```
package com.catdog2025.activity;

import android.Manifest;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.catdog2025.R;

import java.io.File;
import java.io.IOException;

/**
 * 录音播放Activity
 * 录制用户人声，播放对应的动物叫声
 */
public class RecordPlayActivity extends Activity {
    private static final String TAG = "RecordPlayActivity";
    private static final int PERMISSION_REQUEST_CODE = 1000;
    private static final String AUDIO_FILE_NAME = "user_voice.3gp"
; // 固定文件名

    // UI组件
    private ImageView mPetImageView;
```

```
39      private TextView mPetNameTextView;
40      private TextView mRecordTimeTextView;
41      private Button mRecordButton;
42      private Button mPlayButton;
43      private Button mBackButton;
44
45      // 宠物类型
46      private String mPetType; // "cat" 或 "dog"
47
48      // 录音相关
49      private MediaRecorder mMediaRecorder;
50      private MediaPlayer mMediaPlayer;
51      private String mAudioFilePath;
52      private boolean mIsRecording = false;
53      private boolean mIsPlaying = false;
54
55      // 计时相关
56      private Handler mTimeHandler = new Handler(Looper.
    getMainLooper());
57      private Runnable mTimeRunnable;
58      private int mRecordSeconds = 0; // 录音时长（秒）
59
60      @Override
61      protected void onCreate(@Nullable Bundle savedInstanceState) {
62          super.onCreate(savedInstanceState);
63          setContentView(R.layout.activity_record_play);
64
65          // 获取宠物类型
66          Intent intent = getIntent();
67          mPetType = intent.getStringExtra("pet_type");
68          if (mPetType == null) {
69              mPetType = "cat"; // 默认为cat
70          }
71
72          Log.d(TAG, "onCreate: 录音播放界面启动，宠物类型: " +
    mPetType);
73
74          // 初始化视图
75          initViews();
```

```
76
77          // 检查录音权限
78          checkPermissions();
79
80          // 初始化音频文件路径
81          initAudioPath();
82      }
83
84      /**
85       * 初始化视图组件
86       */
87      private void initViews() {
88          mPetImageView = findViewById(R.id.iv_pet);
89          mPetNameTextView = findViewById(R.id.tv_pet_name);
90          mRecordTimeTextView = findViewById(R.id.tv_record_time);
91          mRecordButton = findViewById(R.id.btn_record);
92          mPlayButton = findViewById(R.id.btn_play);
93          mBackButton = findViewById(R.id.btn_back);
94
95          // 根据宠物类型设置UI
96          if ("cat".equals(mPetType)) {
97              mPetImageView.setImageResource(R.drawable.cat);
98              mPetNameTextView.setText("□ 猫咪沟通器");
99          } else {
100             mPetImageView.setImageResource(R.drawable.dog);
101             mPetNameTextView.setText("□ 狗狗沟通器");
102         }
103
104         // 设置按钮点击事件
105         mRecordButton.setOnClickListener(new View.OnClickListener
        () {
106             @Override
107             public void onClick(View v) {
108                 toggleRecording();
109             }
110         });
111
112         mPlayButton.setOnClickListener(new View.OnClickListener() {
113             @Override
```

```
114             public void onClick(View v) {
115                 togglePlaying();
116             }
117         });
118 
119         mBackButton.setOnClickListener(new View.OnClickListener() {
120             @Override
121             public void onClick(View v) {
122                 finish();
123             }
124         });
125 
126         // 初始状态
127         updateUI();
128 
129         Log.d(TAG, "initViews: 界面初始化完成");
130     }
131 
132     /**
133      * 检查录音权限
134      */
135     private void checkPermissions() {
136         if (ContextCompat.checkSelfPermission(this, Manifest.
    permission.RECORD_AUDIO)
137                 != PackageManager.PERMISSION_GRANTED) {
138             ActivityCompat.requestPermissions(this,
139                     new String[]{Manifest.permission.RECORD_AUDIO},
140                     PERMISSION_REQUEST_CODE);
141         }
142     }
143 
144     @Override
145     public void onRequestPermissionsResult(int requestCode, @
    NonNull String[] permissions,
146                                            @NonNull int[] grantResults) {
147         super.onRequestPermissionsResult(requestCode, permissions
    , grantResults);
148         if (requestCode == PERMISSION_REQUEST_CODE) {
149             if (grantResults.length > 0 && grantResults[0] ==
```

```
149 PackageManager.PERMISSION_GRANTED) {
150             Toast.makeText(this, "录音权限已获取", Toast.
    LENGTH_SHORT).show();
151         } else {
152             Toast.makeText(this, "需要录音权限才能使用录音功能",
    Toast.LENGTH_LONG).show();
153         }
154       }
155   }
156
157   /**
158    * 初始化音频文件路径（固定文件名）
159    */
160   private void initAudioPath() {
161       File cacheDir = getCacheDir();
162       mAudioFilePath = cacheDir.getAbsolutePath() + "/" +
    AUDIO_FILE_NAME;
163       Log.d(TAG, "录音文件路径: " + mAudioFilePath);
164   }
165
166   /**
167    * 切换录音状态
168    */
169   private void toggleRecording() {
170       if (mIsRecording) {
171           stopRecording();
172       } else {
173           startRecording();
174       }
175   }
176
177   /**
178    * 开始录音
179    */
180   private void startRecording() {
181       // 检查权限
182       if (ContextCompat.checkSelfPermission(this, Manifest.
    permission.RECORD_AUDIO)
183               != PackageManager.PERMISSION_GRANTED) {
```

```
184             Toast.makeText(this, "请先授予录音权限", Toast.
    LENGTH_SHORT).show();
185             return;
186         }
187 
188         try {
189             // 删除旧的录音文件（固定文件名，覆盖保存）
190             File audioFile = new File(mAudioFilePath);
191             if (audioFile.exists()) {
192                 audioFile.delete();
193                 Log.d(TAG, "删除旧录音文件");
194             }
195 
196             // 配置MediaRecorder
197             mMediaRecorder = new MediaRecorder();
198             mMediaRecorder.setAudioSource(MediaRecorder.
    AudioSource.MIC);
199             mMediaRecorder.setOutputFormat(MediaRecorder.
    OutputFormat.THREE_GPP);
200             mMediaRecorder.setOutputFile(mAudioFilePath);
201             mMediaRecorder.setAudioEncoder(MediaRecorder.
    AudioEncoder.AMR_NB);
202 
203             mMediaRecorder.prepare();
204             mMediaRecorder.start();
205 
206             mIsRecording = true;
207             mRecordSeconds = 0;
208 
209             startTimeCounter();
210             updateUI();
211 
212             Log.d(TAG, "开始录音");
213             //Toast.makeText(this, "开始录制你的声音，准备与宠物沟通
    ...", Toast.LENGTH_SHORT).show();
214 
215         } catch (IOException e) {
216             Log.e(TAG, "录音失败: " + e.getMessage());
217             Toast.makeText(this, "录音失败: " + e.getMessage(), Toast.
```

```
217  LENGTH_LONG).show();
218          }
219      }
220
221      /**
222       * 停止录音
223       */
224      private void stopRecording() {
225          if (mMediaRecorder != null) {
226              try {
227                  mMediaRecorder.stop();
228                  mMediaRecorder.release();
229                  mMediaRecorder = null;
230
231                  mIsRecording = false;
232                  stopTimeCounter();
233                  updateUI();
234
235                  Log.d(TAG, "录音结束，时长: " + mRecordSeconds + "秒");
236                  //Toast.makeText(this, "录音完成(" + mRecordSeconds + "
       秒)，现在可以播放给宠物听了！", Toast.LENGTH_LONG).show();
237
238              } catch (Exception e) {
239                  Log.e(TAG, "停止录音失败: " + e.getMessage());
240                  mIsRecording = false;
241                  updateUI();
242              }
243          }
244      }
245
246      /**
247       * 切换播放状态
248       */
249      private void togglePlaying() {
250          if (mIsPlaying) {
251              stopPlaying();
252          } else {
253              startPlaying();
254          }
```

```
255     }
256
257     /**
258      * 开始播放动物叫声（根据录音时长播放对应时长的cat.mp3或dog.
        mp3）
259      */
260     private void startPlaying() {
261         // 检查是否有录音记录
262         if (mRecordSeconds <= 0) {
263             Toast.makeText(this, "请先录音", Toast.LENGTH_SHORT).
        show();
264             return;
265         }
266
267         try {
268             // 根据宠物类型选择音频资源
269             int audioResId;
270             String animalName;
271             if ("cat".equals(mPetType)) {
272                 audioResId = R.raw.cat;
273                 animalName = "猫咪";
274             } else {
275                 audioResId = R.raw.dog;
276                 animalName = "狗狗";
277             }
278
279             mMediaPlayer = MediaPlayer.create(this, audioResId);
280             if (mMediaPlayer == null) {
281                 Toast.makeText(this, "音频文件加载失败", Toast.
        LENGTH_SHORT).show();
282                 return;
283             }
284
285             // 设置播放完成监听器
286             mMediaPlayer.setOnCompletionListener(new MediaPlayer.
        OnCompletionListener() {
287                 @Override
288                 public void onCompletion(MediaPlayer mp) {
289                     stopPlaying();
```

```
290             }
291         });
292
293         mMediaPlayer.start();
294         mIsPlaying = true;
295         updateUI();
296
297         Log.d(TAG, "开始播放" + animalName + "叫声，原录音时长
    : " + mRecordSeconds + "秒");
298         //Toast.makeText(this, "正在播放" + animalName + "
    叫声给宠物听...", Toast.LENGTH_SHORT).show();
299
300         // 根据录音时长控制播放时长
301         if (mRecordSeconds > 0) {
302             mTimeHandler.postDelayed(new Runnable() {
303                 @Override
304                 public void run() {
305                     if (mIsPlaying) {
306                         stopPlaying();
307                         Toast.makeText(RecordPlayActivity.this,
308                             "播放完成(" + mRecordSeconds + "秒)，
    观察宠物反应吧！", Toast.LENGTH_LONG).show();
309                     }
310                 }
311             }, mRecordSeconds * 1000); // 按录音时长停止播放
312         }
313
314     } catch (Exception e) {
315         Log.e(TAG, "播放失败: " + e.getMessage());
316         Toast.makeText(this, "播放失败: " + e.getMessage(), Toast.
    LENGTH_LONG).show();
317     }
318 }
319
320 /**
321  * 停止播放
322  */
323 private void stopPlaying() {
324     if (mMediaPlayer != null) {
```

```
325             if (mMediaPlayer.isPlaying()) {
326                 mMediaPlayer.stop();
327             }
328             mMediaPlayer.release();
329             mMediaPlayer = null;
330 
331             mIsPlaying = false;
332             updateUI();
333 
334             Log.d(TAG, "停止播放");
335         }
336     }
337 
338     /**
339      * 开始计时
340      */
341     private void startTimeCounter() {
342         mTimeRunnable = new Runnable() {
343             @Override
344             public void run() {
345                 if (mIsRecording) {
346                     mRecordSeconds++;
347                     mRecordTimeTextView.setText(formatTime(
        mRecordSeconds));
348                     mTimeHandler.postDelayed(this, 1000);
349                 }
350             }
351         };
352         mTimeHandler.postDelayed(mTimeRunnable, 1000);
353     }
354 
355     /**
356      * 停止计时
357      */
358     private void stopTimeCounter() {
359         if (mTimeHandler != null && mTimeRunnable != null) {
360             mTimeHandler.removeCallbacks(mTimeRunnable);
361         }
362     }
```

```
363
364     /**
365      * 格式化时间显示
366      */
367     private String formatTime(int seconds) {
368         int minutes = seconds / 60;
369         int secs = seconds % 60;
370         return String.format("%02d:%02d", minutes, secs);
371     }
372
373     /**
374      * 更新UI状态
375      */
376     private void updateUI() {
377         if (mIsRecording) {
378             mRecordButton.setText("停止录音");
379             mRecordButton.setEnabled(true);
380             mPlayButton.setEnabled(false);
381         } else if (mIsPlaying) {
382             mRecordButton.setEnabled(false);
383             mPlayButton.setText("停止播放");
384             mPlayButton.setEnabled(true);
385         } else {
386             mRecordButton.setText("录制人声");
387             mRecordButton.setEnabled(true);
388
389             // 检查是否有录音记录
390             if (mRecordSeconds > 0) {
391                 String petType = "cat".equals(mPetType) ? "猫咪" : "狗狗";
392                 mPlayButton.setText("播放给" + petType);
393                 mPlayButton.setEnabled(true);
394             } else {
395                 String petType = "cat".equals(mPetType) ? "猫咪" : "狗狗";
396                 mPlayButton.setText("播放给" + petType);
397                 mPlayButton.setEnabled(false);
398             }
399         }
400
401         // 显示录音时间
```

```
402         if (!mIsRecording && mRecordSeconds > 0) {
403             mRecordTimeTextView.setText("录音时长: " + formatTime(
        mRecordSeconds));
404         } else if (!mIsRecording) {
405             mRecordTimeTextView.setText("00:00");
406         }
407     }
408 
409     @Override
410     protected void onDestroy() {
411         super.onDestroy();
412 
413         // 清理资源
414         if (mMediaRecorder != null) {
415             mMediaRecorder.release();
416             mMediaRecorder = null;
417         }
418 
419         if (mMediaPlayer != null) {
420             if (mMediaPlayer.isPlaying()) {
421                 mMediaPlayer.stop();
422             }
423             mMediaPlayer.release();
424             mMediaPlayer = null;
425         }
426 
427         stopTimeCounter();
428 
429         Log.d(TAG, "onDestroy: 录音播放页面销毁");
430     }
431 }
```