custom Siren Effect: Syncs ambulance siren effects with visual signals to simulate emergency alerts.

2. Technology Stack

The project will utilize modern technologies to ensure a robust application:

- Frontend: Developed with Android Studio and Java for a native mobile experience.
- SVG Graphics: Utilizes SVG for scalable and high-quality graphics.
- Sound and Visual Libraries: Integrates sound libraries for audio effects and dynamic UI components.

3. Security and Data Privacy

The app prioritizes user data security, focusing on protecting personal information and complying with data privacy regulations.

4. Testing and Debugging

Thorough testing of all functionalities, including unit and user acceptance testing, will ensure reliability.

5. Deployment

The app will be deployed on the Google Play Store for public access, designed for easy updates and scalability.

6. Future Extensions

Future enhancements may include user profiles, a history of Morse code translations, tutorials, and wearable device integrations.

Project Features

The Morse Aid app is designed with a comprehensive set of features that enable effective communication through Morse code. It caters to a variety of users, including individuals, emergency responders, and educational institutions, ensuring accessibility and reliability. The key features of the app include:

1. Morse Code Translation

The app allows users to easily convert text to Morse code and vice versa. This feature facilitates straightforward communication, enabling users to send and receive messages in Morse code format.

2. Emergency Communication

Quick access buttons for emergency helplines enable users to dial directly from the app. This feature ensures that users can swiftly connect with emergency services when needed.

3. Visual and Auditory Signals

Morse code can be displayed as flashing lights and sound signals, providing multiple methods of communication. This functionality is particularly useful for users with hearing impairments or in situations where silence is necessary.

4. Color-Coded Alerts

Users can select emergency situations through color-coded buttons that trigger corresponding signals. This feature enhances usability during high-stress scenarios, allowing for quick identification of the alert type.

5. Custom Siren Effect

The app includes a customizable ambulance siren effect that syncs with visual signals to simulate emergency alerts. This feature helps to attract attention and signal distress effectively.

6. User-Friendly Interface

Built with a clean and intuitive design, the user interface ensures a seamless experience across different devices. The app is designed to be accessible to users of all technical levels, making navigation easy.

7. Security and Data Privacy

The app prioritizes user data security, focusing on protecting personal information and ensuring compliance with data privacy regulations.

8. Future Extensions

The scope for future enhancements includes user profiles, a history of Morse code translations, tutorials, and integrations with wearable devices to improve functionality and user experience.

Project Resources

The development of the Morse Aid app utilizes a combination of modern tools, technologies, and services to deliver an efficient, secure, and user-friendly communication solution. The following resources are integral to the project:

1. **Real-Time Communication: Stream SDK**

For managing real-time communication and audio-visual signals, the app leverages the Stream SDK. This platform provides the necessary infrastructure for sound and visual signaling, including audio effects and flashing light features. By utilizing Stream's capabilities, the app avoids the complexity of developing a backend from scratch, allowing a focus on optimizing user experience. Additionally, Stream's cloud services ensure scalability, accommodating a high volume of users and emergency alerts without performance degradation.

2. **Development Environment: Android Studio**

Android Studio serves as the primary Integrated Development Environment (IDE) for developing the Morse Aid app. Its comprehensive suite of tools for coding, debugging, and testing Android applications ensures a streamlined development process. The built-in emulator and support for Java make it an ideal choice for this project, allowing for efficient testing and iteration.

3. **Tech Stack: Java and SVG**

The app is built using a modern tech stack that guarantees performance and maintainability:

- **Java:** The core programming language for developing the app, offering robustness and reliability essential for mobile applications.
- **SVG Graphics:** Scalable Vector Graphics (SVG) are used for the app's visuals, ensuring high-quality graphics that adapt seamlessly to various screen sizes.

4. **Audio and Visual Libraries**

The project integrates various audio and visual libraries to enhance functionality. These libraries provide the necessary tools for implementing sound effects and dynamic UI changes, ensuring a rich user experience.

5. **Testing and Debugging Tools**

Thorough testing and debugging are critical to ensuring app reliability. Tools such as JUnit for unit testing and Android Debug Bridge (ADB) for debugging are utilized to verify functionality and improve the overall quality of the application.

6. **Hosting: Google Play Store**

The Morse Aid app is deployed on the Google Play Store for public access. The app is designed for easy updates and scalability, allowing for the integration of new features based on user feedback and technological advancements.

System Requirements

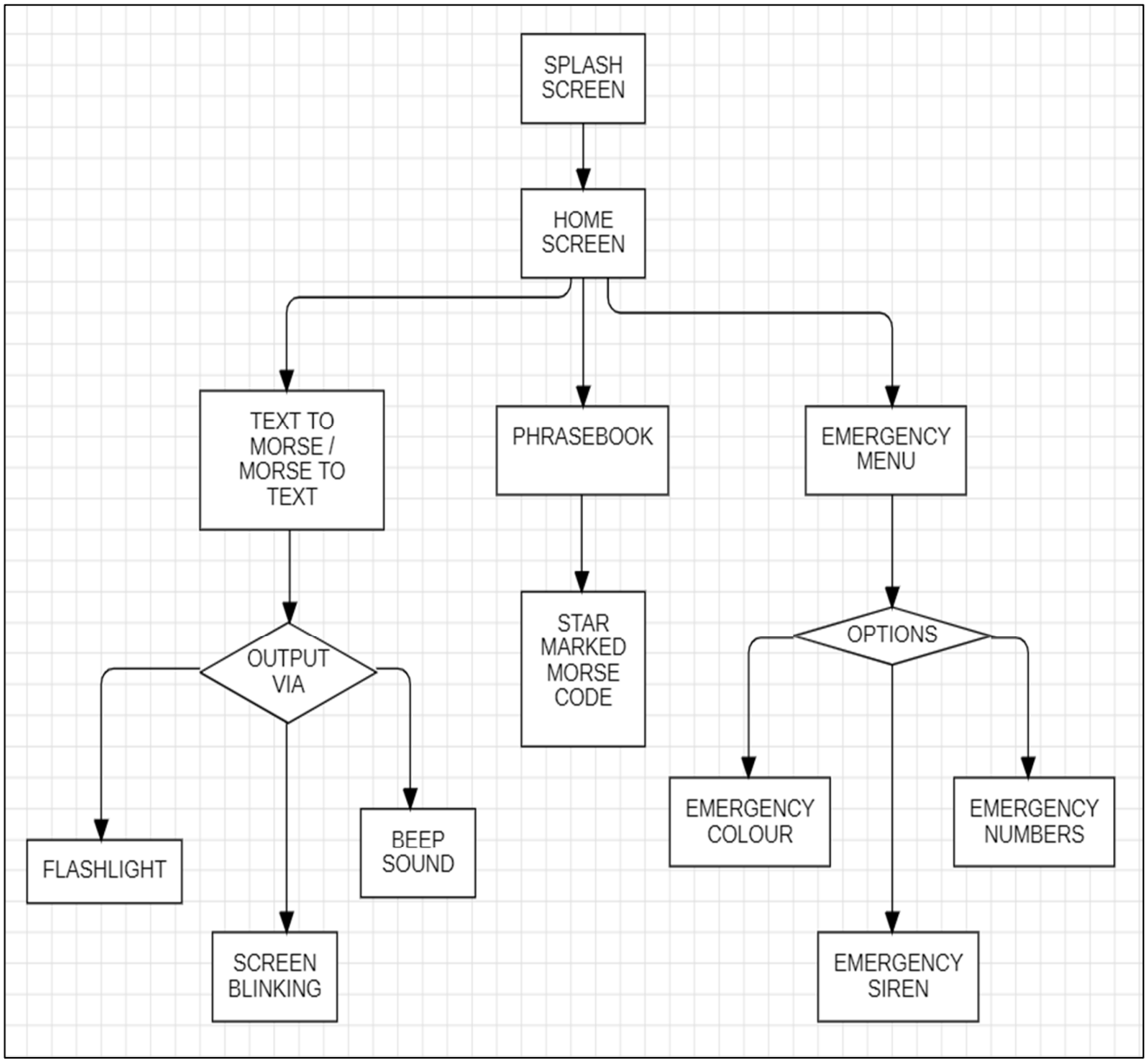
- Minimum Requirements:
    - OS: Android 6.0 (Marshmallow) or higher
    - Processor: Quad-core CPU
    - RAM: 2 GB
    - Storage: 50 MB free space
    - Display: 720p resolution (1280 x 720)

- Recommended Requirements:
    - OS: Android 8.0 (Oreo) or higher
    - Processor: Octa-core CPU
    - RAM: 4 GB or more
    - Storage: 100 MB free space
    - Display: Full HD (1920 x 1080)

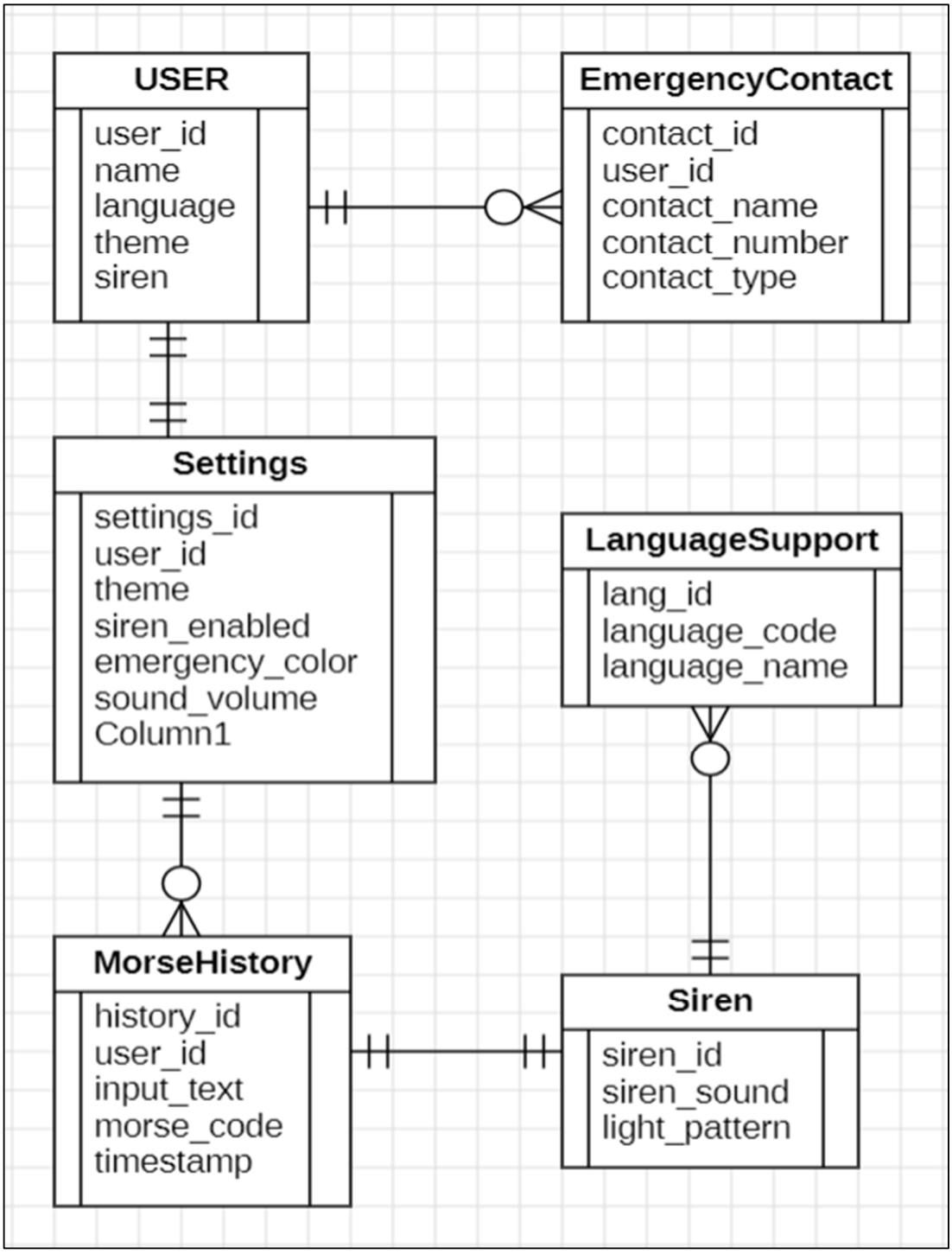
- Additional Permissions:
    - Phone: Emergency calling
    - Camera/Flashlight: Flashing light effects
    - Sound: Emergency siren

System Analysis & Design

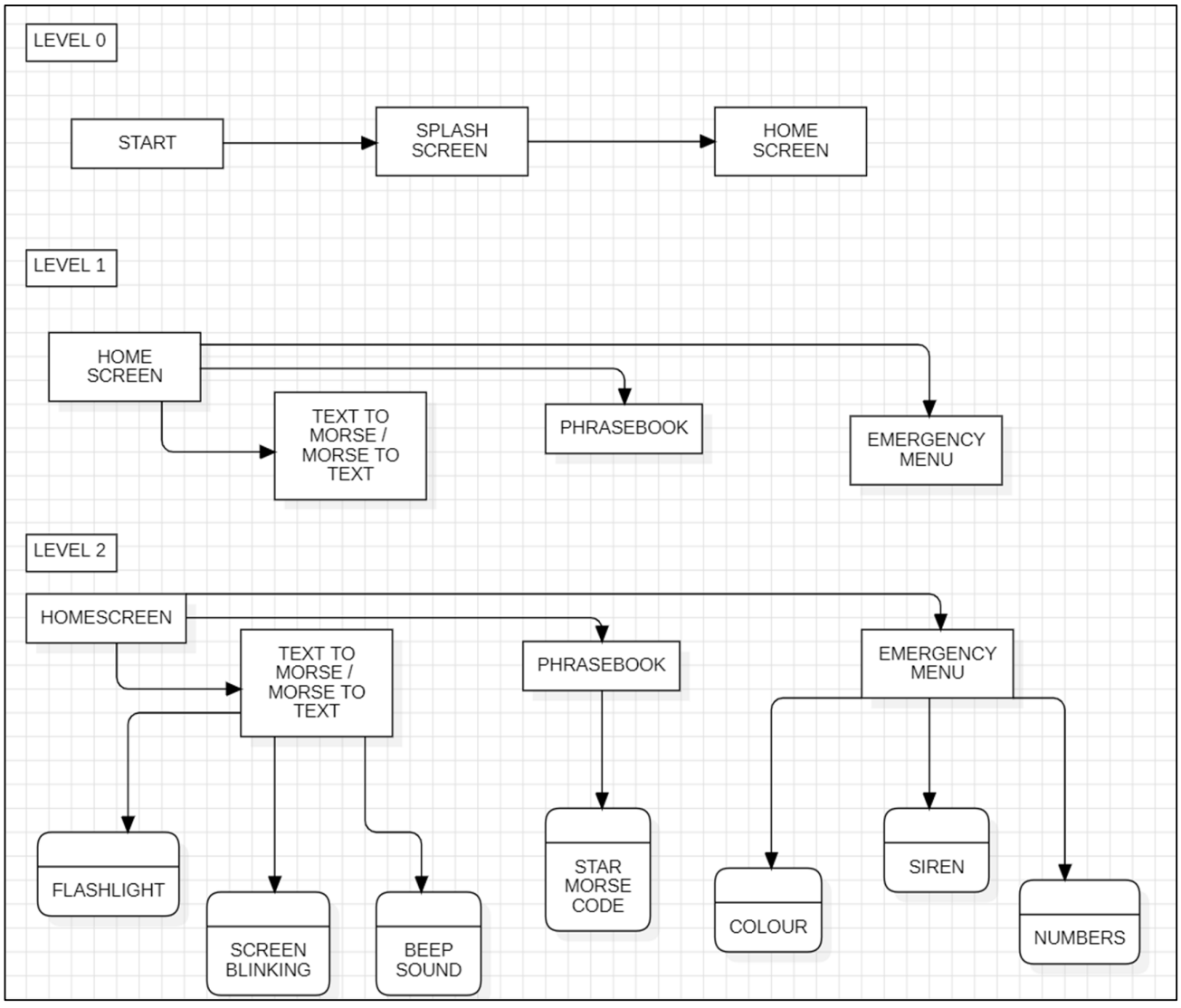
FLOW CHART


SPLASH SCREEN
HOME SCREEN
TEXT TO MORSE / MORSE TO TEXT
PHRASEBOOK
EMERGENCY MENU
OUTPUT VIA
STAR MARKED MORSE CODE
OPTIONS
FLASHLIGHT
BEEP SOUND
SCREEN BLINKING
EMERGENCY COLOUR
EMERGENCY NUMBERS
EMERGENCY SIREN

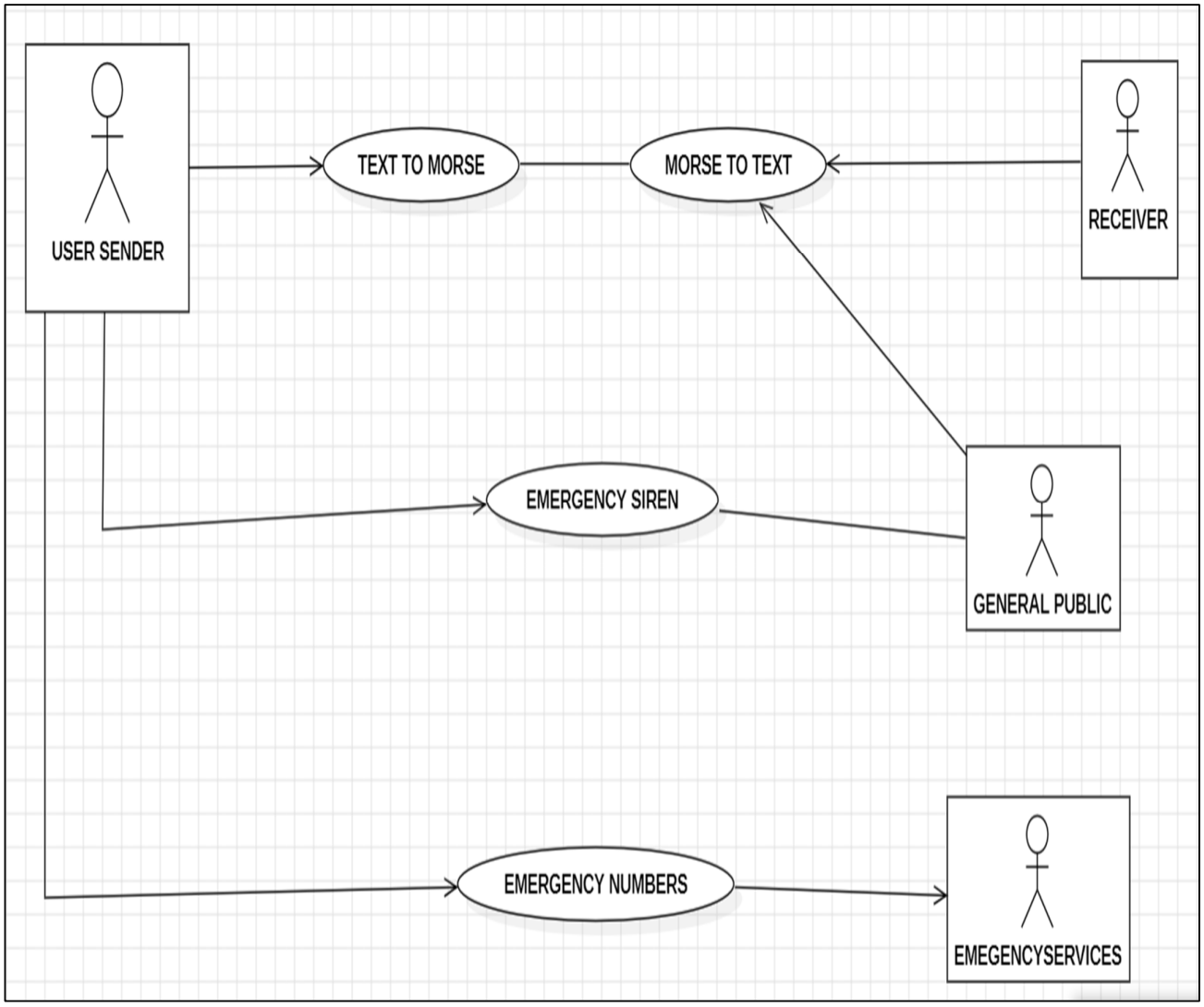
ENTITY-RELATIONSHIP DIAGRAM



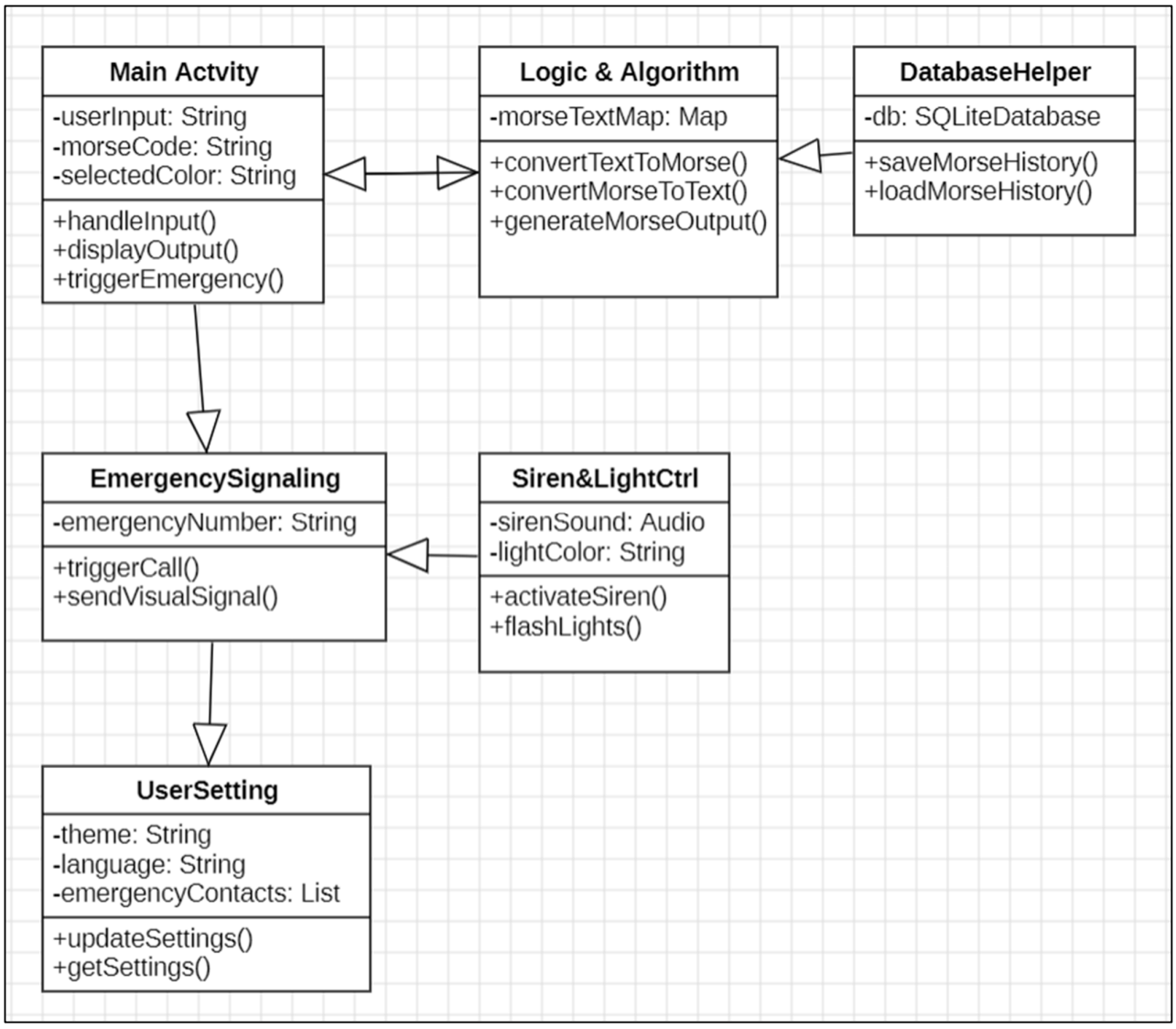
DATA FLOW DIAGRAM



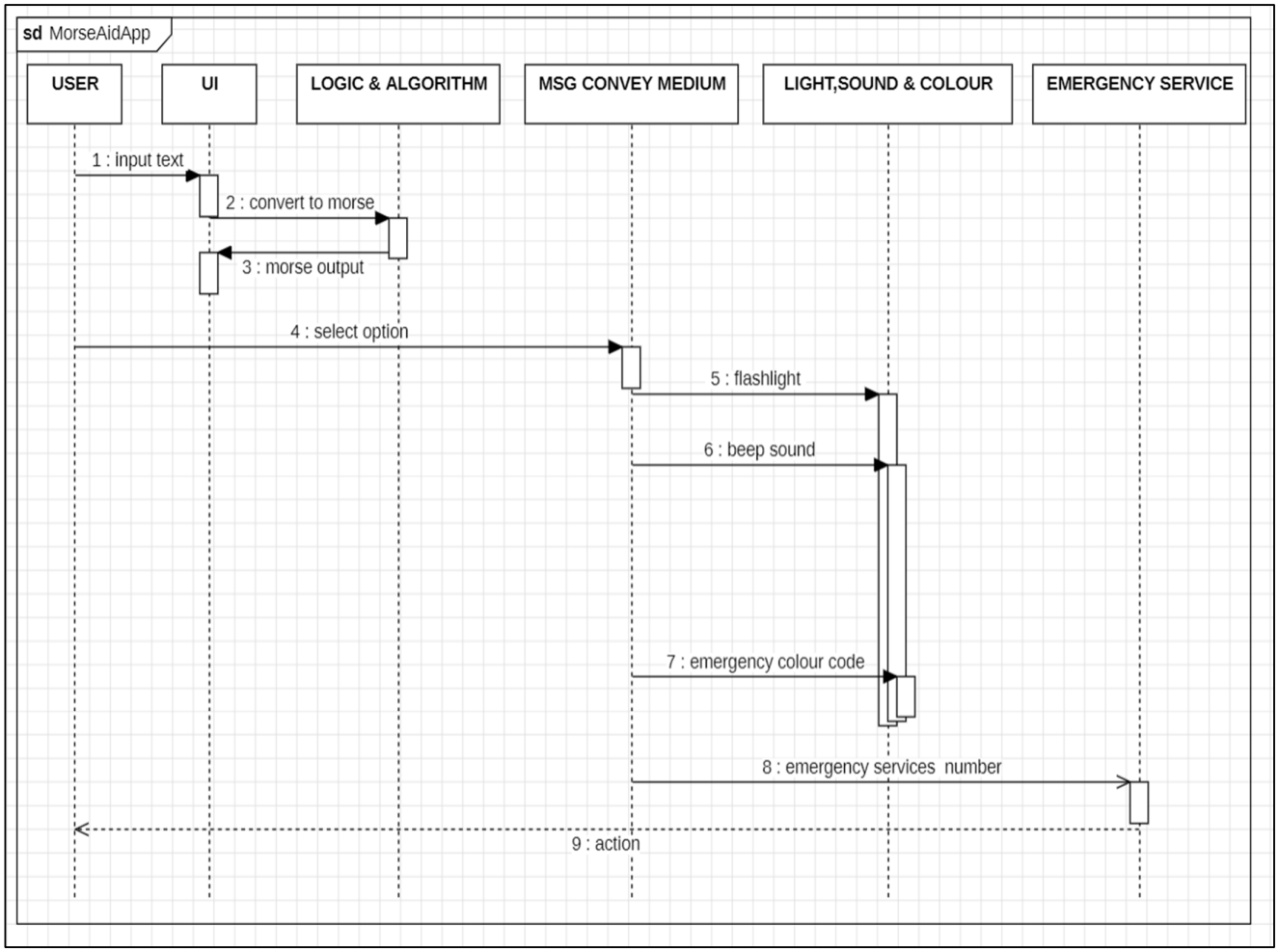
USE CASE DIAGRAM



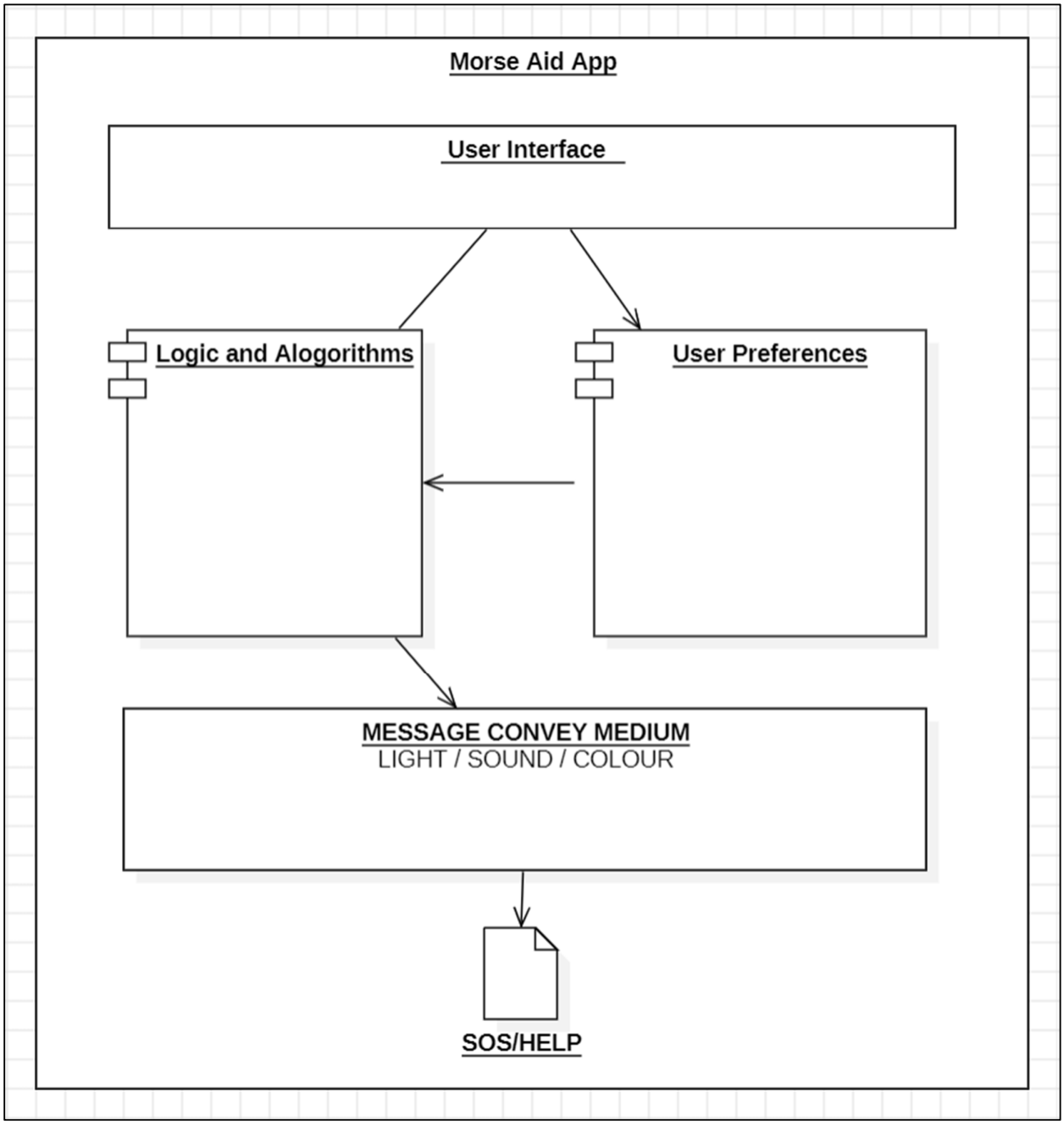
CLASS DIAGRAM



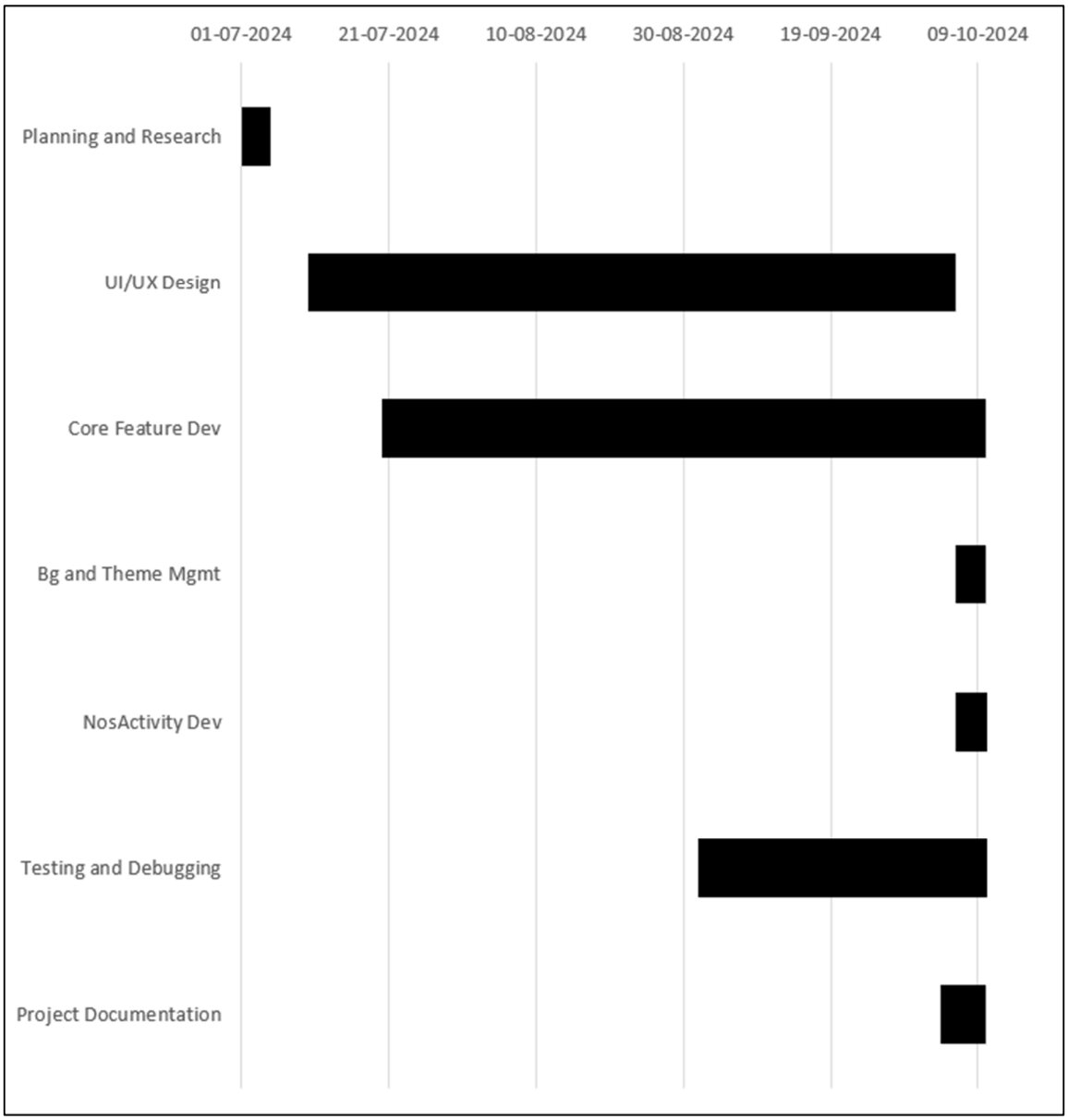
SEQUENCE DIAGRAM



COMPONENT DIAGRAM



GANTT CHART


01-07-2024
21-07-2024
10-08-2024
30-08-2024
19-09-2024
09-10-2024
Planning and Research
UI/UX Design
Core Feature Dev
Bg and Theme Mgmt
NosActivity Dev
Testing and Debugging
Project Documentation

App Structure & Programming

Source Code

AndroidManifest.xml

```
<?xml version="1.1" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools">
   <uses-permission android:name="android.permission.CAMERA" />
   <uses-feature android:name="android.hardware.camera" />
   <uses-feature android:name="android.hardware.camera.autofocus" />
   <uses-permission android:name="android.permission.WAKE_LOCK" />
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:localeConfig="@xml/locales_config"
      android:theme="@style/AppTheme"
      tools:ignore="GoogleAppIndexingWarning"
      tools:targetApi="tiramisu"
      android:forceDarkAllowed="false">
      <uses-library android:name="org.apache.http.legacy"
android:required="false"/>
      <activity android:name=".MainActivity"
         android:windowSoftInputMode="adjustNothing"
         android:exported="true">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <activity android:name=".NosActivity" />
      <activity android:name=".PhraseBookActivity" />
      <activity android:name=".ColorActivity" />
      <activity android:name=".SirenActivity" />
      <activity android:name=".ColorDisplayActivity" />
   </application>
</manifest>
```

MainActivity.java

```
package rocks.poopjournal.morse;
import static android.hardware.Camera.Parameters.FLASH_MODE_AUTO;
import static android.hardware.Camera.Parameters.FLASH_MODE_ON;
import static android.hardware.Camera.Parameters.FLASH_MODE_TORCH;
import static android.os.Process.*;
import android.Manifest;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.Rect;
import android.graphics.SurfaceTexture;
import android.graphics.Typeface;
import android.graphics.drawable.Drawable;
import android.graphics.drawable.GradientDrawable;
import android.hardware.Camera;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.SoundPool;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerThread;
import android.os.Looper;
import android.os.SystemClock;
import android.text.Editable;
import android.text.SpannableStringBuilder;
import android.text.Spanned;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.text.style.ForegroundColorSpan;
```

```
import android.text.style.StyleSpan;
import android.util.Log;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.view.inputmethod.InputMethodManager;
import android.widget.ImageView;
import android.widget.PopupMenu;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.Nullable;
import androidx.annotation.WorkerThread;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.drawable.DrawableCompat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.concurrent.atomic.AtomicBoolean;
public class MainActivity extends AppCompatActivity implements
Camera.AutoFocusCallback {
   public static Camera camera = null;
   final AtomicBoolean textToMorse = new AtomicBoolean(true);
   ImageView settings, history, flash;
   TextView buttonOne;
   TextView buttonTwo;
   RelativeLayout switchImageContainer;
   EditTextTouch input;
   RelativeLayout popularMorseSuggestionContainer;
   TextView output;
   ImageView copy;
   ImageView sound;
   RelativeLayout container, mic, fullscreen;
   int global_counter = 0;
   RelativeLayout bottomNavigation;
```

```
RelativeLayout flare_view;
RelativeLayout morseInputContainer;
RelativeLayout dot;
RelativeLayout dash;
RelativeLayout space;
RelativeLayout makeInputVisible;
RelativeLayout backspace;
@Nullable RelativeLayout containerTools;
@Nullable  RelativeLayout telegraphContainer;
@Nullable RelativeLayout telegraphAudio;
@Nullable RelativeLayout telegraphFlash;
@Nullable RelativeLayout telegraphKey;
@Nullable ImageView telegraphKeyboard;
@Nullable ImageView telegraphFlashIV;
@Nullable ImageView telegraphAudioIV;
boolean visibilityCheck = false;
ArrayList<String> popularMorse = new ArrayList<>();
HashMap<String, String> popularMorseConversion = new HashMap<>();
HashMap<String, String> popularMorseConversionText = new HashMap<>();
String flashText = null;
DBHelper helper;
ImageView star;
ArrayList<PhrasebookModel> arrayList;
private int telegraphSelected = 1;
MediaPlayer telegraphPlayer = null;
private final SoundPool morseSoundPool = new SoundPool(2,
AudioManager.STREAM_MUSIC, 0);
private final HashMap<MorseSoundType, MorseSound> morseSoundMap = new
HashMap<>(2);
private final HandlerThread morseSoundThread = new
HandlerThread("MorseSoundThread", THREAD_PRIORITY_BACKGROUND);
private Handler morseSoundHandler;
long time =0;
ViewTreeObserver.OnGlobalLayoutListener listener = new
ViewTreeObserver.OnGlobalLayoutListener() {
   @Override
   public void onGlobalLayout() {
      final String[] something = flashText.trim().replaceAll("\\s+", "").split("(?!^)");
      Log.d("flare", "flash_text: " + flashText);
```

```
        Log.d("flare", "counter: " + global_counter);
        if (global_counter == something.length) {
            flare_view.setVisibility(View.GONE);
            flare_view.setTag(flare_view.getVisibility());
            flare_view.getViewTreeObserver().removeOnGlobalLayoutListener(this);
            return;
        }
        if ((int) flare_view.getTag() == View.VISIBLE) {
            Log.d("flare", "here in if");
            if (something[global_counter].equals(".")) {
                Log.d("flare", "dot");
                final Handler handler = new Handler();
                handler.postDelayed(() -> {
                    Log.d("flare", "dot_post_delayed");
                    flare_view.setVisibility(View.GONE);
                    global_counter++;
                    flare_view.setTag(flare_view.getVisibility());
                    if (global_counter != something.length) {
                        handler.postDelayed(MainActivity.this::flash_display, 150);
                    }
                }, 350);
            } else if (something[global_counter].equals("-")) {
                final Handler handler = new Handler();
                Log.d("flare", "dash");
                handler.postDelayed(() -> {
                    Log.d("flare", "dash_post_delayed");
                    flare_view.setVisibility(View.GONE);
                    global_counter++;
                    flare_view.setTag(flare_view.getVisibility());
                    if (global_counter != something.length) {
                        handler.postDelayed(MainActivity.this::flash_display, 150);
                    }
                }, 1000);
            } else {
                Log.d("flare", "can't identifyt");
            }}}};
    public static void setMargins(View v, int l, int t, int r, int b) {
        if (v.getLayoutParams() instanceof ViewGroup.MarginLayoutParams) {
            ViewGroup.MarginLayoutParams p = (ViewGroup.MarginLayoutParams)
    v.getLayoutParams();
```

```
        p.setMargins(l, t, r, b);
        v.requestLayout();
    }}
public static void hideKeyboard(Activity activity) {
    InputMethodManager imm = (InputMethodManager)
activity.getSystemService(Activity.INPUT_METHOD_SERVICE);
    View view = activity.getCurrentFocus();
    if (view == null) {
        view = new View(activity);
    }
    imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
}
static String morseEncode(String x) {
    switch (x.toLowerCase(Locale.getDefault())) {
        case "a":
            return ".-";
        case "b":
            return "-...";
        case "c":
            return "-.-.";
        case "d":
            return "-..";
        case "e":
            return ".";
        case "f":
            return "..-.";
        case "g":
            return "--.";
        case "h":
            return "....";
        case "i":
            return "..";
        case "j":
            return ".---";
        case "k":
            return "-.-";
        case "l":
            return ".-..";
        case "m":
            return "--";
```

```
case "n":
   return "-.";
case "o":
   return "---";
case "p":
   return ".--.";
case "q":
   return "--.-";
case "r":
   return ".-.";
case "s":
   return "...";
case "t":
   return "-";
case "u":
   return "..-";
case "v":
   return "...-";
case "w":
   return ".--";
case "x":
   return "-..-";
case "y":
   return "-.--";
case "  ":
   return " ";
case "z":
   return "--..";
case " ":
   return "/ ";
case "0":
   return "-----";
case "1":
   return ".----";
case "2":
   return "..---";
case "3":
   return "...--";
case "4":
   return "....-";
```

```
case "5":
   return ".....";
case "6":
   return "-....";
case "7":
   return "--...";
case "8":
   return "---..";
case "9":
   return "----.";
case "а":
   return ".-";
case "б":
   return "-...";
case "в":
   return ".--";
case "г":
   return "--.";
case "д":
   return "-..";
case "е":
   return ".";
case "ё":
   return ".";
case "ж":
   return "...-";
case "з":
   return "--..";
case "и":
   return "..";
case "й":
   return ".---";
case "к":
   return "-.-";
case "л":
   return ".-..";
case "м":
   return "--";
case "н":
return "-.";
```

```
case "о":
   return "---";
case "п":
   return ".--.";
case "р":
   return ".-.";
case "с":
   return "...";
case "т":
   return "-";
case "у":
   return "..-";
case "ф":
   return "..-.";
case "х":
   return "....";
case "ц":
   return "-.-.";
case "ч":
   return "---.";
case "ш":
   return "----";
case "щ":
   return "--.-";
case "ъ":
   return "--.--";
case "ы":
   return "-.--";
case "ь":
   return "-..-";
case "э":
   return "..-..";
case "ю":
   return "..--";
case "я":
   return ".-.-";
case "ñ":
   return "--.--";
   default:
      return "";
```

```
static String morseDecode(String morse) {
   switch (morse) {
      case ".-":
         return "a";
      case "-...":
         return "b";
      case "-.-.":
         return "c";
      case "-..":
         return "d";
      case ".":
         return "e";
      case "..-.":
         return "f";
      case "--.":
         return "g";
      case "....":
         return "h";
      case "..":
         return "i";
      case ".---":
         return "j";
      case "-.-":
         return "k";
      case ".-..":
         return "l";
      case "--":
         return "m";
      case "-.":
         return "n";
      case "---":
         return "o";
      case ".--.":
         return "p";
      case "--.-":
         return "q";
      case ".-.":
         return "r";
      case "...":
         return "s";
      case "-":
         return "t";
      case "..-":
         return "u";
      case "...-":
         return "v";
      case ".--":
         return "w";
      case "-..-":
         return "x";
      case "-.--":
         return "y";
      case "--..":
         return "z";
      case "   ":
         return " ";
      case "/ ":
         return " ";
      case "-----":
         return "0";
      case ".----":
         return "1";
      case "..---":
         return "2";
      case "...--":
         return "3";
      case "....-":
         return "4";
      case ".....":
         return "5";
      case "-....":
         return "6";
      case "--...":
         return "7";
      case "---..":
         return "8";
      case "----.":
         return "9";
      case "--.--":
```

```
@SuppressLint("ClickableViewAccessibility")
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   settings = findViewById(R.id.settings);
   fullscreen = findViewById(R.id.input_fullscreen_container);
   history = findViewById(R.id.history);
   buttonOne = findViewById(R.id.buttonOne);
   flare_view = findViewById(R.id.flare_view);
   buttonTwo = findViewById(R.id.buttonTwo);
   switchImageContainer = findViewById(R.id.switchImageContainer);
   input = findViewById(R.id.input);
   output = findViewById(R.id.output);
   copy = findViewById(R.id.copyText);
   sound = findViewById(R.id.playAudio);
   flash = findViewById(R.id.flash);
   popularMorseSuggestionContainer =
findViewById(R.id.bottom_suggestion_container);
   telegraphAudio = findViewById(R.id.rl_audio_telegraph);
   telegraphContainer = findViewById(R.id.morseTelegraphContainer);
   telegraphFlash = findViewById(R.id.rl_flash_telegraph);
   telegraphFlashIV = findViewById(R.id.flash_telegraph);
telegraphAudioIV = findViewById(R.id.audio_telegraph);
telegraphKey = findViewById(R.id.container_dits_dah);
telegraphKeyboard = findViewById(R.id.keyboard_telegraph);
containerTools = findViewById(R.id.container_tools);
ImageView settingsImageView = findViewById(R.id.settings);
settingsImageView.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      showPopupMenu(view);
   }});
helper = new DBHelper(getApplicationContext());
initMorseSounds(getApplicationContext());
morseSoundThread.start();
morseSoundHandler = new Handler(morseSoundThread.getLooper());
star = findViewById(R.id.star);
popularMorse.add("...---...");
```

```
popularMorse.add("-.-.--.--..");
popularMorse.add(".--.....--.....--....--.----...--.-.---..---.....-");
popularMorse.add(".-..--...");
flare_view.setVisibility(View.GONE);
popularMorseConversion.put("...---...", "... --- ...");
popularMorseConversion.put("-.-.--.--..", "-.-. --.- -..");
popularMorseConversion.put(".--.....--.....--....--.----...--.-.---..---.....-", ".-- .... .- -   .... .- -
....   --. --- -..   .-- .-. --- ..- --. .... -");
popularMorseConversion.put(".-..--...", ".-. .- - ...");
popularMorseConversionText.put("...---...", "SOS");
popularMorseConversionText.put("-.-.--.--..", "CQD");
popularMorseConversionText.put(".--.....--.....--....--.----...--.-.---..---.....-", "What hath
God wrought");
popularMorseConversionText.put(".-..--...", "rats");
if (telegraphKeyboard != null) {
   telegraphKeyboard.setOnClickListener(view -> hideTelegraphKey());
}
fullscreen.setOnClickListener(view -> showTelegraphKey());
if (telegraphFlash != null) {
   telegraphFlash.setOnClickListener(v -> setFlashSelectedForTelegraph());
}
if (telegraphAudio != null) {
   telegraphAudio.setOnClickListener(v -> setAudioSelectedForTelegraph());
}
telegraphPlayer= MediaPlayer.create(MainActivity.this,R.raw.beepflac);
if (telegraphKey != null) {
   telegraphKey.setOnTouchListener((v, event) ->
setKeySelectedForTelegraph(event));
flash.setOnClickListener(view -> {
   int hasCameraPermission = 0;
   if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
      hasCameraPermission = checkSelfPermission(Manifest.permission.CAMERA);
   }
   if (hasCameraPermission != PackageManager.PERMISSION_GRANTED) {
      Toast.makeText(getApplicationContext(), "Please give camera permission to use
flash", Toast.LENGTH_SHORT).show();
      return;
   }
```

```
if (textToMorse.get()) {
   if (!TextUtils.isEmpty(output.getText().toString())) {
      String[] something =
TextUtils.split(output.getText().toString().trim().replaceAll("\\s+", ""), "");
      Log.d("test_string", output.getText().toString().trim().replace(" ", "").replace("
", ""));
      Log.d("test_string", ".....");
      Log.d("test_length_string", String.valueOf(something.length));
      for (String s : something) {
         Log.d("skkk", s);
      }
      int len = something.length;
      int currentCounter = 0;
      if (!Build.MANUFACTURER.equals("HUAWEI")) {
         openCamera();
      }
      for (String s : something) {
         if (s.equals(".")) {
            turnOn();
            SystemClock.sleep(200);
            turnOff();
         } else if (s.equals("-")) {
            turnOn();
            SystemClock.sleep(600);
            turnOff();
         }}
      releaseCamera();
   }
} else {
   if (!TextUtils.isEmpty(input.getText().toString())) {
      String[] something =
TextUtils.split(input.getText().toString().trim().replaceAll("\\s+", ""), "");
      Log.d("test_string", input.getText().toString().trim().replace(" ", "").replace("
", ""));
      Log.d("test_string", ".....");
      Log.d("test_length_string", String.valueOf(something.length))
      ;
      for (String s : something) {
         Log.d("skkk", s);
```

```
            int len = something.length;
            int currentcounter = 0;
            for (String s : something) {
               if (s.equals(".")) {
                  turnOn();
                  SystemClock.sleep(200);
                  turnOff();
               } else if (s.equals("-")) {
                  turnOn();
                  SystemClock.sleep(600);
                  turnOff();
               }}
            releaseCamera();
         }}});
history.setOnClickListener(view -> startActivity(new Intent(MainActivity.this,
PhraseBookActivity.class)));
settings.setOnClickListener(v -> showPopupMenu(v));
container = findViewById(R.id.container);
bottomNavigation = findViewById(R.id.bottomLayout);
morseInputContainer = findViewById(R.id.morseInputContainer);
makeInputVisible = findViewById(R.id.input_visible_container);
makeInputVisible.setOnClickListener(v -> {
   if (morseInputContainer.getVisibility() == View.GONE)
      morseInputContainer.setVisibility(View.VISIBLE);
   else {
      morseInputContainer.setVisibility(View.GONE);
   }
});
copy.setOnClickListener(v -> {
   ClipboardManager clipboard = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
   ClipData clip = ClipData.newPlainText("something",
output.getText().toString());
   clipboard.setPrimaryClip(clip);
   Toast.makeText(getApplicationContext(), "Text copied",
Toast.LENGTH_SHORT).show();
addKeyboardVisibilityListener(container, isVisible -> visibilityCheck = isVisible);
final ImageView flare = findViewById(R.id.flare);
final Runnable hide = () -> {
   flare_view.setVisibility(View.GONE);
```

```
    Log.d("flare", "set to gone");
};
star.setOnClickListener(view -> {
    if (!TextUtils.isEmpty(output.getText().toString().trim()) &&
!TextUtils.isEmpty(input.getText().toString().trim())) {
        if (textToMorse.get())
            helper.addPhrase(input.getText().toString(), output.getText().toString());
        else {
            helper.addPhrase(output.getText().toString(), input.getText().toString());
        }}
    arrayList = helper.getAllPhrases();
    checkForStarColor();
});
flare.setOnClickListener(view -> {
    if (textToMorse.get()) {
        if (!TextUtils.isEmpty(output.getText().toString())) {
            flare_view.getViewTreeObserver().addOnGlobalLayoutListener(listener);
            flashText = output.getText().toString().trim().replace(" ",
"").replaceAll("\\s+", "");
            global_counter = 0;
            flash_display();
        }
    } else {
        if (!TextUtils.isEmpty(input.getText().toString())) {
            flare_view.getViewTreeObserver().addOnGlobalLayoutListener(listener);
            flashText = input.getText().toString().trim().replace(" ",
"").replaceAll("\\s+", "");
            global_counter = 0;
            flash_display();
        }}});
sound.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (textToMorse.get()) {
            if (!TextUtils.isEmpty(output.getText().toString())) {
                String[] sequence =
TextUtils.split(output.getText().toString().trim().replaceAll("\\s+", ""), "");
                playSoundSequence(sequence);
            }
} else {
```

```
            if (!TextUtils.isEmpty(input.getText().toString())) {
                String[] sequence =
TextUtils.split(input.getText().toString().trim().replaceAll("\\s+", ""), "");
                playSoundSequence(sequence);
            }}}});
switchImageContainer.setOnClickListener(v -> {
    if (textToMorse.get()) {
        buttonOne.setText("MORSE");
        buttonTwo.setText("TEXT");
        input.setText("");
        output.setText("");
        textToMorse.set(false);
        bottomNavigation.setVisibility(View.VISIBLE);
        morseInputContainer.setVisibility(View.VISIBLE);
        if (visibilityCheck)
            hideKeyboard(MainActivity.this);
        dot = findViewById(R.id.inputDotContainer);
        dash = findViewById(R.id.inputDashContainer);
        space = findViewById(R.id.input_space_container);
        backspace = findViewById(R.id.input_clear_container);
        dot.setOnClickListener(v1 -> {
            input.append(".");
            Log.d("test", "clicked");
        });
        dash.setOnClickListener(v12 -> input.append("-"));
        space.setOnClickListener(v13 -> input.append(" "));
        backspace.setOnClickListener(v14 -> {
            if (input.getText().toString().length() == 0)
                return;
            input.setText(input.getText().toString().substring(0,
input.getText().toString().length() - 1));
            input.setSelection(input.getText().toString().length());
        });
        backspace.setOnLongClickListener(v15 -> {
            input.setText("");
            return false;
        });
} else {
    input.setText("");
    output.setText("");
```

```
        buttonOne.setText("TEXT");
        buttonTwo.setText("MORSE");
        textToMorse.set(true);
        bottomNavigation.setVisibility(View.GONE);
        morseInputContainer.setVisibility(View.GONE);
    }});
input.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after)
{}
    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        if (TextUtils.isEmpty(s.toString())) {
            output.setText("");
            return;
        }
        if (textToMorse.get()) {
            output.setText("");
            String text = input.getText().toString();
            String[] letters = text.split("");
            for (String letter : letters) {
                output.append(morseEncode(letter) + " ");
            }
        } else {
            if (popularMorse.contains(input.getText().toString())) {
                Log.d("popular_morse", "true");
                RelativeLayout.LayoutParams params = (RelativeLayout.LayoutParams)
input.getLayoutParams();
                setMargins(input, params.leftMargin, params.topMargin,
params.rightMargin, 2);

input.setBackground(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.bg_top_suggestion));
                popularMorseSuggestionContainer.setVisibility(View.VISIBLE);
                TextView suggestionTV = findViewById(R.id.suggestion_text_tv);
                TextView replace = findViewById(R.id.replace_suggestion);
                TextView ignore = findViewById(R.id.ignore_suggestion);
```

```
suggestionTV.setText("Did you mean " +
popularMorseConversion.get(input.getText().toString()) + " (" +
popularMorseConversionText.get(input.getText().toString()) + ")?");
            change(suggestionTV.getText().toString(),
popularMorseConversion.get(input.getText().toString()), suggestionTV);
            replace.setOnClickListener(view -> {

input.setText(popularMorseConversion.get(input.getText().toString()));
                input.setSelection(input.getText().length());
                Toast.makeText(getApplicationContext(), "Changed morse",
Toast.LENGTH_SHORT).show();
                RelativeLayout.LayoutParams params1 =
(RelativeLayout.LayoutParams) input.getLayoutParams();
                setMargins(input, params1.leftMargin, params1.topMargin,
params1.rightMargin, 20);

input.setBackground(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.et_morse));
                popularMorseSuggestionContainer.setVisibility(View.GONE);
            });
            ignore.setOnClickListener(view -> {
                RelativeLayout.LayoutParams params12 =
(RelativeLayout.LayoutParams) input.getLayoutParams();
                setMargins(input, params12.leftMargin, params12.topMargin,
params12.rightMargin, 20);

input.setBackground(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.et_morse));
                popularMorseSuggestionContainer.setVisibility(View.GONE);
            });
        } else {
            RelativeLayout.LayoutParams params = (RelativeLayout.LayoutParams)
input.getLayoutParams();
            setMargins(input, params.leftMargin, params.topMargin,
params.rightMargin, 20);

input.setBackground(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.et_morse));
            popularMorseSuggestionContainer.setVisibility(View.GONE);
        }
```

```
            output.setText("");
            String text = input.getText().toString();
            String[] letters = text.split("\\s");
            for (String morse : letters) {
               if (morse.equals("/")){
                  morse = morse + " ";
               }
               output.append(morseDecode(morse));
            }}}
      @Override
      public void afterTextChanged(Editable s) {
         checkForStarColor();
      }
   });}
private void initMorseSounds(Context context) {
   MediaPlayer player = MediaPlayer.create(context, R.raw.dot);
   int dotDuration = player.getDuration();
   player.release();
   MorseSound dot = new MorseSound(MorseSoundType.DOT, ".", R.raw.dot,
         morseSoundPool.load(this, R.raw.dot, 1), dotDuration);
   player = MediaPlayer.create(context, R.raw.dash);
   int dashDuration = player.getDuration();
   player.release();
   MorseSound dash = new MorseSound(MorseSoundType.DASH, "-", R.raw.dash,
         morseSoundPool.load(this, R.raw.dash, 1), dashDuration);
   morseSoundMap.put(MorseSoundType.DOT, dot);
   morseSoundMap.put(MorseSoundType.DASH, dash);
}
private void playSoundSequence(String[] sequence) {
   int len = sequence.length;
   ArrayList<MorseSoundType> soundSequence = new ArrayList<>(len);
   for (String s : sequence) {
      if (s.equals(".")) {
         soundSequence.add(MorseSoundType.DOT);
      } else if (s.equals("-")) {
         soundSequence.add(MorseSoundType.DASH);
      }}
   AudioManager audioManager =
         (AudioManager)
getApplicationContext().getSystemService(Context.AUDIO_SERVICE);
```

```
   float streamVolumeCurrent =
audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
   float streamVolumeMax =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
   float volume = streamVolumeCurrent / streamVolumeMax;
   MorseSoundRunnable morseSoundRunnable =
         new MorseSoundRunnable(morseSoundMap, soundSequence,
morseSoundPool, volume);
   morseSoundHandler.removeCallbacksAndMessages(null);
   morseSoundHandler.post(morseSoundRunnable);
}
@WorkerThread
private static class MorseSoundRunnable implements Runnable {
   private final HashMap<MorseSoundType, MorseSound> soundMap;
   private final ArrayList<MorseSoundType> soundSequence;
   private final SoundPool soundPool;
   private final float volume;
   private static final int PAUSE_MS = 200;
   private MorseSoundRunnable(HashMap<MorseSoundType, MorseSound>
soundMap,
                        ArrayList<MorseSoundType> soundSequence,
                        SoundPool soundPool,
                        float volume) {
      this.soundMap = soundMap;
      this.soundSequence = soundSequence;
      this.soundPool = soundPool;
      this.volume = volume;
   }
   @Override
   public void run() {
      for (MorseSoundType soundType : soundSequence) {
         MorseSound sound = soundMap.get(soundType);
         soundPool.play(sound.soundPoolSoundId, volume, volume, 1, 0, 1.0f);
         SystemClock.sleep(sound.soundLength + PAUSE_MS);
      }}}
private class MorseSound {
   final MorseSoundType soundType;
   final String textual;
   final int soundResId;
   final int soundPoolSoundId;
```

```
   final int soundLength;
   MorseSound(MorseSoundType soundType, String textual, int soundResId, int
soundPoolSoundId, int soundLength) {
      this.soundType = soundType;
      this.textual = textual;
      this.soundResId = soundResId;
      this.soundPoolSoundId = soundPoolSoundId;
      this.soundLength = soundLength;
   }}
enum MorseSoundType {
   DOT,
   DASH
}
private void checkForStarColor() {
   boolean isTrue = false;
   if (arrayList == null || arrayList.size() == 0) {
      star.setImageDrawable(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.ic_star_border_black_24dp));
      star.setColorFilter(Color.parseColor("#7DD3D8"));
      Log.d("debug_star", "did not match new text: " + input.getText().toString() + ",
setting star to off");
      return;
   }
   if (textToMorse.get()) {
      for (PhrasebookModel model : arrayList) {
         if (model.text.trim().equals(input.getText().toString().trim())) {

star.setImageDrawable(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.ic_star_black_24dp));
            star.setColorFilter(Color.parseColor("#F9A825"));
            Log.d("debug_star", "matched new text: " + input.getText().toString() + ",
setting star to on");
            return;
         } else {
            isTrue = true;
         }}
      if (isTrue) {

star.setImageDrawable(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.ic_star_border_black_24dp));
```

```
            star.setColorFilter(Color.parseColor("#7DD3D8"));
            Log.d("debug_star", "did not match new text: " + input.getText().toString()
+ ", setting star to on");
         }
      } else {
         for (PhrasebookModel model : arrayList) {
            if (model.text.trim().equals(output.getText().toString().trim())) {

star.setImageDrawable(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.ic_star_black_24dp));
               star.setColorFilter(Color.parseColor("#F9A825"));
               Log.d("debug_star", "matched new text: " + output.getText().toString()
+ ", setting star to on");
               return;
            } else {

               isTrue = true;
            }
         }
         if (isTrue) {

star.setImageDrawable(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.ic_star_border_black_24dp));
            star.setColorFilter(Color.parseColor("#7DD3D8"));
            Log.d("debug_star", "did not match new text: " +
output.getText().toString() + ", setting star to on");
         }}}
void flash_display() {
   flare_view.setVisibility(View.VISIBLE);
   flare_view.bringToFront();
   flare_view.setTag(flare_view.getVisibility());
}
public void addKeyboardVisibilityListener(
      final View rootView, final KeyboardVisibilityCallback callback) {
   rootView.getViewTreeObserver().addOnGlobalLayoutListener(
         () -> {
            Rect r = new Rect();
            rootView.getWindowVisibleDisplayFrame(r);
            int screenHeight = rootView.getRootView().getHeight();
            int keypadHeight = screenHeight - r.bottom;
```

```
            if (keypadHeight > screenHeight * 0.15) {
                callback.onChange(true);
            } else {
                callback.onChange(false);
            }
        });}
@Override
public void onAutoFocus(boolean b, Camera camera) {}
public void turnOn() {
    try {
        if (android.os.Build.MANUFACTURER.equals("HUAWEI")) {
            openCamera();
        }
        Log.d("cameraMorseCheck","turning camera on at " +
System.currentTimeMillis());
        Camera.Parameters parameters = camera.getParameters();
        parameters.setFlashMode(getFlashOnParameter());
        camera.setParameters(parameters);
        camera.setPreviewTexture(new SurfaceTexture(0));
        camera.startPreview();
        camera.autoFocus(this);
    } catch (Exception e) {}
}
private String getFlashOnParameter() {
    List<String> flashModes =
camera.getParameters().getSupportedFlashModes();
    if (flashModes.contains(FLASH_MODE_TORCH)) {
        return FLASH_MODE_TORCH;
    } else if (flashModes.contains(FLASH_MODE_ON)) {
        return FLASH_MODE_ON;
    } else if (flashModes.contains(FLASH_MODE_AUTO)) {
        return FLASH_MODE_AUTO;
    }
    throw new RuntimeException();
}
public void turnOff() {
    try {
        if (android.os.Build.MANUFACTURER.equals("HUAWEI")){
            camera.release();
```

```
            SystemClock.sleep(100);
        }
        else {
            camera.stopPreview();
        }
        Log.d("cameraMorseCheck","turning camera off at " +
System.currentTimeMillis());
    } catch (Exception e) {
        Log.d("cameraMorseCheck","exception caught: " + e.getMessage());
    }}
@Override
protected void onStart() {
    super.onStart();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        int hasCameraPermission =
checkSelfPermission(Manifest.permission.CAMERA);
        List<String> permissions = new ArrayList<>();
        if (hasCameraPermission != PackageManager.PERMISSION_GRANTED) {
            permissions.add(Manifest.permission.CAMERA);
        }
        if (!permissions.isEmpty()) {
            requestPermissions(permissions.toArray(new String[0]), 111);
        }}}
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions,
int[] grantResults) {
    if (requestCode == 111) {
        for (int i = 0; i < permissions.length; i++) {
            if (grantResults[i] == PackageManager.PERMISSION_GRANTED) {
                System.out.println("Permissions --> " + "Permission Granted: " +
permissions[i]);
            } else if (grantResults[i] == PackageManager.PERMISSION_DENIED) {
                System.out.println("Permissions --> " + "Permission Denied: " +
permissions[i]);
            }}
    } else {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
    }}
void change(String s, String newSuggestion, TextView t) {
```

```
   int i = s.indexOf(newSuggestion);
   SpannableStringBuilder sb = new SpannableStringBuilder(s);
   sb.setSpan(new
ForegroundColorSpan(ContextCompat.getColor(getApplicationContext(),
R.color.colorMorse)), i, i + newSuggestion.length(),
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
   int k = s.indexOf("(");
   sb.setSpan(new StyleSpan(Typeface.ITALIC), k + 1, k +
popularMorseConversionText.get(input.getText().toString()).length() + 1, 0);
   sb.setSpan(new
ForegroundColorSpan(ContextCompat.getColor(getApplicationContext(),
R.color.colorMorse)), k + 1, k +
popularMorseConversionText.get(input.getText().toString()).length() + 1, 0);
   t.setText(sb);
}
@Override
protected void onResume() {
   super.onResume();
   arrayList = helper.getAllPhrases();
}
@Override
public void onConfigurationChanged(Configuration newConfig) {
   super.onConfigurationChanged(newConfig);
   helper = new DBHelper(getApplicationContext());
   arrayList = helper.getAllPhrases();
}
public interface KeyboardVisibilityCallback {
   void onChange(boolean isVisible);
}
private class AsyncTaskRunner extends AsyncTask<String, String, String> {
   @Override
   protected String doInBackground(String... params) {
      publishProgress("Sleeping..."); // Calls onProgressUpdate()
      String resp;
      try {
         int time = Integer.parseInt(params[0]);
         Thread.sleep(time);
         resp = "Slept for " + params[0] + " seconds";
      } catch (Exception e) {
         e.printStackTrace();
```

```
            resp = e.getMessage();
        }
        return resp;
    }
    @Override
    protected void onPostExecute(String result) {
        flare_view.setVisibility(View.GONE);
    }
    @Override
    protected void onPreExecute() {
        flare_view.setVisibility(View.VISIBLE);
    }
    @Override
    protected void onProgressUpdate(String... text) {}
}
private void showTelegraphKey(){
    if (containerTools == null || telegraphContainer == null) {
        return;
    }
    bottomNavigation.setVisibility(View.GONE);
    containerTools.setVisibility(View.GONE);
    telegraphContainer.setVisibility(View.VISIBLE);
    setAudioSelectedForTelegraph();
}
private void hideTelegraphKey(){
    if (containerTools == null || telegraphContainer == null) {
        return;
    }
    bottomNavigation.setVisibility(View.VISIBLE);
    containerTools.setVisibility(View.VISIBLE);
    telegraphContainer.setVisibility(View.GONE);
}
private void setAudioSelectedForTelegraph(){
    Drawable d;
    if (telegraphAudio != null) {
        d = (GradientDrawable) telegraphAudio.getBackground();
        DrawableCompat.setTint(d, Color.parseColor("#227DD3D8"));
    }
    Drawable d2;
    if (telegraphFlash != null) {
```

```
        d2 = (GradientDrawable) telegraphFlash.getBackground();
        DrawableCompat.setTint(d2, Color.parseColor("#373945"));
    }
    if (telegraphFlashIV != null && telegraphAudioIV != null) {
        telegraphFlashIV.setColorFilter(Color.parseColor("#9C9CA4"),
PorterDuff.Mode.SRC_IN);
        telegraphAudioIV.setColorFilter(Color.parseColor("#7DD3D8"),
android.graphics.PorterDuff.Mode.SRC_IN);
    }
    telegraphSelected =1;
}
private void setFlashSelectedForTelegraph(){
    Log.d("flashselected","yes");
    Drawable d;
    if (telegraphFlash != null) {
        d = (GradientDrawable) telegraphFlash.getBackground();
        DrawableCompat.setTint(d, Color.parseColor("#227DD3D8"));
    }
    Drawable d2;
    if (telegraphAudio != null) {
        d2 = (GradientDrawable) telegraphAudio.getBackground();
        DrawableCompat.setTint(d2, Color.parseColor("#373945"));
    }
    if (telegraphFlashIV != null && telegraphAudioIV != null) {
        telegraphAudioIV.setColorFilter(Color.parseColor("#9C9CA4"),
PorterDuff.Mode.SRC_IN);
        telegraphFlashIV.setColorFilter(Color.parseColor("#7DD3D8"),
android.graphics.PorterDuff.Mode.SRC_IN);
    }
    telegraphSelected =2;
}
private boolean setKeySelectedForTelegraph(MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        if (telegraphSelected == 1) {
            time = System.currentTimeMillis();
            telegraphPlayer.start();
        } else {
            time = System.currentTimeMillis();
            openCamera();
            turnOn();
```

```
        }
        return true;
    }
    if (event.getAction() == MotionEvent.ACTION_UP) {
        if (telegraphSelected == 1) {
            if (System.currentTimeMillis() - time >= 200) {
                telegraphPlayer.pause();
                telegraphPlayer.seekTo(0);
            } else {
                final Handler handler = new Handler(Looper.getMainLooper());
                handler.postDelayed(() -> {
                    telegraphPlayer.pause();
                    telegraphPlayer.seekTo(0);
                }, 100);
            }
        } else {

            if (System.currentTimeMillis() - time >= 200) {
                turnOff();
                releaseCamera();
            } else {
                final Handler handler = new Handler(Looper.getMainLooper());
                handler.postDelayed(() -> {
                    turnOff();
                    releaseCamera();
                }, 100);
            }}
        return true;
    }
    return false;
}
private void openCamera() {
    camera = Camera.open();
}
private void releaseCamera() {
    if (camera != null) {
        camera.release();
        camera = null;
    }
}
```

```
private void showPopupMenu(View view) {
   PopupMenu popupMenu = new PopupMenu(this, view);
   popupMenu.getMenuInflater().inflate(R.menu.fn_menu,
popupMenu.getMenu());
   popupMenu.setOnMenuItemClickListener(new
PopupMenu.OnMenuItemClickListener() {
      @Override
      public boolean onMenuItemClick(MenuItem item) {
         int itemId = item.getItemId();
         if (itemId == R.id.color) {
            Intent intent = new Intent(MainActivity.this, ColorActivity.class);
            startActivity(intent);
            return true;
         } else if (itemId == R.id.siren) {
            Intent intent = new Intent(MainActivity.this, SirenActivity.class);
            startActivity(intent);
            return true;
         } else if (itemId == R.id.nos) {
            Intent intent = new Intent(MainActivity.this, NosActivity.class);
            startActivity(intent);
            return true;
         } else {
            return false;
         }}});
   popupMenu.show();
}}
```

ColourActivity.java

```
package rocks.poopjournal.morse;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
public class ColorActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.color_activity);
      setColorChangeListener(R.id.fire, "#FF0000");
      setColorChangeListener(R.id.me, "#0000FF");
      setColorChangeListener(R.id.bt, "#008000");
      setColorChangeListener(R.id.ica, "#FFC0CB");
      setColorChangeListener(R.id.hmse, "#ff8c00");
      setColorChangeListener(R.id.disaster, "#FFFF00");
      setAggressionOrViolenceListener();
   }
   private void setAggressionOrViolenceListener() {
      TextView textView = findViewById(R.id.av);
      textView.setOnClickListener(v -> {
         Intent intent = new Intent(ColorActivity.this,
ColorDisplayActivity.class);
         intent.putExtra("color", "#FFFFFF");
         startActivity(intent);
      });}
   private void setColorChangeListener(int textViewId, final String colorHex) {
      TextView textView = findViewById(textViewId);
      textView.setOnClickListener(v -> {
         Intent intent = new Intent(ColorActivity.this,
ColorDisplayActivity.class);
         intent.putExtra("color", colorHex);
         startActivity(intent);
      });}}
```

SirenActivity.java

```
package rocks.poopjournal.morse;
import android.animation.ArgbEvaluator;
import android.animation.ValueAnimator;
import android.graphics.Color;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;
import androidx.appcompat.app.AppCompatActivity;
public class SirenActivity extends AppCompatActivity {
   private MediaPlayer mediaPlayer;
   private boolean isSoundOn = false;
   private ImageButton soundButton;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.siren_activity);
      View fullScreenView = findViewById(R.id.fullScreenView);
      soundButton = findViewById(R.id.soundButton);
      mediaPlayer = MediaPlayer.create(this, R.raw.ambulance_siren);
      mediaPlayer.setLooping(true);
      ValueAnimator colorAnim = ValueAnimator.ofObject(new ArgbEvaluator(),
            Color.RED, Color.BLUE);
      colorAnim.setDuration(100);
      colorAnim.setRepeatCount(ValueAnimator.INFINITE);
      colorAnim.setRepeatMode(ValueAnimator.REVERSE);
      colorAnim.addUpdateListener(animator -> {
         int color = (int) animator.getAnimatedValue();
         fullScreenView.setBackgroundColor(color);
      });
      colorAnim.start();
      soundButton.setOnClickListener(v -> toggleSound());
   }
   private void toggleSound() {
      if (isSoundOn) {
         mediaPlayer.pause();
         soundButton.setImageResource(R.drawable.ic_speaker_off);
      } else {
         mediaPlayer.start();
         soundButton.setImageResource(R.drawable.ic_speaker_on);
      }
      isSoundOn = !isSoundOn;
   }
   @Override
   protected void onDestroy() {
      super.onDestroy();
      if (mediaPlayer != null) {
         mediaPlayer.stop();
         mediaPlayer.release();
         mediaPlayer = null;
      }}}
```

NosActivity.java

```
package rocks.poopjournal.morse;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;
public class NosActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.nos_activity);
      Button policeButton = findViewById(R.id.police);
      Button fireButton = findViewById(R.id.fire);
      Button ambulanceButton = findViewById(R.id.ambulance);
      Button disasterManagementButton = findViewById(R.id.disaster_management);
      Button nationalEmergencyButton = findViewById(R.id.national_emergency);
      Button womenHelplineButton = findViewById(R.id.women_helpline);
      Button covidHelplineButton = findViewById(R.id.covid_helpline);
      Button aidsHelplineButton = findViewById(R.id.aids_helpline);
      Button childrenHelplineButton = findViewById(R.id.children_helpline);
      Button touristHelplineButton = findViewById(R.id.tourist_helpline);
      Button lpgLeakButton = findViewById(R.id.lpg_leak);
      Button seniorCitizenButton = findViewById(R.id.senior_citizen_helpline);
      policeButton.setOnClickListener(v -> makeCall("100"));
      fireButton.setOnClickListener(v -> makeCall("101"));
      ambulanceButton.setOnClickListener(v -> makeCall("102"));
      disasterManagementButton.setOnClickListener(v -> makeCall("108"));
      nationalEmergencyButton.setOnClickListener(v -> makeCall("112"));
      womenHelplineButton.setOnClickListener(v -> makeCall("181"));
      covidHelplineButton.setOnClickListener(v -> makeCall("1075"));
      aidsHelplineButton.setOnClickListener(v -> makeCall("1097"));
      childrenHelplineButton.setOnClickListener(v -> makeCall("1098"));
      touristHelplineButton.setOnClickListener(v -> makeCall("1363"));
      lpgLeakButton.setOnClickListener(v -> makeCall("1906"));
      seniorCitizenButton.setOnClickListener(v -> makeCall("14567"));
   }
   private void makeCall(String number) {
      Intent intent = new Intent(Intent.ACTION_DIAL);
      intent.setData(Uri.parse("tel:" + number));
      startActivity(intent);
   }
}
```

PhraseBookActivity.java

```
package rocks.poopjournal.morse;
import android.os.Bundle;
import android.widget.ImageView;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;
public class PhraseBookActivity extends AppCompatActivity {
   RecyclerView phrasebookRv;
   PhrasebookAdapter phrasebookAdapter;
   DBHelper helper;
   ImageView back;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_phrasebook);
      helper = new DBHelper(getApplicationContext());
      back = findViewById(R.id.back);
      back.setOnClickListener(view -> finish());
      ArrayList<PhrasebookModel> arrayList = helper.getAllPhrases();
      phrasebookAdapter = new PhrasebookAdapter(this, helper);
      phrasebookRv = findViewById(R.id.rv_phrasebook);
      phrasebookRv.setHasFixedSize(true);// same
      RecyclerView.LayoutManager mLayoutManager =
            new LinearLayoutManager(PhraseBookActivity.this);
      phrasebookRv.setLayoutManager(mLayoutManager);
      phrasebookRv.setAdapter(phrasebookAdapter);
      phrasebookAdapter.setPhrasebookList(arrayList);
      ItemTouchHelper itemTouchHelper = new
            ItemTouchHelper(new SwipeToDeleteCallback(phrasebookAdapter));
      itemTouchHelper.attachToRecyclerView(phrasebookRv);
   }}
```

LoopMediaPlayer.java

```
package rocks.poopjournal.morse;
import android.content.Context;
import android.media.MediaPlayer;
import android.util.Log;
public class LoopMediaPlayer {
   public static final String TAG = LoopMediaPlayer.class.getSimpleName();
   private Context mContext = null;
   private int mResId = 0;
   private int mCounter = 1;
   private MediaPlayer mCurrentPlayer = null;
   private MediaPlayer mNextPlayer = null;
   public static LoopMediaPlayer create(Context context, int resId) {
      return new LoopMediaPlayer(context, resId);
   }
   private LoopMediaPlayer(Context context, int resId) {
      mContext = context;
      mResId = resId;
      mCurrentPlayer = MediaPlayer.create(mContext, mResId);
      mCurrentPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
         @Override
         public void onPrepared(MediaPlayer mediaPlayer) {
            mCurrentPlayer.start();
         }
      });
      createNextMediaPlayer();
   }
   private void createNextMediaPlayer() {
      mNextPlayer = MediaPlayer.create(mContext, mResId);
      mCurrentPlayer.setNextMediaPlayer(mNextPlayer);
      mCurrentPlayer.setOnCompletionListener(onCompletionListener);
   }
   private MediaPlayer.OnCompletionListener onCompletionListener = new
MediaPlayer.OnCompletionListener() {
      @Override
      public void onCompletion(MediaPlayer mediaPlayer) {
         mediaPlayer.release();
         mCurrentPlayer = mNextPlayer;
         createNextMediaPlayer();
         Log.d(TAG, String.format("Loop #%d", ++mCounter));
      }
   };
   public void stopPlayers(){
      if (mCurrentPlayer.isPlaying()){
         mCurrentPlayer.stop();
      }}}
```

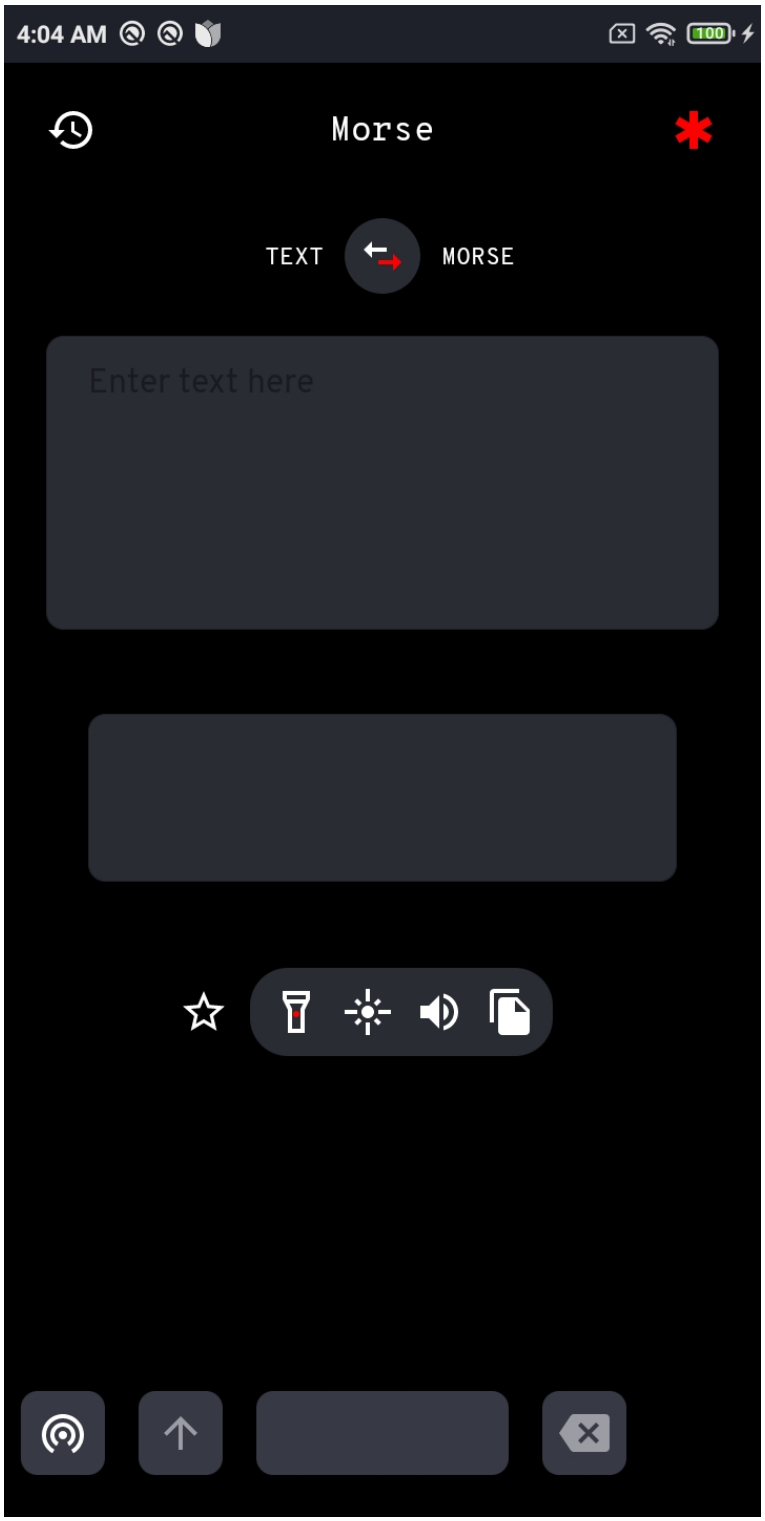
DBHelper.java

```
package rocks.poopjournal.morse;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import java.util.ArrayList;
public class DBHelper extends SQLiteOpenHelper {
   private static final String DATABASE_NAME = "morse";
   private static final int DATABASE_VERSION = 1;
   private static final String TABLE_PHRASEBOOK = "phrasebook";
   private static final String KEY_ID = "id";
   private static final String TAG = "DBhandler";
   public DBHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }
   @Override
   public void onCreate(SQLiteDatabase db) {
      String CREATE_NOTES_PHRASEBOOK = "CREATE TABLE IF NOT EXISTS "
            + TABLE_PHRASEBOOK + "("
            + KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,"
            + "sentence" + " TEXT,"
            + "morse" + " TEXT);";
      db.execSQL(CREATE_NOTES_PHRASEBOOK);
   }
   @Override
   public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1){}
   public void addPhrase(String text, String morse) {
      for (PhrasebookModel model : getAllPhrases()) {
         if (model.text.trim().equals(text.trim())) {
            deleteNote(model.id);
            Log.d("debug_star", "deleted note, now returning");
            return;
         }
      }
      Log.d("debug_star", "adding note, now returning");
      SQLiteDatabase db = this.getWritableDatabase();
      ContentValues values = new ContentValues();
      values.put("sentence", text);
      values.put("morse", morse);
      db.insert(TABLE_PHRASEBOOK, null, values);
      db.close();
   }
```

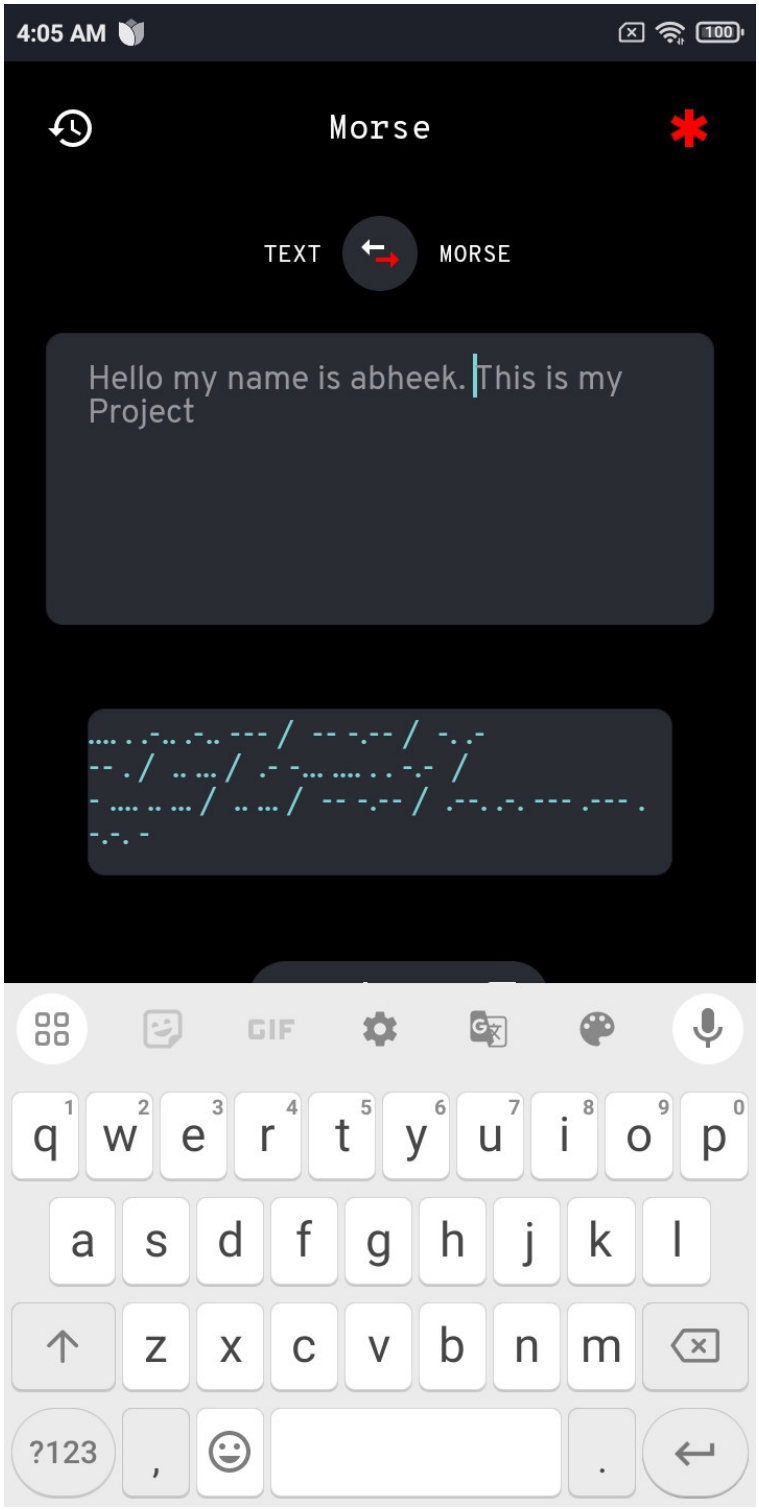
```
    public void deleteNote(int id) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_PHRASEBOOK, "id= " + id, null);
        db.close();
    }
    public ArrayList<PhrasebookModel> getAllPhrases() {
        String selectQuery = "SELECT * FROM " + TABLE_PHRASEBOOK;
        ArrayList<PhrasebookModel> mList = new ArrayList<>();
        try {
            SQLiteDatabase db = this.getWritableDatabase();
            Cursor cursor = db.rawQuery(selectQuery, null);
            while (cursor.moveToNext()) {
                mList.add(new PhrasebookModel(cursor.getInt(0), cursor.getString(1),
cursor.getString(2)));
            }
            db.close();
            return mList;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Output

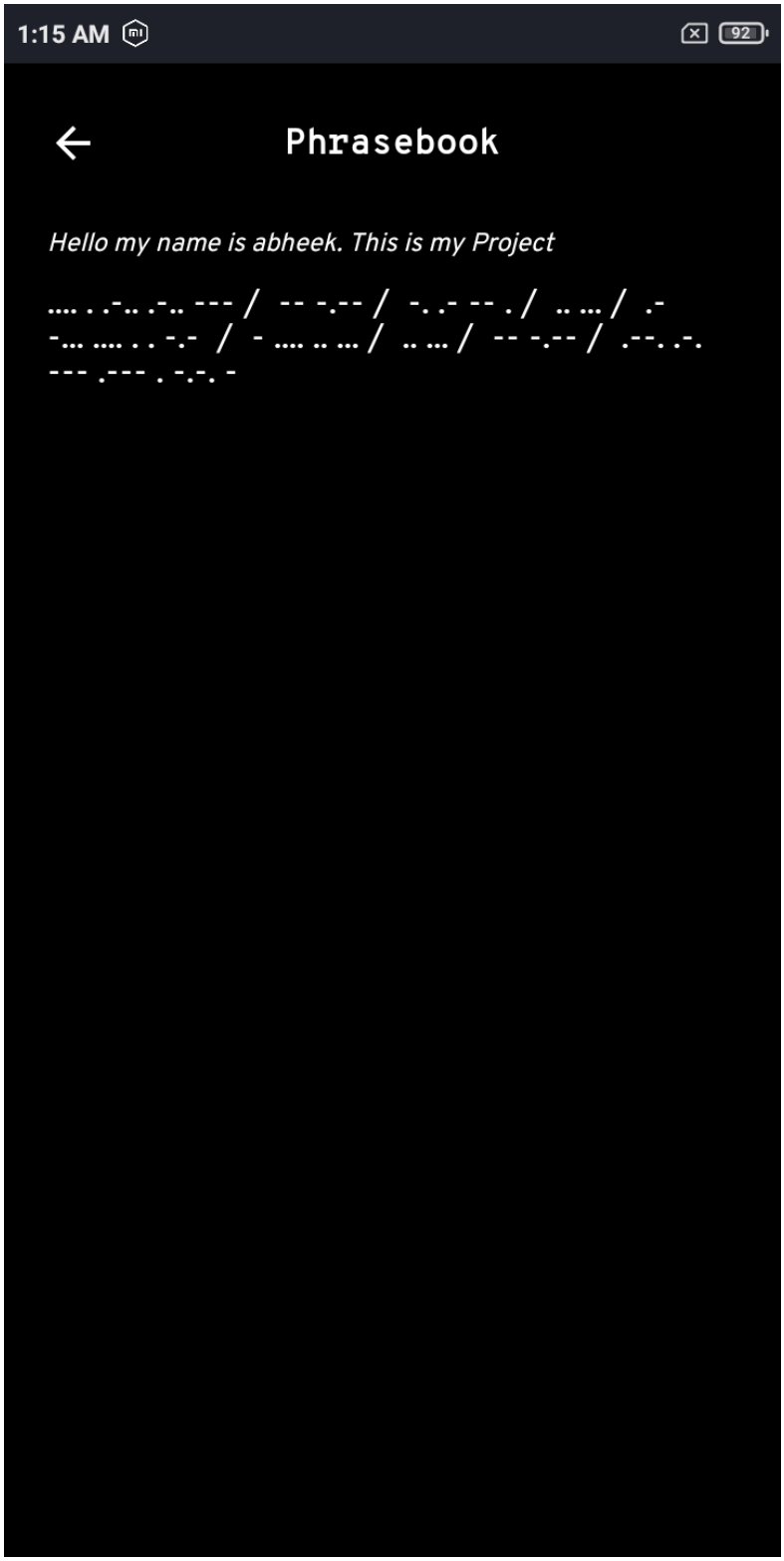
Home Screen



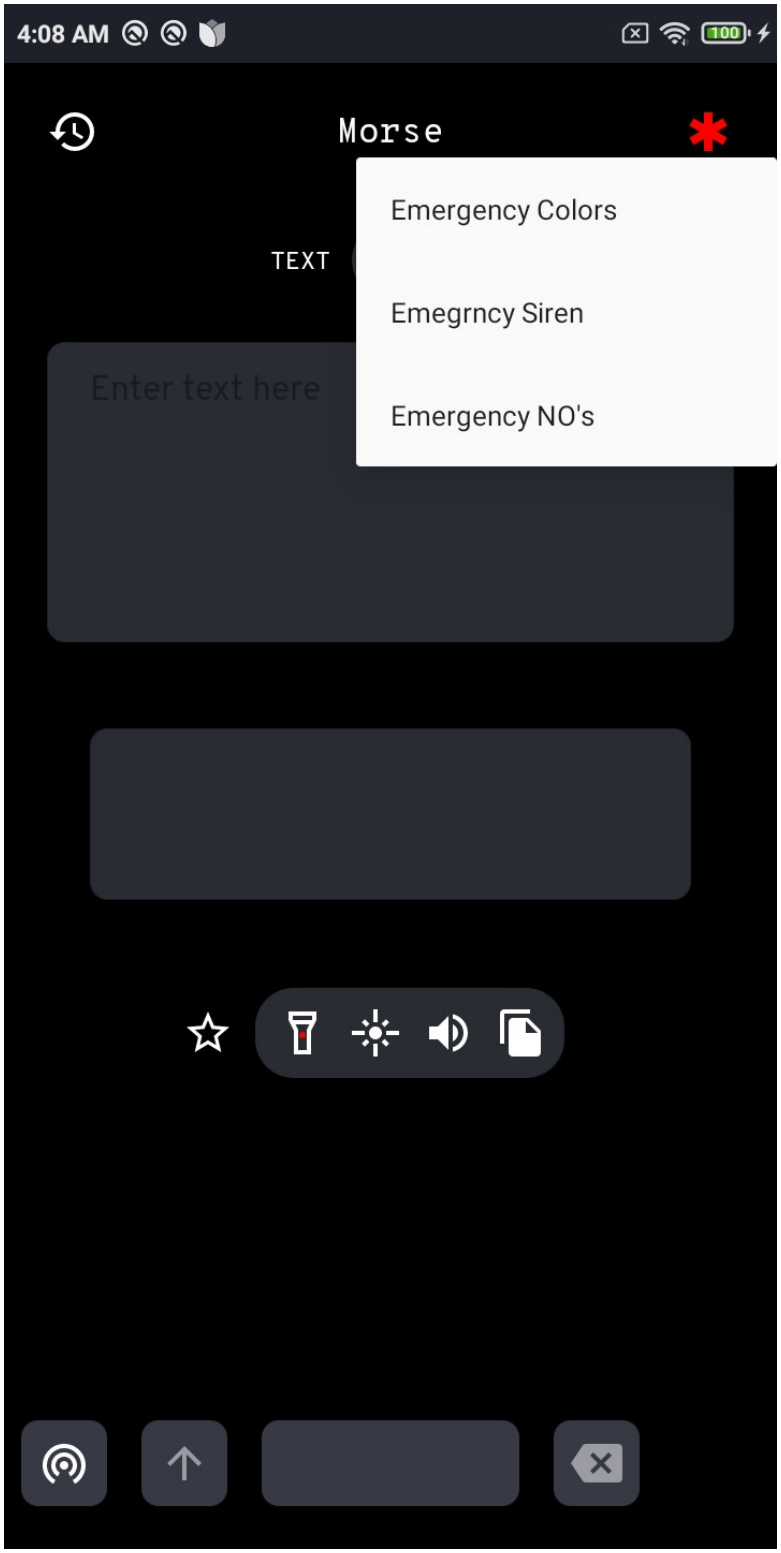
Keyboard In Use



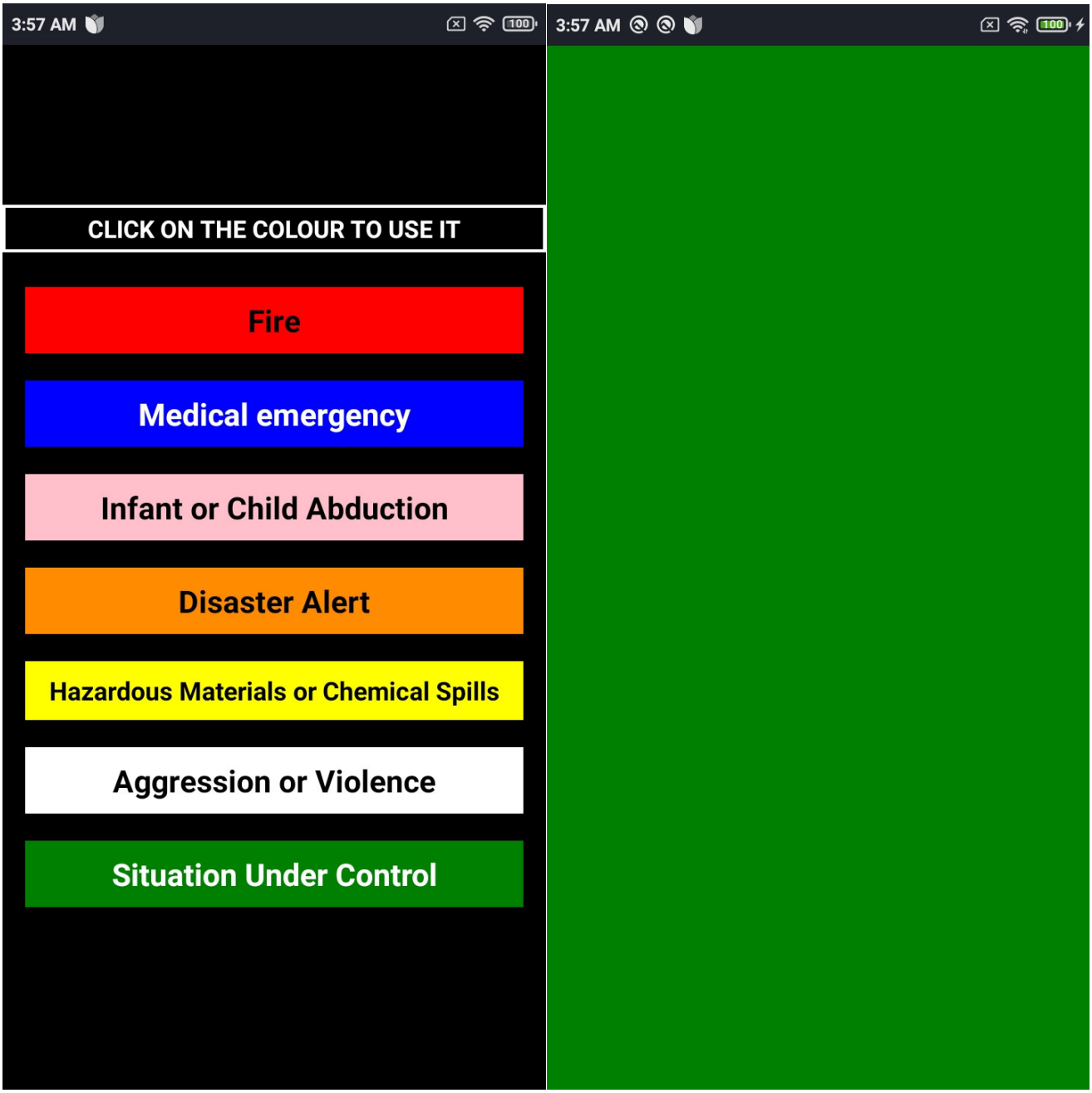
Phrasebook


1:15 AM
92
Phrasebook
Hello my name is abheek. This is my Project
.... . .-.. .-.. --- / -- -.-- / -. .- -- . / .. ... / .-
-... .... . . -.- / - .... / .. ... / -- -.-- / .--. .-.
--- .--- . -.-. -

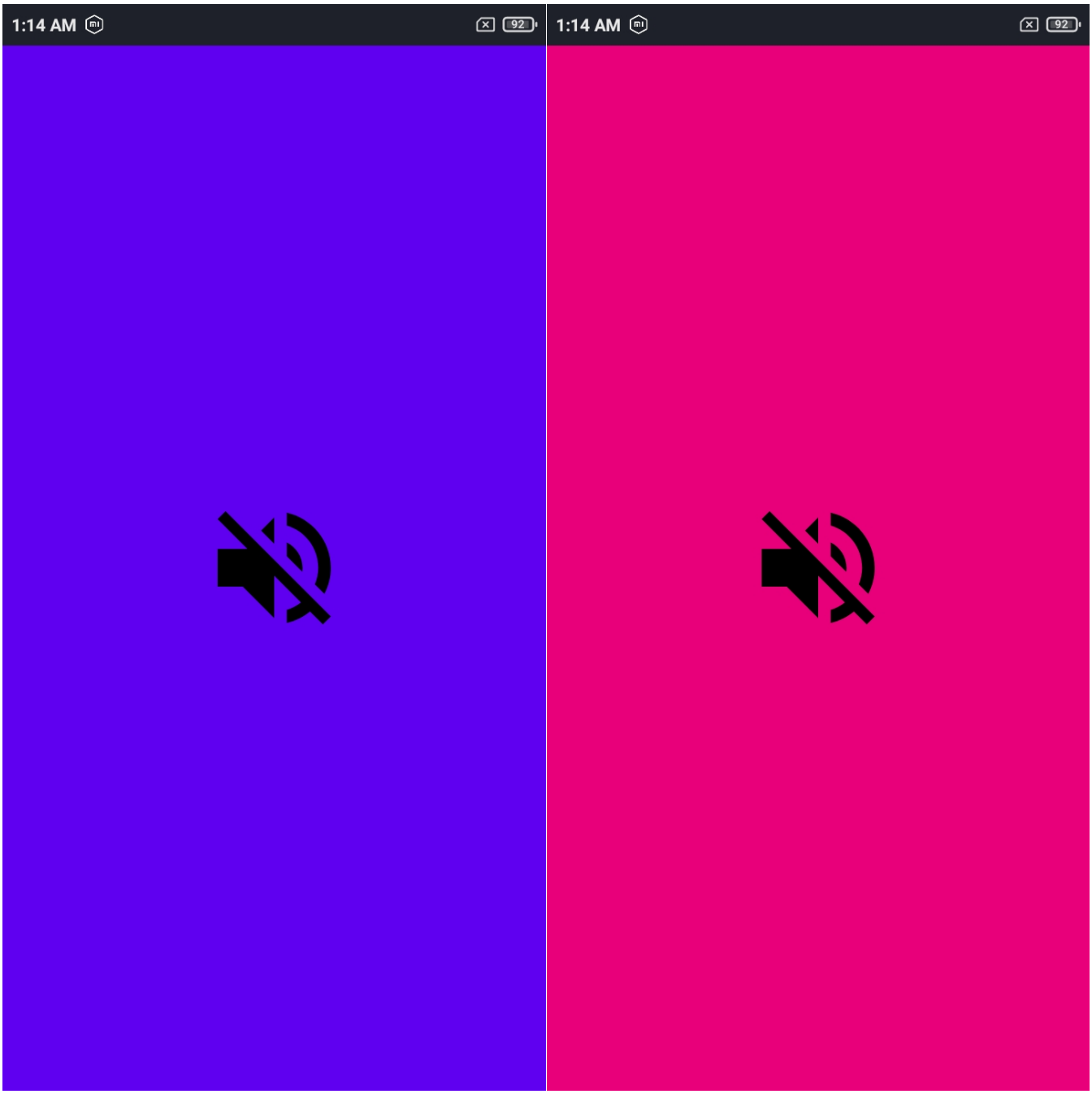
Emergency Menu



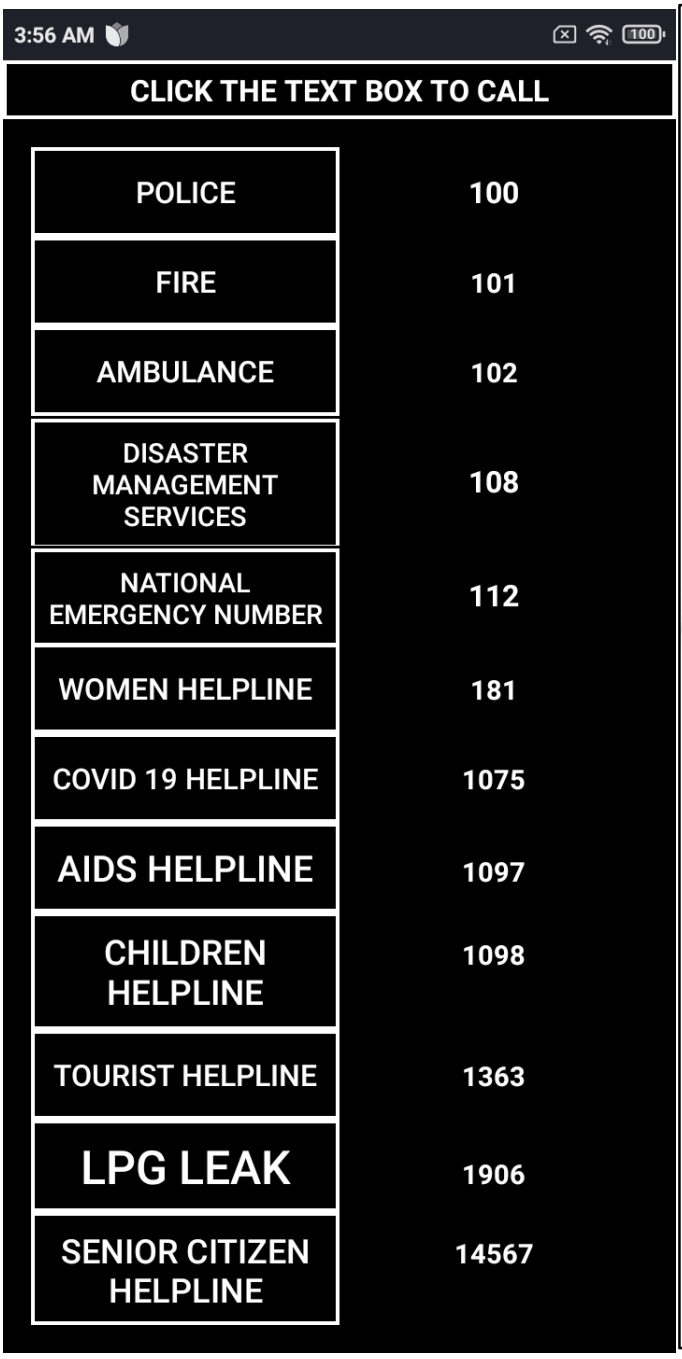
Emergency Colours

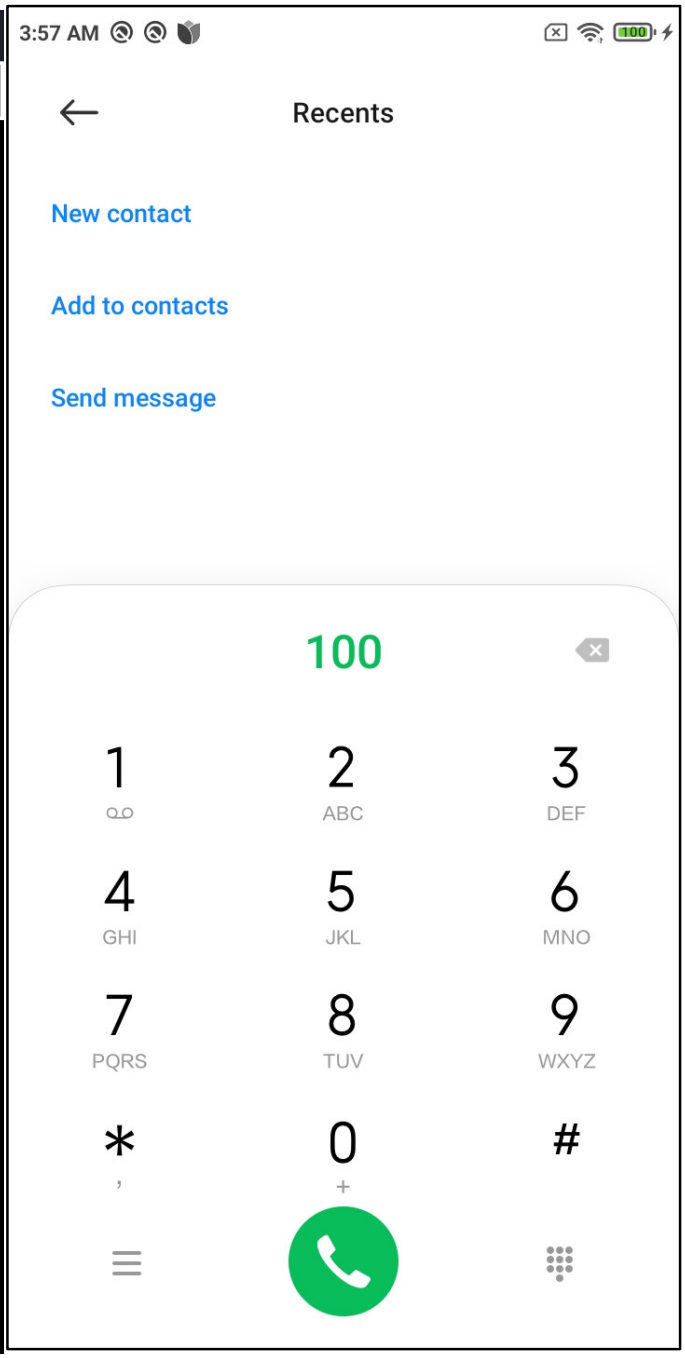


Emergency Siren

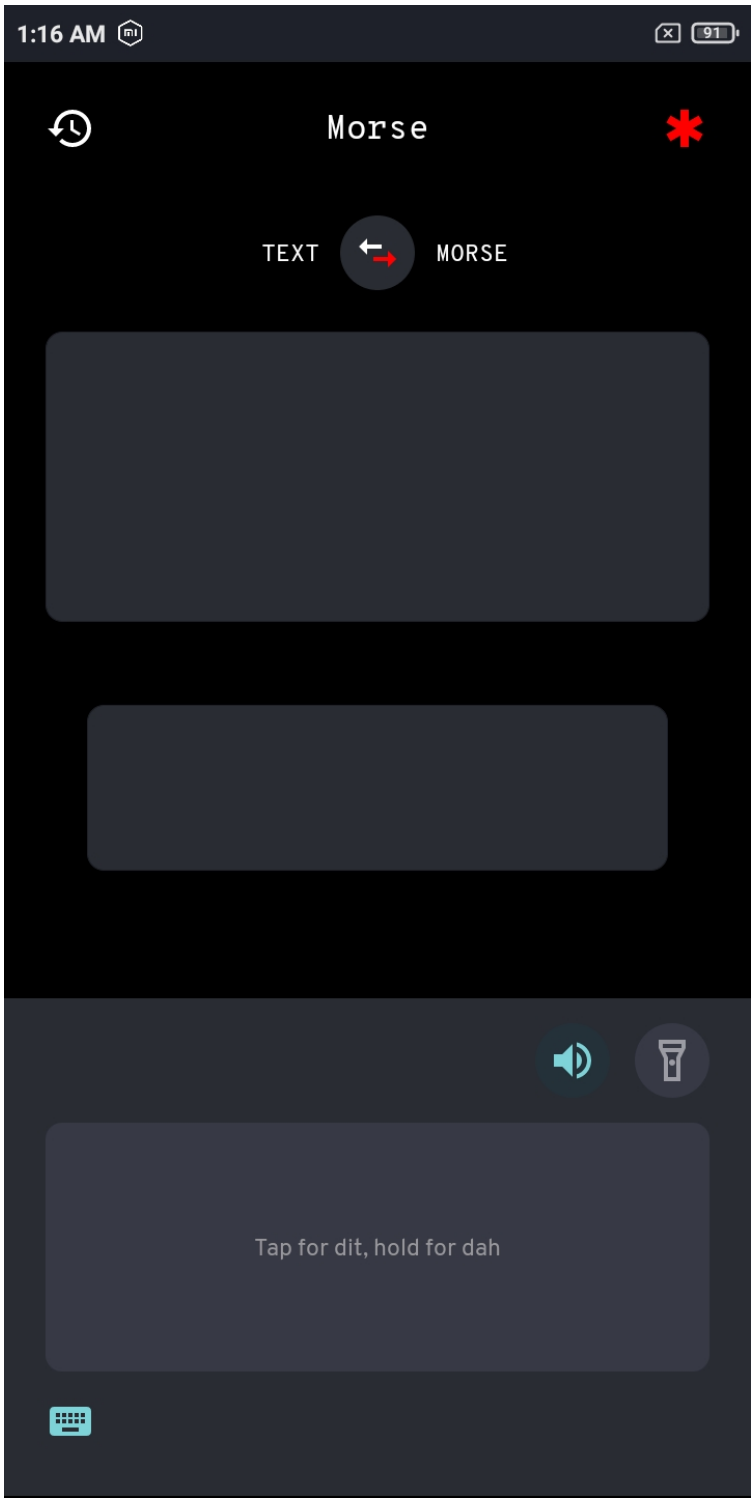


Emergency Numbers (Speed Dial)

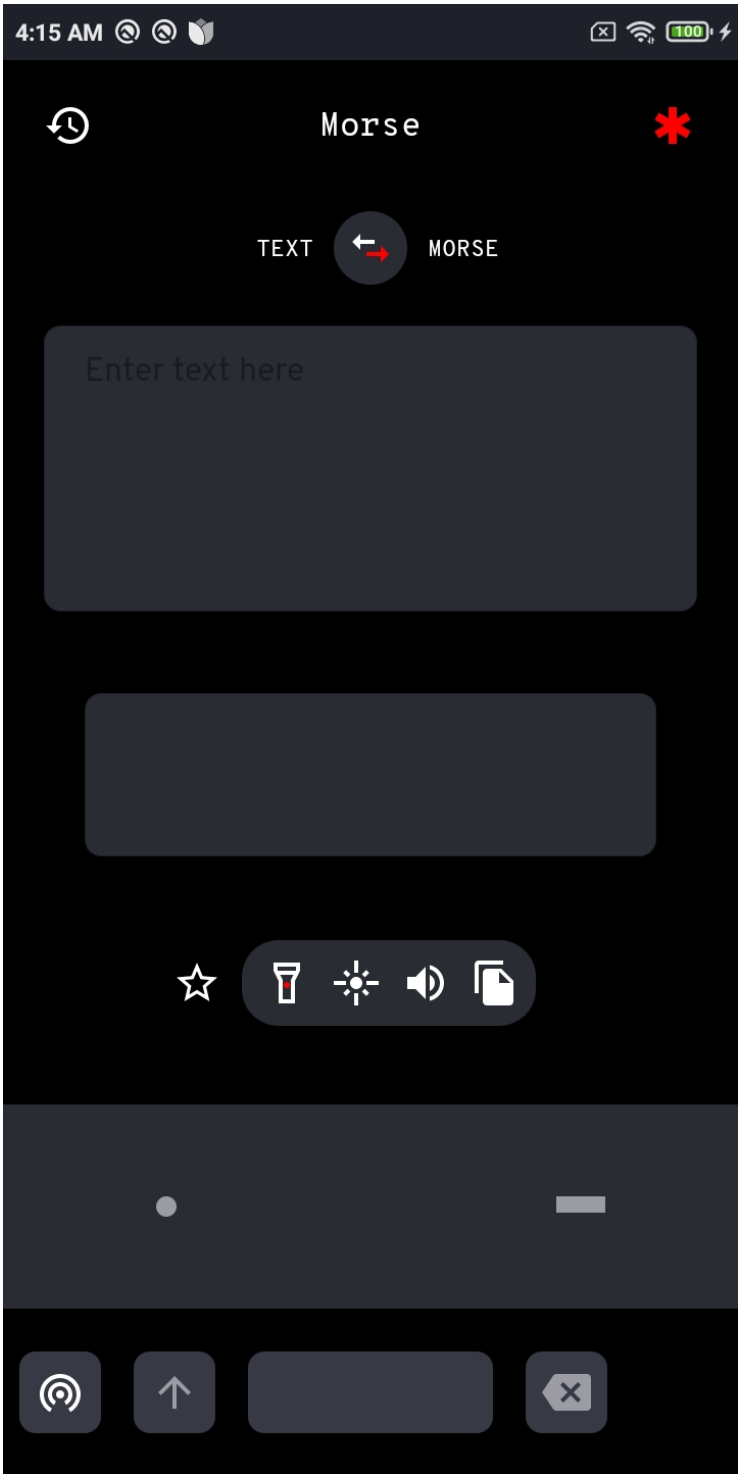




Morse Touchpad



Morse Keypad



References

- **ChatGPT**
  - ChatGPT is an AI language model that assists with generating text, answering questions, and solving problems across various topics.
  - Available at: *https://chatgpt.com/*

- **YouTube**
  - YouTube is a vast platform where users can upload, share, and explore videos on nearly any topic, from entertainment to education and beyond.
  - Available at: *https://www.youtube.com/*

- **GitHub**
  - GitHub is a platform for developers to collaborate, share, and manage code through version control and open-source contributions.
  - Available at: *https://github.com/Crazy-Marvin/Morse*

- **F-Droid**
  - F-Droid is an open-source app store for Android devices, offering free and privacy-friendly applications.
  - Available at*: https://f-droid.org/*

- **Figma**
  - Figma is a cloud-based design tool for creating and collaborating on user interface (UI) and user experience (UX) designs in real time.
  - Available at: *https://www.figma.com/*

- **Android Studio**
  - Android Studio is Google's official integrated development environment (IDE) for building Android apps, offering tools for coding, debugging, and testing.
  - Available at: *https://developer.android.com/studio*