PROGRAM:

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.multilangstttts">

    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Activity_main.xml

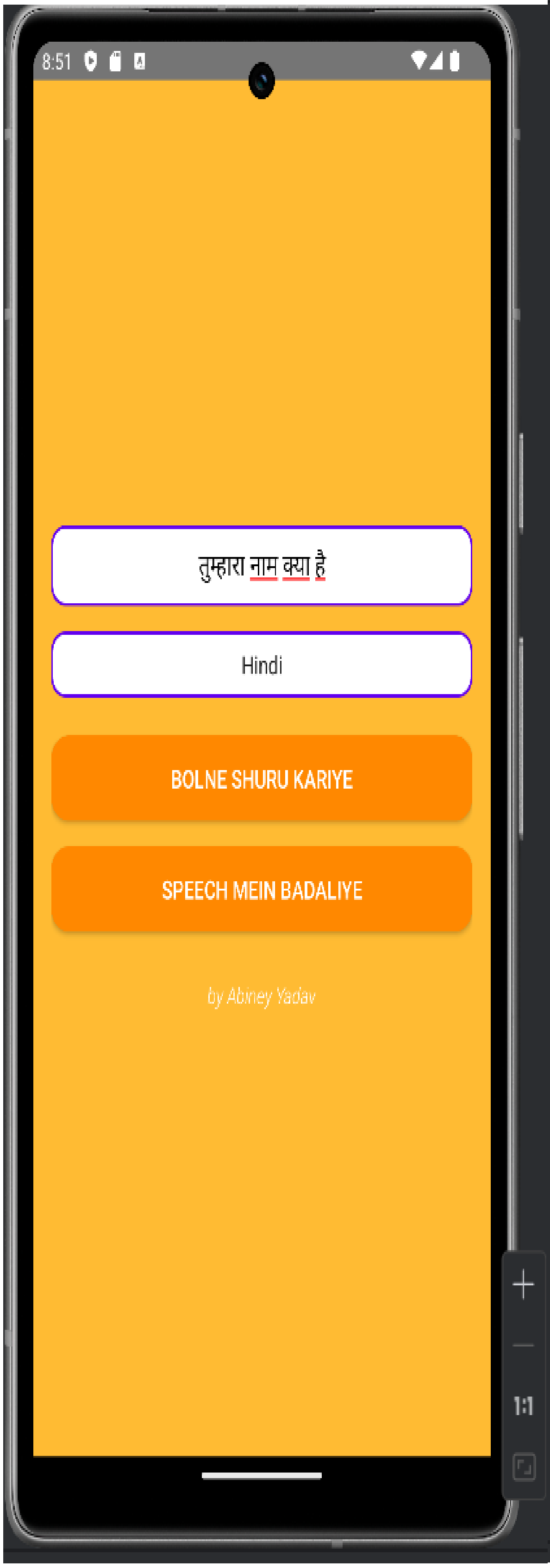
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:gravity="center"
    android:background="@drawable/rounded_edittext">

    <!-- Input Text Field with Stylish Hint -->
    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Your Text Will Appear Here"
        android:gravity="center"
        android:textSize="18sp"
        android:textColor="@android:color/black"
        android:padding="12dp"
        android:background="@drawable/rounded_edittext" />


    <Spinner
        android:id="@+id/languageSpinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:padding="10dp"
        android:entries="@array/languages"
        android:background="@drawable/rounded_spinner"
        android:textAlignment="center" />

    <!-- Speak Button with Modern Design -->
    <Button
        android:id="@+id/btnSpeak"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Bolne shuru kariye"
        android:layout_marginTop="24dp"
        android:textColor="@android:color/white"
        android:textSize="16sp"
        android:background="@drawable/gradient_background"
        android:padding="16dp" />

    <!-- Convert Text-to-Speech Button with Modern Design -->
    <Button
        android:id="@+id/btnConvert"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Speech mein badaliye"
        android:layout_marginTop="16dp"
        android:textColor="@android:color/white"
        android:textSize="16sp"
        android:background="@drawable/rounded_button"
        android:padding="16dp" />

    <!-- Signature with Beautiful Styling -->
    <TextView
        android:id="@+id/signature"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="by Abiney Yadav"
        android:textSize="24sp"
        android:layout_gravity="center"
        android:layout_marginTop="30dp"
        android:textColor="@color/black"
        android:textStyle="italic"
        android:fontFamily="sans-serif-light"/>

</LinearLayout>
```

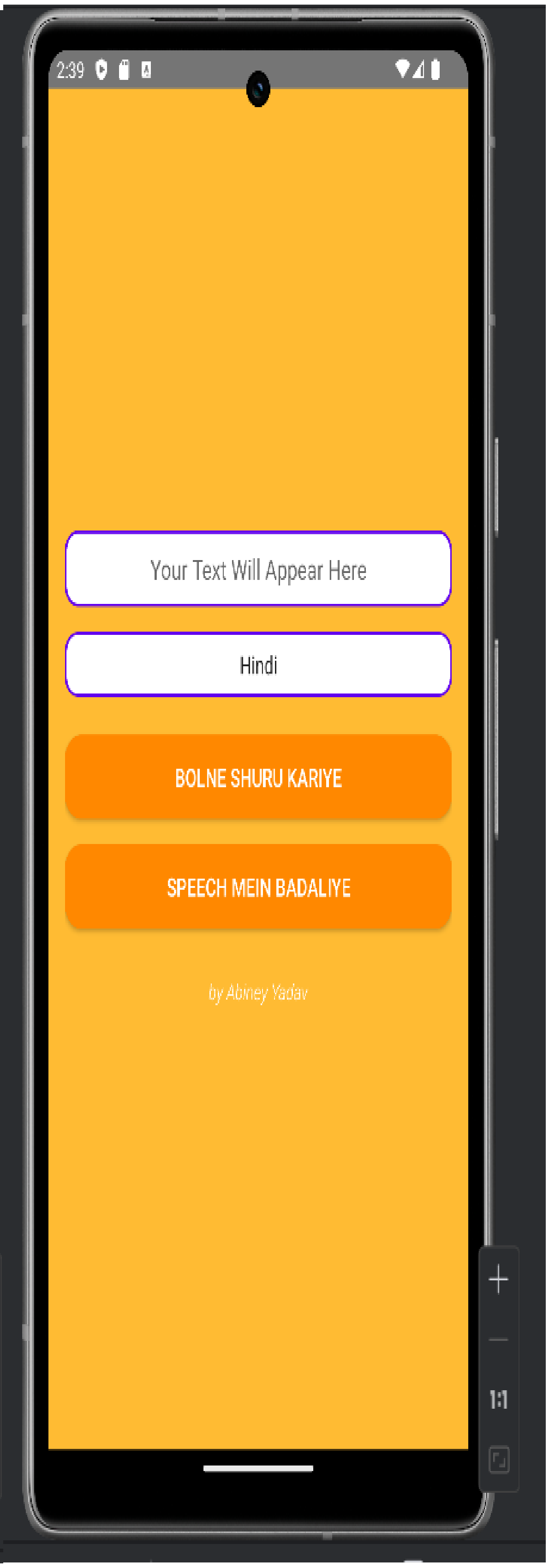
MainActivity.java

```
package com.example.multilangstttts;
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import org.tensorflow.lite.Interpreter;

import java.io.FileInputStream;
import java.io.IOException;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;
import java.util.ArrayList;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {
    private static final int REQUEST_CODE_PERMISSION = 1;
    private SpeechRecognizer speechRecognizer;
    private TextToSpeech textToSpeech;
    private EditText editText;
    private Spinner languageSpinner;
    private Button btnSpeak, btnConvert;

    private Interpreter sttInterpreter;
    private Interpreter ttsInterpreter;

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

        // Initialize UI components
        editText = findViewById(R.id.editText);
        languageSpinner = findViewById(R.id.languageSpinner);
        btnSpeak = findViewById(R.id.btnSpeak);
        btnConvert = findViewById(R.id.btnConvert);

        // Initialize SpeechRecognizer
        speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);

        // Initialize TextToSpeech
        textToSpeech = new TextToSpeech(this, status -> {
            if (status == TextToSpeech.SUCCESS) {
                textToSpeech.setLanguage(Locale.ENGLISH); // Set default language to English
            }
        });

        try {
            sttInterpreter = new Interpreter(loadModelFile("stt_model.tflite"));
            ttsInterpreter = new Interpreter(loadModelFile("tts_model.tflite"));
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Handle runtime permissions
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.RECORD_AUDIO},
REQUEST_CODE_PERMISSION);
        }

        // Set up SpeechRecognizer
        speechRecognizer.setRecognitionListener(new RecognitionListener() {
            @Override
            public void onReadyForSpeech(Bundle params) {
```

```
            Toast.makeText(MainActivity.this, "Listening...", Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onBeginningOfSpeech() {}

        @Override
        public void onRmsChanged(float rmsdB) {}

        @Override
        public void onBufferReceived(byte[] buffer) {}

        @Override
        public void onEndOfSpeech() {}

        @Override
        public void onError(int error) {
            Toast.makeText(MainActivity.this, "Error: " + error, Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onResults(Bundle results) {
            ArrayList<String> matches =
results.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
            if (matches != null) {
                editText.setText(matches.get(0));
            }
        }

        @Override
        public void onPartialResults(Bundle partialResults) {}

        @Override
        public void onEvent(int eventType, Bundle params) {}
    });

    // Start STT on button click
    btnSpeak.setOnClickListener(v -> startSpeechRecognition());

    // Convert text to speech on button click
    btnConvert.setOnClickListener(v -> convertTextToSpeech());
  }

  // Handle permission result
  @Override
  public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if (requestCode == REQUEST_CODE_PERMISSION) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "Permission Granted", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Permission Denied", Toast.LENGTH_SHORT).show();
        }
    }
}

// Start speech recognition based on the selected language
private void startSpeechRecognition() {
    String selectedLanguage = getSelectedLanguage();
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, selectedLanguage);
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Speak now...");
    speechRecognizer.startListening(intent);
}

// Convert the text in EditText to speech
private void convertTextToSpeech() {
    String text = editText.getText().toString();
    if (text.isEmpty()) {
        Toast.makeText(this, "Please enter some text", Toast.LENGTH_SHORT).show();
        return;
    }

    String selectedLanguage = getSelectedLanguage();
    Locale locale = new Locale(selectedLanguage);
    textToSpeech.setLanguage(locale);

    textToSpeech.speak(text, TextToSpeech.QUEUE_FLUSH, null, null);
}

// Get selected language code from the spinner
private String getSelectedLanguage() {
    String language = languageSpinner.getSelectedItem().toString();
    switch (language) {
        case "Tamil":
            return "ta-IN";  // Sri Lankan Tamil
        default:
            return "EN-IN";  // Default to Hindi if not found
    }
}

// Load the TensorFlow Lite model file
private MappedByteBuffer loadModelFile(String modelPath) throws IOException {
    FileInputStream fileInputStream = new FileInputStream(modelPath);
    FileChannel fileChannel = fileInputStream.getChannel();
    long startOffset = 0;
    long declaredLength = fileChannel.size();
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength);
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if (speechRecognizer != null) {
        speechRecognizer.destroy();
    }
    if (textToSpeech != null) {
        textToSpeech.shutdown();
    }
    if (sttInterpreter != null) {
        sttInterpreter.close();
    }
    if (ttsInterpreter != null) {
        ttsInterpreter.close();
    }
}
}

}
}
```

OUTPUT:


8:51
तुम्हारा नाम क्या है
Hindi
BOLNE SHURU KARIYE
SPEECH MEIN BADALIYE
by Abiney Yadav
1:1


2:39
Your Text Will Appear Here
Hindi
BOLNE SHURU KARIYE
SPEECH MEIN BADALIYE
by Abiney Yadav
1:1

TRAINING MODEL:

```
import os
import numpy as np
import tensorflow as tf
import tensorflow_io as tfio
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, models


def load_audio(file_path):
    audio = tfio.audio.AudioIOTensor(file_path)
    audio = tf.squeeze(audio, axis=-1) # Remove channel dimension
    return audio.numpy()

def preprocess_audio(audio):
    audio = audio.astype(np.float32)  # Convert to float32
    audio = np.pad(audio, (0, max_length - len(audio)), 'constant') # Pad to max_length
    return audio

def load_dataset(data_dir):
    X, y = [], []
    labels = os.listdir(data_dir)

    for label in labels:
        label_dir = os.path.join(data_dir, label)
        if os.path.isdir(label_dir):
            for file in os.listdir(label_dir):
                if file.endswith('.wav'):
                    audio = load_audio(os.path.join(label_dir, file))
                    X.append(preprocess_audio(audio))
                    y.append(labels.index(label))  # Use index as label

    return np.array(X), np.array(y)


def create_model(input_shape, num_classes):
    model = models.Sequential([
        layers.Input(shape=input_shape),
        layers.Conv1D(64, 3, activation='relu'),
        layers.MaxPooling1D(2),
        layers.Conv1D(128, 3, activation='relu'),
        layers.MaxPooling1D(2),
        layers.Flatten(),
        layers.Dense(64, activation='relu'),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model

data_dir = 'path/to/your/dataset' # Update with your dataset path
max_length = 16000 # Set to the maximum length of audio (in samples)
X, y = load_dataset(data_dir)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

num_classes = len(set(y)) # Number of unique labels
model = create_model((max_length,), num_classes)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, batch_size=32)

converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

with open('speech_to_text_model.tflite', 'wb') as f:
    f.write(tflite_model)

interpreter = tf.lite.Interpreter(model_path='speech_to_text_model.tflite')
interpreter.allocate_tensors()

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

def infer(audio_file):
    audio = load_audio(audio_file)
    input_data = preprocess_audio(audio)
    input_data = np.expand_dims(input_data, axis=0)  # Add batch dimension

    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()

    output_data = interpreter.get_tensor(output_details[0]['index'])
    return np.argmax(output_data)  # Return the predicted class
```