

```
1 package com.catdog2025.activity;
2
3 import android.app.Activity;
4 import android.app.AlertDialog;
5 import android.content.DialogInterface;
6 import android.content.Intent;
7 import android.os.Bundle;
8 import android.util.Log;
9 import android.view.View;
10 import android.widget.FrameLayout;
11 import android.widget.ImageView;
12 import android.widget.Toast;
13 import androidx.annotation.Nullable;
14
15 import com.bytedance.sdk.openadsdk.AdSlot;
16 import com.bytedance.sdk.openadsdk.TTAdConstant;
17 import com.bytedance.sdk.openadsdk.TTAdDislike;
18 import com.bytedance.sdk.openadsdk.TTAdNative;
19 import com.bytedance.sdk.openadsdk.TTNativeExpressAd;
20 import com.bytedance.sdk.openadsdk.TTRewardVideoAd;
21 import com.catdog2025.R;
22 import com.catdog2025.config.TTAdManagerHolder;
23 import com.catdog2025.utils.UIUtils;
24
25 import java.util.List;
26
27 /**
28  * Cat和Dog图片展示页面
29  * 显示上方的cat图片和下方的dog图片
30  */
31 public class CatDogActivity extends Activity {
32     private static final String TAG = "CatDogActivity";
33
34     private ImageView mCatImageView;
35     private ImageView mDogImageView;
36
37     // Cat Banner广告相关
38     private FrameLayout mCatBannerContainer;
39     private TTNativeExpressAd mCatBannerAd;
```

```

40     private TAdNative.NativeExpressAdListener mCatBannerListener
41     ;
42     private TTNativeExpressAd.ExpressAdInteractionListener
43     mCatBannerInteractionListener;
44     private TAdDislike.DislikeInteractionCallback
45     mCatDislikeCallback;
46
47     // Dog Banner广告相关
48     private FrameLayout mDogBannerContainer;
49     private TTNativeExpressAd mDogBannerAd;
50     private TAdNative.NativeExpressAdListener
51     mDogBannerListener;
52     private TTNativeExpressAd.ExpressAdInteractionListener
53     mDogBannerInteractionListener;
54     private TAdDislike.DislikeInteractionCallback
55     mDogDislikeCallback;
56
57     // 激励广告相关
58     private TTRewardVideoAd mTTRewardVideoAd;
59     private TAdNative.RewardVideoAdListener
60     mRewardVideoListener;
61     private TTRewardVideoAd.RewardAdInteractionListener
62     mRewardVideoAdInteractionListener;
63     private String mPendingPetType; // 等待跳转的宠物类型
64
65     @Override
66     protected void onCreate(@Nullable Bundle savedInstanceState) {
67         super.onCreate(savedInstanceState);
68         Log.d(TAG, "onCreate: 启动CatDogActivity");
69         setContentView(R.layout.activity_cat_dog);
70
71         // 初始化视图
72         initView();
73
74         Log.d(TAG, "onCreate: CatDogActivity初始化完成");
75
76         // 记录激励广告配置信息
77         String rewardId = getResources().getString(R.string.
78         reward_media_id);

```

```
70     Log.d(TAG, "□ 激励广告配置信息:");
71     Log.d(TAG, " 广告位ID: " + rewardId);
72     Log.d(TAG, " SDK初始化状态: " + (TtAdManagerHolder.get() != null ? "已初始化" : "未初始化"));
73     if (TtAdManagerHolder.get() != null) {
74         Log.d(TAG, " SDK版本: " + TtAdManagerHolder.get().getSDKVersion());
75     }
76 }
77
78 /**
79  * 初始化视图组件
80  */
81 private void initView() {
82     mCatImageView = findViewById(R.id.iv_cat);
83     mDogImageView = findViewById(R.id.iv_dog);
84
85     // 初始化Banner广告容器
86     mCatBannerContainer = findViewById(R.id.
cat_banner_container);
87     mDogBannerContainer = findViewById(R.id.
dog_banner_container);
88
89     // 设置图片资源
90     mCatImageView.setImageResource(R.drawable.cat);
91     mDogImageView.setImageResource(R.drawable.dog);
92
93     // 设置图片点击事件
94     mCatImageView.setOnClickListener(new View.OnClickListener
() {
95         @Override
96         public void onClick(View v) {
97             Log.d(TAG, "Cat图片被点击，显示激励广告确认对话框");
98             showRewardAdConfirmDialog("cat");
99         }
100     });
101
102     mDogImageView.setOnClickListener(new View.
OnClickListener() {
```

```
103         @Override
104         public void onClick(View v) {
105             Log.d(TAG, "Dog图片被点击，显示激励广告确认对话框");
106             showRewardAdConfirmDialog("dog");
107         }
108     });
109
110     // 初始化广告监听器
111     initAdListeners();
112
113     // 加载Banner广告
114     loadCatBannerAd();
115     loadDogBannerAd();
116
117     Log.d(TAG, "initViews: 图片设置完成，开始加载Banner广告");
118 }
119
120 @Override
121 protected void onResume() {
122     super.onResume();
123     Log.d(TAG, "onResume: 页面恢复");
124 }
125
126 @Override
127 protected void onPause() {
128     super.onPause();
129     Log.d(TAG, "onPause: 页面暂停");
130 }
131
132 /**
133  * 初始化广告监听器
134  */
135 private void initAdListeners() {
136     // 猫咪Banner广告监听器
137     mCatBannerListener = new TTAdNative.
NativeExpressAdListener() {
138         @Override
139         public void onError(int code, String msg) {
140             Log.d(TAG, "Cat banner load fail: errCode: " + code + ",
```

```
140 errMsg: " + msg);
141     }
142
143     @Override
144     public void onNativeExpressAdLoad(List<
TTNativeExpressAd> list) {
145         if (list != null && list.size() > 0) {
146             Log.d(TAG, "Cat banner load success");
147             mCatBannerAd = list.get(0);
148             showCatBannerAd();
149         } else {
150             Log.d(TAG, "Cat banner load success, but list is null");
151         }
152     }
153 };
154
155     mCatBannerInteractionListener = new TTNativeExpressAd.
ExpressAdInteractionListener() {
156         @Override
157         public void onAdClicked(View view, int type) {
158             Log.d(TAG, "Cat banner clicked");
159         }
160
161         @Override
162         public void onAdShow(View view, int type) {
163             Log.d(TAG, "Cat banner showed");
164         }
165
166         @Override
167         public void onRenderFail(View view, String msg, int code) {
168             Log.d(TAG, "Cat banner render fail: " + msg);
169         }
170
171         @Override
172         public void onRenderSuccess(View view, float width, float
height) {
173             Log.d(TAG, "Cat banner render success");
174         }
175     };
```

```

176
177     mCatDislikeCallback = new TAdDislike.
DislikeInteractionCallback() {
178         @Override
179         public void onShow() {}
180
181         @Override
182         public void onSelected(int position, String value, boolean
enforce) {
183             if (mCatBannerContainer != null) {
184                 mCatBannerContainer.removeAllViews();
185             }
186             Log.d(TAG, "Cat banner closed");
187         }
188
189         @Override
190         public void onCancel() {}
191     };
192
193     // 狗狗Banner广告监听器
194     mDogBannerListener = new TAdNative.
NativeExpressAdListener() {
195         @Override
196         public void onError(int code, String msg) {
197             Log.d(TAG, "Dog banner load fail: errCode: " + code + ",
errMsg: " + msg);
198         }
199
200         @Override
201         public void onNativeExpressAdLoad(List<
TTNativeExpressAd> list) {
202             if (list != null && list.size() > 0) {
203                 Log.d(TAG, "Dog banner load success");
204                 mDogBannerAd = list.get(0);
205                 showDogBannerAd();
206             } else {
207                 Log.d(TAG, "Dog banner load success, but list is null");
208             }
209         }

```

```
210     };
211
212     mDogBannerInteractionListener = new TTNativeExpressAd.
ExpressAdInteractionListener() {
213         @Override
214         public void onAdClicked(View view, int type) {
215             Log.d(TAG, "Dog banner clicked");
216         }
217
218         @Override
219         public void onAdShow(View view, int type) {
220             Log.d(TAG, "Dog banner showed");
221         }
222
223         @Override
224         public void onRenderFail(View view, String msg, int code) {
225             Log.d(TAG, "Dog banner render fail: " + msg);
226         }
227
228         @Override
229         public void onRenderSuccess(View view, float width, float
height) {
230             Log.d(TAG, "Dog banner render success");
231         }
232     };
233
234     mDogDislikeCallback = new TTAdDislike.
DislikeInteractionCallback() {
235         @Override
236         public void onShow() {}
237
238         @Override
239         public void onSelected(int position, String value, boolean
enforce) {
240             if (mDogBannerContainer != null) {
241                 mDogBannerContainer.removeAllViews();
242             }
243             Log.d(TAG, "Dog banner closed");
244         }
    }
```

```
245
246     @Override
247     public void onCancel() {}
248 };
249 }
250
251 /**
252  * 加载猫咪Banner广告
253  */
254 private void loadCatBannerAd() {
255     String catBannerId = getResources().getString(R.string.
cat_banner_media_id);
256     // 创建AdSlot对象
257     AdSlot adSlot = new AdSlot.Builder()
258         .setCodeId(catBannerId) // 使用Cat专用Banner广告位ID
259         .setImageAcceptedSize(UIUtils.dp2px(this, 350f), UIUtils.
dp2px(this, 120f))
260         .build();
261
262     // 创建TAdNative对象
263     TAdNative adNativeLoader = TAdManagerHolder.get().
createAdNative(this);
264
265     // 加载广告
266     if (adNativeLoader != null) {
267         adNativeLoader.loadBannerExpressAd(adSlot,
mCatBannerListener);
268         Log.d(TAG, "开始加载Cat Banner广告，广告位ID: " +
catBannerId);
269     }
270 }
271
272 /**
273  * 加载狗狗Banner广告
274  */
275 private void loadDogBannerAd() {
276     String dogBannerId = getResources().getString(R.string.
dog_banner_media_id);
277     // 创建AdSlot对象
```



```
278     AdSlot adSlot = new AdSlot.Builder()
279         .setCodeId(dogBannerId) // 使用Dog专用Banner广告位ID
280         .setImageAcceptedSize(UIUtils.dp2px(this, 350f), UIUtils.
dp2px(this, 120f))
281         .build();
282
283     // 创建TAdNative对象
284     TAdNative adNativeLoader = TAdManagerHolder.get().
createAdNative(this);
285
286     // 加载广告
287     if (adNativeLoader != null) {
288         adNativeLoader.loadBannerExpressAd(adSlot,
mDogBannerListener);
289         Log.d(TAG, "开始加载Dog Banner广告 · 广告位ID: " +
dogBannerId);
290     }
291 }
292
293 /**
294  * 显示猫咪Banner广告
295  */
296 private void showCatBannerAd() {
297     if (mCatBannerAd != null && mCatBannerContainer != null) {
298         mCatBannerAd.setExpressInteractionListener(
mCatBannerInteractionListener);
299         mCatBannerAd.setDislikeCallback(this, mCatDislikeCallback
);
300
301         View bannerView = mCatBannerAd.getExpressAdView();
302         if (bannerView != null) {
303             mCatBannerContainer.removeAllViews();
304             mCatBannerContainer.addView(bannerView);
305             Log.d(TAG, "Cat Banner广告显示成功");
306         }
307     }
308 }
309
310 /**
```

```

311     * 显示狗狗Banner广告
312     */
313     private void showDogBannerAd() {
314         if (mDogBannerAd != null && mDogBannerContainer != null
315         ) {
316             mDogBannerAd.setExpressInteractionListener(
317             mDogBannerInteractionListener);
318             mDogBannerAd.setDislikeCallback(this,
319             mDogDislikeCallback);
320
321             View bannerView = mDogBannerAd.getExpressAdView();
322             if (bannerView != null) {
323                 mDogBannerContainer.removeAllViews();
324                 mDogBannerContainer.addView(bannerView);
325                 Log.d(TAG, "Dog Banner广告显示成功");
326             }
327         }
328     }
329
330     /**
331     * 显示激励广告确认对话框
332     * @param petType 宠物类型
333     */
334     private void showRewardAdConfirmDialog(String petType) {
335         Log.d(TAG, "□ 显示激励广告确认对话框");
336         Log.d(TAG, "  宠物类型: " + petType);
337
338         mPendingPetType = petType;
339         String petName = "cat".equals(petType) ? "小猫" : "小狗";
340
341         Log.d(TAG, "  宠物名称: " + petName);
342
343         AlertDialog.Builder builder = new AlertDialog.Builder(this);
344         builder.setTitle("□ 解锁" + petName + "沟通器");
345         builder.setMessage("观看一个短视频广告即可解锁" + petName
346         + "沟通功能，与您的宠物开始对话吧！");
347         builder.setICon("cat".equals(petType) ? R.drawable.cat : R.
348         drawable.dog);
349     }

```

```

345     builder.setPositiveButton("观看广告", new DialogInterface.
        OnClickListener() {
346         @Override
347         public void onClick(DialogInterface dialog, int which) {
348             Log.d(TAG, "□ 用户点击'观看广告'按钮");
349             Log.d(TAG, " 即将开始加载激励广告...");
350             loadAndShowRewardAd();
351         }
352     });
353
354     builder.setNegativeButton("稍后再说", new DialogInterface.
        OnClickListener() {
355         @Override
356         public void onClick(DialogInterface dialog, int which) {
357             Log.d(TAG, "□ 用户点击'稍后再说'按钮");
358             Log.d(TAG, " 用户取消观看激励广告");
359             Toast.makeText(CatDogActivity.this, "
需要观看广告才能使用沟通功能哦", Toast.LENGTH_SHORT).show();
360         }
361     });
362
363     builder.setCancelable(false); // 不允许点击外部取消
364     AlertDialog dialog = builder.create();
365     dialog.show();
366
367     Log.d(TAG, "□ 激励广告确认对话框已显示");
368 }
369
370 /**
371  * 显示激励广告重试对话框
372  * @param message 提示信息
373  */
374 private void showRewardAdRetryDialog(String message) {
375     Log.d(TAG, "□ 显示激励广告重试对话框");
376
377     AlertDialog.Builder builder = new AlertDialog.Builder(this);
378     builder.setTitle("广告加载失败");
379     builder.setMessage(message + "\n\n• 重试：重新加载广告\n•
取消：返回主页面");

```

```

380     builder.setCancelable(false);
381
382     // 重试按钮
383     builder.setPositiveButton("重试", new DialogInterface.
OnClickListener() {
384         @Override
385         public void onClick(DialogInterface dialog, int which) {
386             Log.d(TAG, "□ 用户选择重试广告");
387             dialog.dismiss();
388             // 重新加载广告
389             loadAndShowRewardAd();
390         }
391     });
392
393     // 取消按钮 ( 返回主页面 , 保护广告收入 )
394     builder.setNegativeButton("取消", new DialogInterface.
OnClickListener() {
395         @Override
396         public void onClick(DialogInterface dialog, int which) {
397             Log.d(TAG, "□ 用户取消广告 , 返回主页面 ( 保护广告收入
) ");
398             dialog.dismiss();
399             Toast.makeText(CatDogActivity.this, "取消广告 ,
请稍后再试", Toast.LENGTH_SHORT).show();
400             // 清空待处理状态 , 返回主页面
401             mPendingPetType = null;
402         }
403     });
404
405     AlertDialog dialog = builder.create();
406     dialog.show();
407
408     Log.d(TAG, "□ 激励广告重试对话框已显示");
409 }
410
411 /**
412  * 显示激励广告失败选择对话框
413  */
414 private void showRewardAdFailDialog() {

```

```

415     Log.d(TAG, "□ 显示激励广告失败选择对话框");
416
417     String petName = "cat".equals(mPendingPetType) ? "小猫" : "
    小狗";
418
419     AlertDialog.Builder builder = new AlertDialog.Builder(this);
420     builder.setTitle("广告暂时无法加载");
421     builder.setMessage("抱歉 · 当前暂时无法加载广告\n\n
    您可以选择：\n• 重新尝试加载广告\n• 稍后再试（返回主页面）");
422     builder.setCancelable(false);
423
424     // 重试按钮
425     builder.setPositiveButton("重试", new DialogInterface.
    OnClickListener() {
426         @Override
427         public void onClick(DialogInterface dialog, int which) {
428             Log.d(TAG, "□ 用户选择重试广告（无广告可用情况）");
429             dialog.dismiss();
430             loadAndShowRewardAd();
431         }
432     });
433
434     // 稍后再试按钮（保护广告收入）
435     builder.setNegativeButton("稍后再试", new DialogInterface.
    OnClickListener() {
436         @Override
437         public void onClick(DialogInterface dialog, int which) {
438             Log.d(TAG, "□ 用户选择稍后再试（保护广告收入）");
439             dialog.dismiss();
440             //Toast.makeText(CatDogActivity.this, "
    请稍后再试观看广告解锁功能", Toast.LENGTH_SHORT).show();
441             // 清空待处理状态 · 不解锁功能
442             mPendingPetType = null;
443         }
444     });
445
446     AlertDialog dialog = builder.create();
447     dialog.show();
448

```

```
449     Log.d(TAG, "□ 激励广告失败选择对话框已显示");
450 }
451
452 /**
453  * 加载并展示激励广告
454  */
455 private void loadAndShowRewardAd() {
456     Log.d(TAG, "=== 开始激励广告流程 ===");
457
458     // 检查SDK初始化状态
459     if (TtAdManagerHolder.get() == null) {
460         Log.e(TAG, "TtAdManagerHolder为空，SDK未正确初始化");
461         Toast.makeText(this, "广告SDK未初始化，请重启应用", Toast.
LENGTH_LONG).show();
462         return;
463     }
464     Log.d(TAG, "□ TtAdManagerHolder已初始化");
465
466     // 初始化监听器
467     initRewardAdListeners();
468     Log.d(TAG, "□ 激励广告监听器已初始化");
469
470     // 创建AdSlot对象
471     String rewardId = getResources().getString(R.string.
reward_media_id);
472     Log.d(TAG, "□ 获取激励广告位ID: " + rewardId);
473
474     AdSlot adSlot = new AdSlot.Builder()
475         .setCodeId(rewardId)
476         .setOrientation(TtAdConstant.ORIENTATION_VERTICAL)
477         .build();
478     Log.d(TAG, "□ AdSlot对象创建成功，方向: 竖屏");
479
480     // 创建TtAdNative对象
481     TtAdNative adNativeLoader = TtAdManagerHolder.get().
createAdNative(this);
482     if (adNativeLoader == null) {
483         Log.e(TAG, "□ createAdNative返回null，
无法创建广告加载器");

```

```

484         Toast.makeText(this, "广告加载器创建失败", Toast.
LENGTH_LONG).show();
485         return;
486     }
487     Log.d(TAG, "□ TAdNative广告加载器创建成功");
488
489     // 加载激励广告
490     Log.d(TAG, "□ 开始加载激励广告...");
491     adNativeLoader.loadRewardVideoAd(adSlot,
mRewardVideoListener);
492     //Toast.makeText(this, "正在加载广告...", Toast.
LENGTH_SHORT).show();
493     Log.d(TAG, "□ 激励广告加载请求已发送，等待回调...");
494 }
495
496 /**
497  * 初始化激励广告监听器
498  */
499 private void initRewardAdListeners() {
500     Log.d(TAG, "□ 开始初始化激励广告监听器...");
501
502     // 广告加载监听器
503     mRewardVideoListener = new TAdNative.
RewardVideoAdListener() {
504         @Override
505         public void onError(int code, String msg) {
506             Log.e(TAG, "□ 激励广告加载失败详情:");
507             Log.e(TAG, "  错误码: " + code);
508             Log.e(TAG, "  错误信息: " + msg);
509             Log.e(TAG, "  广告位ID: " + getResources().getString(R.
string.reward_media_id));
510             Log.e(TAG, "  宠物类型: " + mPendingPetType);
511
512             // 分析错误原因
513             analyzeRewardAdError(code, msg);
514
515             // 检查是否有本地配置可用
516             if (TAdManagerHolder.isLocalConfigSupported()) {
517                 Log.i(TAG, "□□ 检测到本地配置支持，

```

```

517 可能从本地配置重试");
518      showRewardAdRetryDialog("广告加载失败，是否重试？"
    );
519      } else {
520          //Log.w(TAG, "□□ 无本地配置支持，提供用户选择");
521          // 显示用户选择对话框，而不是直接跳转
522          showRewardAdFailDialog();
523      }
524  }
525
526  @Override
527  public void onRewardVideoAdLoad(TTRewardVideoAd
    ttRewardVideoAd) {
528      Log.d(TAG, "□ 激励广告加载成功！");
529      Log.d(TAG, " 广告对象: " + (ttRewardVideoAd != null ? "
    不为空" : "为空"));
530      if (ttRewardVideoAd != null) {
531          Log.d(TAG, " 广告类型: 激励视频");
532          Log.d(TAG, " 是否已缓存: " + (ttRewardVideoAd.
    getMediationManager() != null));
533      }
534      mTTRewardVideoAd = ttRewardVideoAd;
535      // 加载成功后立即展示
536      Log.d(TAG, "□ 准备展示激励广告...");
537      showRewardAd();
538  }
539
540  @Override
541  public void onRewardVideoCached() {
542      Log.d(TAG, "□ 激励广告缓存成功（方法1）");
543  }
544
545  @Override
546  public void onRewardVideoCached(TTRewardVideoAd
    ttRewardVideoAd) {
547      Log.d(TAG, "□ 激励广告缓存成功（方法2）");
548      Log.d(TAG, " 缓存的广告对象: " + (ttRewardVideoAd !=
    null ? "不为空" : "为空"));
549      mTTRewardVideoAd = ttRewardVideoAd;

```



```
550     }
551 };
552
553 // 广告展示监听器
554 mRewardVideoAdInteractionListener = new
TTRewardVideoAd.RewardAdInteractionListener() {
555     @Override
556     public void onAdShow() {
557         Log.d(TAG, "□ 激励广告开始展示");
558         Log.d(TAG, "  当前时间: " + System.currentTimeMillis());
559         Log.d(TAG, "  宠物类型: " + mPendingPetType);
560     }
561
562     @Override
563     public void onAdVideoBarClick() {
564         Log.d(TAG, "□ 激励广告被点击");
565     }
566
567     @Override
568     public void onAdClose() {
569         Log.d(TAG, "□ 激励广告关闭");
570         Log.d(TAG, "  用户可能未看完广告");
571     }
572
573     @Override
574     public void onVideoComplete() {
575         Log.d(TAG, "□ 激励广告视频播放完成");
576         Log.d(TAG, "  用户已观看完整个视频");
577     }
578
579     @Override
580     public void onVideoError() {
581         Log.e(TAG, "□ 激励广告视频播放错误");
582         Log.e(TAG, "  可能是网络问题或视频文件损坏");
583         Toast.makeText(CatDogActivity.this, "视频播放出错",
Toast.LENGTH_SHORT).show();
584     }
585
586     @Override
```

```

587     public void onRewardVerify(boolean rewardVerify, int
rewardAmount, String rewardName, int errorCode, String
errorMsg) {
588         Log.d(TAG, "□ 激励广告奖励验证");
589         Log.d(TAG, "  验证结果: " + rewardVerify);
590         Log.d(TAG, "  奖励数量: " + rewardAmount);
591         Log.d(TAG, "  奖励名称: " + rewardName);
592         Log.d(TAG, "  错误码: " + errorCode);
593         Log.d(TAG, "  错误信息: " + errorMsg);
594     }
595
596     @Override
597     public void onRewardArrived(boolean isRewardValid, int
rewardType, Bundle extraInfo) {
598         Log.d(TAG, "□ 激励广告奖励到达");
599         Log.d(TAG, "  奖励有效: " + isRewardValid);
600         Log.d(TAG, "  奖励类型: " + rewardType);
601         Log.d(TAG, "  额外信息: " + (extraInfo != null ? extraInfo.
toString() : "无"));
602         Log.d(TAG, "  目标宠物: " + mPendingPetType);
603
604         if (isRewardValid) {
605             // 奖励验证成功 · 跳转到录音界面
606             String petName = "cat".equals(mPendingPetType) ? "
小猫" : "小狗";
607             Log.d(TAG, "□ 奖励验证成功 · 准备跳转到" + petName
+ "录音界面");
608             Toast.makeText(CatDogActivity.this, "广告观看完成！
解锁" + petName + "沟通器成功！", Toast.LENGTH_LONG).show();
609             startRecordPlayActivity(mPendingPetType);
610         } else {
611             Log.w(TAG, "□□ 奖励验证失败 · 用户可能未完整观看广告
");
612             Toast.makeText(CatDogActivity.this, "奖励验证失败 ·
请重新尝试", Toast.LENGTH_SHORT).show();
613         }
614     }
615
616     @Override

```

```

617     public void onSkippedVideo() {
618         Log.w(TAG, "□□ 激励广告被跳过");
619         Log.w(TAG, " 用户未看完整个广告，无法获得奖励");
620         Toast.makeText(CatDogActivity.this, "
        需要看完整个广告才能解锁功能", Toast.LENGTH_SHORT).show();
621     }
622 };
623
624     Log.d(TAG, "□ 激励广告监听器初始化完成");
625 }
626
627 /**
628  * 分析激励广告加载错误
629  * @param code 错误代码
630  * @param msg 错误信息
631  */
632 private void analyzeRewardAdError(int code, String msg) {
633     Log.d(TAG, "□ 激励广告错误分析:");
634
635     // 常见错误代码分析
636     switch (code) {
637         case 40001:
638             Log.d(TAG, " 原因: 配置拉取失败 ( 网络问题或服务器异常
        ) ");
639             Log.d(TAG, " 建议: 本地配置和兜底机制将发挥作用");
640             break;
641         case 40002:
642             Log.d(TAG, " 原因: 广告位配置错误");
643             Log.d(TAG, " 建议: 检查广告位ID是否正确");
644             break;
645         case 40003:
646             Log.d(TAG, " 原因: 无广告填充");
647             Log.d(TAG, " 建议: 正常现象，可启用降级策略");
648             break;
649         case 40004:
650             Log.d(TAG, " 原因: 广告已被过滤");
651             Log.d(TAG, " 建议: 内容安全过滤，可重试");
652             break;
653         case 20005:

```

```

654         Log.d(TAG, " 原因: 全部代码位请求失败 (
聚合广告位无可用资源 ) ");
655         Log.d(TAG, " 说明: 所有聚合的广告平台都无法返回广告");
656         break;
657     case -8:
658         Log.d(TAG, " 原因: 网络超时");
659         Log.d(TAG, " 建议: 检查网络连接或使用本地配置");
660         break;
661     case -9:
662         Log.d(TAG, " 原因: 网络错误");
663         Log.d(TAG, " 建议: 本地配置将提供保障");
664         break;
665     default:
666         Log.d(TAG, " 原因: 其他错误 (" + code + ")");
667         Log.d(TAG, " 描述: " + msg);
668         break;
669     }
670
671     // 兜底机制状态检查
672     boolean localConfigSupported = TAdManagerHolder.
isLocalConfigSupported();
673     boolean fallbackSupported = TAdManagerHolder.
isFallbackSupported();
674
675     Log.d(TAG, " 本地配置: " + (localConfigSupported ? "□ 可用" :
"□ 不可用"));
676     Log.d(TAG, " 自定义兜底: " + (fallbackSupported ? "□ 可用" :
"□ 不可用"));
677
678     if (!localConfigSupported && !fallbackSupported) {
679         Log.i(TAG, " 说明: 当前为标准SDK · 基础功能完全正常");
680     }
681 }
682
683 /**
684  * 展示激励广告
685  */
686 private void showRewardAd() {
687     Log.d(TAG, "□ === 开始展示激励广告 ===");

```

```
688
689     if (mTTRewardVideoAd == null) {
690         Log.e(TAG, "□ 激励广告对象为空，无法展示");
691         Log.e(TAG, " 可能原因：广告加载失败或尚未完成");
692         Toast.makeText(this, "广告加载失败，请重试", Toast.
LENGTH_SHORT).show();
693         return;
694     }
695
696     Log.d(TAG, "□ 激励广告对象检查通过");
697     Log.d(TAG, " 广告对象状态: 不为空");
698     Log.d(TAG, " MediationManager: " + (mTTRewardVideoAd.
getMediationManager() != null ? "已创建" : "未创建"));
699
700     // 设置展示监听器并展示广告
701     Log.d(TAG, "□ 设置广告展示监听器...");
702     mTTRewardVideoAd.setRewardAdInteractionListener(
mRewardVideoAdInteractionListener);
703
704     Log.d(TAG, "□ 准备调用showRewardVideoAd...");
705     Log.d(TAG, " Activity状态: " + (this.isFinishing() ? "正在结束" :
"正常"));
706     Log.d(TAG, " 宠物类型: " + mPendingPetType);
707
708     try {
709         mTTRewardVideoAd.showRewardVideoAd(this);
710         Log.d(TAG, "□ showRewardVideoAd调用成功，
等待广告展示...");
711     } catch (Exception e) {
712         Log.e(TAG, "□ showRewardVideoAd调用异常: " + e.
getMessage());
713         e.printStackTrace();
714         Toast.makeText(this, "广告展示失败: " + e.getMessage(),
Toast.LENGTH_LONG).show();
715     }
716 }
717
718 /**
719  * 启动录音播放界面
```

```
720    * @param petType 宠物类型 : \"cat\" 或 \"dog\"
721    */
722    private void startRecordPlayActivity(String petType) {
723        Intent intent = new Intent(this, RecordPlayActivity.class);
724        intent.putExtra("pet_type", petType);
725        startActivity(intent);
726    }
727
728    @Override
729    protected void onDestroy() {
730        super.onDestroy();
731
732        // 销毁Banner广告
733        if (mCatBannerAd != null) {
734            mCatBannerAd.destroy();
735            Log.d(TAG, "Cat Banner广告已销毁");
736        }
737        if (mDogBannerAd != null) {
738            mDogBannerAd.destroy();
739            Log.d(TAG, "Dog Banner广告已销毁");
740        }
741
742        // 销毁激励广告
743        if (mTTRewardVideoAd != null && mTTRewardVideoAd.
getMediationManager() != null) {
744            mTTRewardVideoAd.getMediationManager().destroy();
745            Log.d(TAG, "激励广告已销毁");
746        }
747
748        Log.d(TAG, "onDestroy: 页面销毁");
749    }
750 }
751
```