电 子 科 技 大 学

实　验　报　告

学生姓名：Lolipop 学号：2018091202000　指导教师：xxx

实验地点：信软学院楼西 305　　实验时间：2020.12.19

一、实验名称：信息获取

二、实验学时：4

三、实验目的：掌握移动开发中信息获取的相关知识

四、实验原理：

1) Flutter 中的 amap_location 插件，是一个跨平台（iOS，Android）的高德地图定位 flutter 组件，目前实现直接获取定位和监听定位功能。使用该插件，需要先在高德官网上根据自己的应用程序信息申请 apikey。
2) 在大多数操作系统上，安装时不只是将权限授予应用程序。相反，开发人员必须在应用程序运行时向用户询问权限。Flutter 中的 permission_handler 插件提供了跨平台（iOS，Android）API 来请求权限并检查其状态，还可以打开设备的应用程序设置，以便用户可以授予权限。在 Android 上，可以显示请求权限的理由。
3) Flutter 中的 flutter_speed_dial 插件，实现了浮动按钮的菜单功能，实现点击该 SpeedDial 组件，跳出多个浮动按钮选项，在该实验中使用该组件进行页面的设计。
4) Flutter 中的 file_picker 插件，可让用户使用本机文件浏览器来选择单个或多个文件，并具有扩展名过滤支持。目前支持的功能有：使用操作

系统默认的本机选择器；使用自定义格式过滤选择文件，可以提供文件扩展名列表（pdf, svg, zip 等）；从云文件（GDrive, Dropbox, iCloud）中选择文件；单个或多个文件选择；不同的默认类型过滤（媒体，图像，视频，音频或其他）等。

5) Flutter 中的 image_picker 插件，适用于 iOS 和 Android，用于从图像库中拾取图像，并使用相机拍摄新照片。

6) Flutter 中的 video_player 插件，适用于 iOS，Android 和 Web 的 Flutter 插件，用于在 Widget 上播放视频，支持本地存储文件和网络视频文件。

五、实验内容：

1) 编写程序获取 GPS 信息

2) 编写程序访问相册获取图片

3) 编写程序调用摄像头

4) 编写程序访问相册获取视频，并播放

5) 编译、调试和查看程序运行结果

六、实验器材（设备、元器件）：

1) Windows 10 系统

2) Android Studio

3) Flutter

七、实验步骤：

1. 获取 GPS 信息

    1) 申请 apikey

2) 修改“项目目录/app/build.gradle”在 android/defaultConfig 节点修改 manifestPlaceholders，新增高德地图 key 配置。

3) 在 main.dart 的 main()方法中，新增高德地图 key。

4) 导入 dart 包，修改 pubspec.yaml，增加依赖。

5) 在 my_account_route.dart 中编写获取定位相关代码。

6) 界面设计：在 my_account_route.dart 完成界面设计代码。

2. 访问相册获取图片

1) 导入 dart 包，修改 pubspec.yaml，增加依赖。

2) 在 base_model_page.dart 中编写访问相册获取图片相关代码。

3) 界面设计

①导入 dart 包，修改 pubspec.yaml，增加依赖。

② 在 base_model_page.dart 中编写 SpeedDial 、FloatActionButton、Card 组件等界面设计相关代码。

3. 调用摄像头

1) 导入 dart 包，修改 pubspec.yaml，增加依赖。

2) 在 base_model_page.dart 中编写调用摄像头相关代码。

4. 获取视频并播放

1) 获取视频：在 base_model_page.dart 中编写获取视频相关代码，并跳转页面，同时传递视频路径参数。

2) 播放视频

①导入 dart 包，修改 pubspec.yaml，增加依赖。

②在 play_video_route.dart 中，接收视频路径参数。

③在 play_video_route.dart 中，编写播放视频相关代码。

八、实验结果与分析（含重要数据结果分析或核心代码流程分析）

1. 获取 GPS 信息

(1)申请 apikey，如图 8-1 所示。

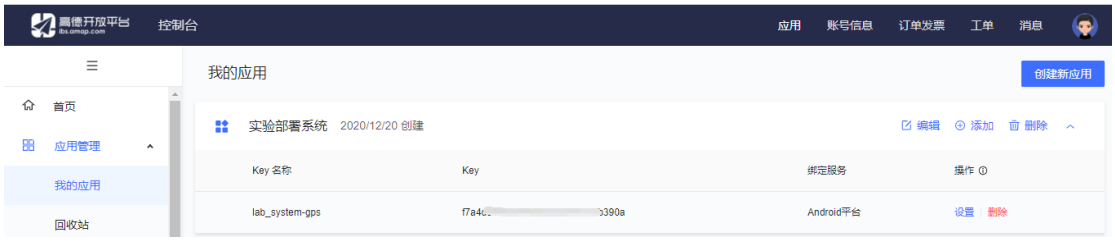

图 8-1 申请 apikey

(2)修改“项目目录/app/build.gradle”在 android/defaultConfig 节点修改 manifestPlaceholders，新增高德地图 key 配置，如代码 8-1 所示。

代码 8-1 配置高德地图 key 代码

```
android {
    ...
    defaultConfig {
        ...
        manifestPlaceholders = [
                AMAP_KEY : "f7a…390a", /// 高德地图 key
        ]
    }

    dependencies {
        // 注意这里需要在主项目增加一条依赖，否则可能发生编译不通过的情况
        implementation 'com.amap.api:location:latest.integration'
        ...
    }
    ...
```

```
    }
}
```

(3)在 main.dart 的 main()方法中，新增高德地图 key，如代码 8-2 所示。

代码 8-2 main()新增高德地图 key 代码

```
import 'package:amap_location/amap_location.dart';

void main() {
  AMapLocationClient.setApiKey("f7a4d5…b390a");

  runApp(new MaterialApp(
    home: SplashScreen(),
  ));
}
```

(4)导入 amap_location 和 permission_handler 的 dart 包，修改 pubspec.yaml，增加依赖。

(5)在定位功能所在的页面 my_account_route.dart 中，导入 amap_location 和 permission_handler 的 dart 包，如代码 8-3 所示。

代码 8-3 导入 dart 包代码

```
import 'package:amap_location/amap_location.dart';
import 'package:permission_handler/permission_handler.dart';
```

(6)在 my_account_route.dart 中编写_checkPersmission()方法，该方法用于申请定位权限 Permission.location。若申请权限成功，则直接获取定位，使用 AMapLocationClient 的 getLocation()方法，如代码 8-4 所示。

代码 8-4 申请定位权限_checkPersmission()代码

```
AMapLocation _loc;

@override
Future<bool> _checkPersmission() async {
  if (await Permission.location.request().isGranted) {
    AMapLocation loc = await AMapLocationClient.getLocation(true);
    setState(() {
      _loc = loc;
    });
    return true;
  }else{
    return false;
  }
}
```

(7)在 my_account_route.dart 中编写 getLocationStr()方法，该方法用于获取定位的结果（包括经纬度和具体位置），如代码 8-5 所示。

代码 8-5 获取定位结果 getLocationStr()代码

```
String getLocationStr(AMapLocation loc) {
  if (loc.isSuccess()) {
    if (loc.hasAddress()) {
      return "经纬度：${loc.latitude} ${loc.longitude}\n地址：${loc.formattedAddress}";
    } else {
      return "经纬度：${loc.latitude} ${loc.longitude}";
    }
  } else {
    return "定位失败:\n 错误:{code=${loc.code}, description=${loc.description}";
  }
}
```

(8)在 my_account_route.dart 中编写 getLocation()方法，该方法用于获取定

位后的展示，在该方法中使用 showDialog()返回一个对话框，将定位结果或"申请定位权限失败"的结果显示在该对话框上，若关闭该对话框，则同时关闭 AMapLocationClient，以防内存泄露，如代码 8-6 所示。

代码 8-6 展示定位结果 getLocation()代码

```
void getLocation(String str) {
  showDialog(
      context: context, //MyComponent
      builder: (BuildContext context) {
        return SimpleDialog(
          title: Text("定位"),
          // 边缘的形状
          shape: BeveledRectangleBorder(
              borderRadius: BorderRadius.all(Radius.circular(10.0)),
              side: BorderSide.none),
          children: <Widget>[
            //不用 SimpleDialogOption，就不会遵循 SimpleDialog 的间距
            SimpleDialogOption(
              child: Text(str),
            ),
            SimpleDialogOption(
              child: Row(
                mainAxisAlignment: MainAxisAlignment.end,
                children: <Widget>[
                  RaisedButton(
                    child: Text("关闭"),
                    onPressed: () {
                      AMapLocationClient.shutdown();
                      Navigator.of(context).pop(); //关闭弹框
                    },
                  )
                ],
```

```
                    ),
                ),
            ],
        );
    });
}
```

(8)界面设计

在 my_account_route.dart 中编写 build(BuildContext context)方法，进行页面设计，该页面主要包含一个 ListView，在该列表内实现“登出”和“定位”功能，在此只展示定位功能的代码。

当点击“GPS 定位”时，启动 AMapLocationClient，然后调用 _checkPersmission()函数，若申请成功，则调用 getLocationStr()方法返回定位结果到 getLocation()上，若申请失败，则返回"申请定位权限失败"的字符串到 getLocation()上。如代码 8-7 所示。

代码 8-7 界面设计代码

```
Widget build(BuildContext context) {
  return Scaffold(
      backgroundColor: backgroundColor,
      appBar: AppBar(
        elevation: elevationAppBar,
        title: Text('我的'),
        centerTitle: true,
      ),
      body: Scrollbar(
        child: ListView(
          children: [
            Card(...),
            Card(
```

```
                    elevation: elevationCard,
                    margin: EdgeInsets.symmetric(
                        vertical: marginVerticalCard,
                        horizontal: marginHorizontalCard),
                    shape: RoundedRectangleBorder(
                        borderRadius:
                        BorderRadius.all(Radius.circular(borderRadiusCard))),
                    child: Container(
                      child: ListTile(
                          leading: Icon(Icons.gps_fixed, color: accentColor),
                          title: Text('GPS 定位'),
                          onTap: () async {
                            AMapLocationClient.startup(new AMapLocationOption(
                                desiredAccuracy:
CLLocationAccuracy.kCLLocationAccuracyHundredMeters));
                            if(await _checkPersmission()){
                              getLocation(getLocationStr(_loc));
                            }else{
                              getLocation("申请定位权限失败");
                            }
                          })
                      ),
                    )
                ],
              ),
            ));
}
```

my_account_route.dart 文件编写的“我的”页面，点击“GPS 定位”，申请定位权限，定位结果如图 8-2 所示。

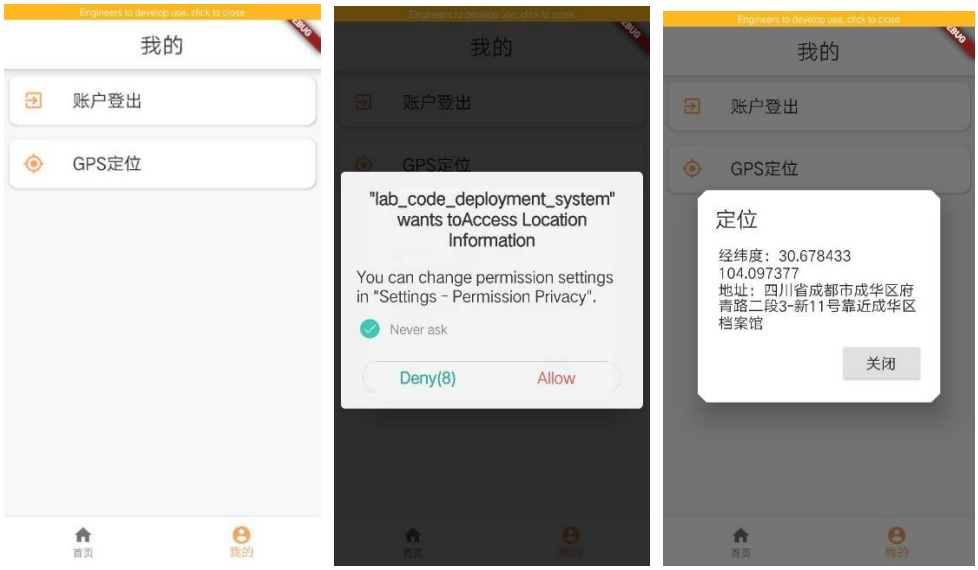
Engineers to develop use, click to close
我的
账户登出
GPS定位
首页
我的
"lab_code_deployment_system" wants toAccess Location Information
You can change permission settings in "Settings - Permission Privacy".
Never ask
Deny(8)
Allow
定位
经纬度：30.678433
104.097377
地址：四川省成都市成华区府青路二段3-新11号靠近成华区档案馆
关闭

图 8-2 定位结果

2. 访问相册获取图片

(1)导入 file_picker 的 dart 包，修改 pubspec.yaml，增加依赖。

(2)在获取图片功能所在的页面 base_model_page.dart 中，导入 file_picker 的 dart 包，如代码 8-8 所示。

代码 8-8 导入 dart 包代码

```
import 'package:file_picker/file_picker.dart';
```

(3)在 base_model_page.dart 中编写_selectFiles()方法, 该方法用于访问相册获取文件，在该页面中自定义访问文件格式_extension 为“jpg”。在该方法中使用 FilePicker.platform.pickFiles()方法访问相册获取文件, 如代码 8-9 所示。

代码 8-9 获取图片_selectFiles()代码

```
// 选择文件并赋值给 _selectedFile
void _selectFiles() async {
  setState(() => _loadingFiles = true);
  try {
    _selectedFile = (await FilePicker.platform.pickFiles(
      // 需要自定义 _extension
      type: FileType.custom,
      // 立即载入内存得到 Uint8List 格式
      withData: true,
      // 禁止压缩文件
      allowCompression: false,
      allowMultiple: _allowMultiple,
      allowedExtensions: (_extension?.isNotEmpty ?? false)
          ? _extension?.replaceAll(' ', '')?.split(',')
          : null,
    ))
        ?.files;
```

```
        if (_selectedFile != null || _selectedFile.isNotEmpty) {
          _selectedFileCount = _selectedFile.length.toString();
          Fluttertoast.showToast(msg: '点击按钮上传文件执行实验代码吧！');
        }
      } on PlatformException catch (e) {
        print("Unsupported operation" + e.toString());
      } catch (ex) {
        print(ex);
      }
      if (!mounted) return;
      setState(() => _loadingFiles = false);
    }
```

base_model_page.dart 文件编写模型详细页面“BaseModelPage”，点击“上传图片”按钮，申请访问相册，选择图片上传，在该页面的“选择照片Card”上显示获取图片的数量和名称，点击“x”清空已选择文件，结果如图 8-3 所示。

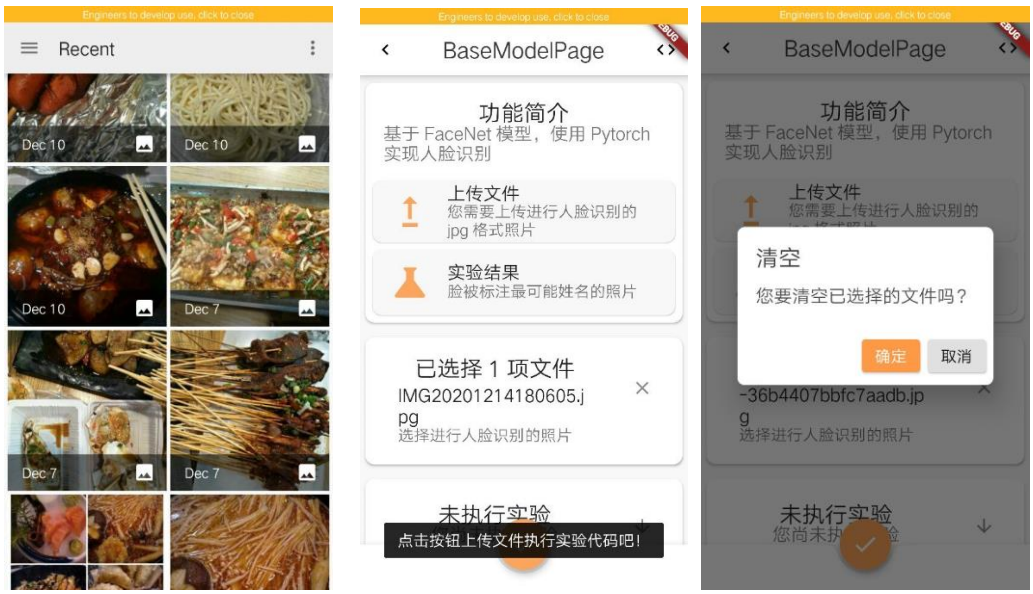

图 8-3 获取图片

(4)界面设计

①导入 flutter_speed_dial 的 dart 包，修改 pubspec.yaml，增加依赖。

②在访问相册获取图片功能所在的页面 base_model_page.dart 中，导入 flutter_speed_dial 的 dart 包，如代码 8-10 所示。

代码 8-10 导入 dart 包代码

```
import 'package:flutter_speed_dial/flutter_speed_dial.dart';
```

③在 base_model_page.dart 页面中编写_getSpeedDial()方法，返回一个 FloatingActionButton 菜单 SpeedDial。点击 SpeedDial，SpeedDial 上方显示三个按钮，分别用于获取照片、调用摄像头拍摄、获取视频并播放，点击按钮时，分别调用_selectFiles()、_takePhotos()、_playVideo()方法，如代码 8-11 所示。

表 8-11 界面设计-SpeedDial 组件代码

```
// Float Action Button 菜单，可选照片、拍摄、视频
SpeedDial _getSpeedDial() {
  return SpeedDial(
      marginBottom: 16,
      marginRight: 16,
      backgroundColor: accentColor,
      tooltip:'选择用于执行实验代码的文件',
      child: Icon(
          Icons.cloud_download,
          color: accentTextColor,
          size: iconSizeTrailing),
      children:[
        SpeedDialChild(
            child: Icon(Icons.photo,
```

```
                    color: accentTextColor,
                    size: iconSizeTrailing),
                backgroundColor: Colors.red,
                label: '照片',
                labelStyle: TextStyle(fontSize: 18.0),
                onTap: () => _selectFiles(),
            ),
            SpeedDialChild(
              child: Icon(Icons.photo_camera,
                  color: accentTextColor,
                  size: iconSizeTrailing),
              backgroundColor: Colors.orange,
              label: '拍摄',
              labelStyle: TextStyle(fontSize: 18.0),
              onTap: () => _takePhotos(),
            ),
            SpeedDialChild(
              child: Icon(Icons.video_call,
                  color: accentTextColor,
                  size: iconSizeTrailing),
              backgroundColor: Colors.green,
              label: '视频',
              labelStyle: TextStyle(fontSize: 18.0),
              onTap: () => _playVideo(),
            ),
          ]
    );
  }
```

base_model_page.dart 文件编写模型详细页面“BaseModelPage”，界面设计_getSpeedDial()方法返回的 SpeedDial 组件如图 8-4 所示。

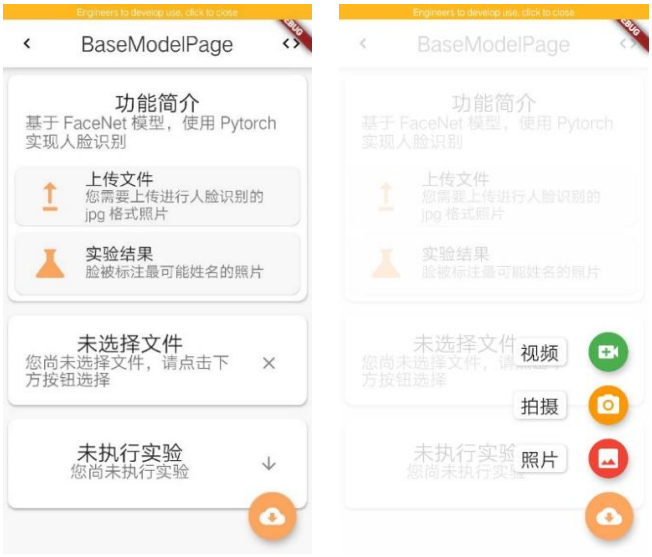

图 8-4 界面设计-SpeedDial 组件

④在 base_model_page.dart 页面中编写_getFloatActionButton()方法，返回一个 FloatingActionButton。点击按钮，上传选择文件并执行实验代码，如代码 8-12 所示。

代码 8-12 界面设计-FloatingActionButton 组件代码

```
  // 根据是否已选择文件获取底部的 Float Action Button
    FloatingActionButton _getFloatActionButton() {
    return FloatingActionButton(
      backgroundColor: accentColor,
      onPressed: () {
          showAlertDialog(context, '执行实验', '您确定要开始执行实验吗？',
confirmFunction: () {
            _handleFiles();
          });
      },
      tooltip: '上传选择文件并执行实验代码',
      child: Icon(
          Icons.done,
          color: accentTextColor,
```

```
            size: iconSizeTrailing),
        splashColor: splashColor,
    );
}
```

base_model_page.dart 编写模型详细页面“BaseModelPage”，界面设计_getFloatActionButton()方法返回的 FloatingActionButton 组件如图 8-5 所示。

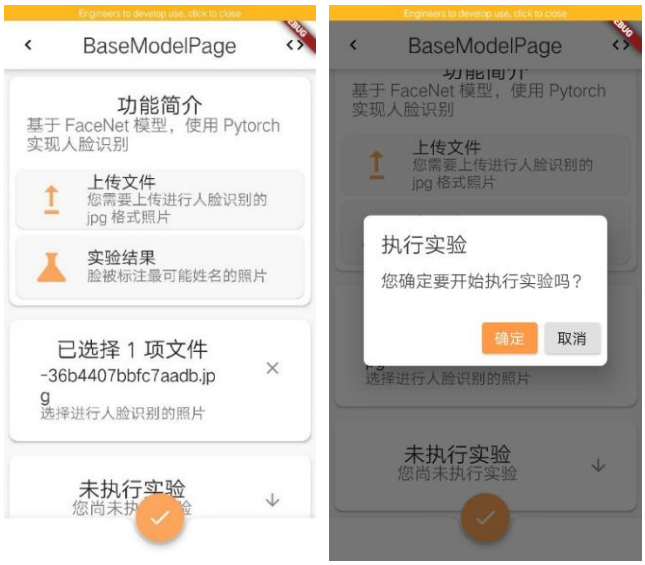

图 8-5 界面设计- FloatingActionButton 组件

⑤在 base_model_page.dart 页面中编写_getUploadSourceWidget()方法，返回一个 listTileWidget 组件，显示获取文件的数量和名称，如代码 8-13 所示。

代码 8-13 界面设计-listTileWidget 代码

```
// 获取选择文件的 ListTile
List<Widget> _getUploadSourceWidget() {
  if (_selectedFile == null || _selectedFile.isEmpty){
    List<Widget> listTileWidget = [
      ListTile(
        title: Text(_pickedFile.path),
```

```
            subtitle: Text(_uploadFileSubtitle),
          )
        }.toList();
        return listTileWidget;
      }else{
        List<Widget> listTileWidget = _selectedFile.map((e) {
          return ListTile(
            title: Text(e.name),
            subtitle: Text(_uploadFileSubtitle),
          );
        }).toList();
        return listTileWidget;
      }
    }
```

⑥在 base_model_page.dart 页面中编写_getUploadSourceSectionCard()方法，返回一个 Card 组件，显示获取文件的信息，如代码 8-14 所示。

表 8-14 界面设计-Card 组件代码

```
  // 获取已选择文件的 Card
  Card _getUploadSourceSectionCard() {
    return Card(
      elevation: elevationCard,
      margin: EdgeInsets.symmetric(
          vertical: marginVerticalCard, horizontal: marginHorizontalCard),
      shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.all(Radius.circular(borderRadiusCard))),
      child: Column(
        children: [
          ListTile(
            contentPadding: EdgeInsets.only(left: 20.0, top: 16.0, right: 20.0,
bottom: 16.0),
            title: Text(
```

```
                (_selectedFile == null || _selectedFile.isEmpty) && (_pickedFile ==
null)
                    ? '未选择文件'
                    : '已选择 $_selectedFileCount 项文件',
                style: TextStyle(fontSize: fontSizeTitle),
                textAlign: TextAlign.center,
              ),
              subtitle: Column(
                children: (_selectedFile == null || _selectedFile.isEmpty) &&
(_pickedFile == null)
                    ? [
                        Text(
                          '您尚未选择文件，请点击下方按钮选择',
                          style: TextStyle(fontSize: fontSizeSubTitle),
                        )
                      ]
                    : _getUploadSourceWidget(),
              ),
              trailing: IconButton(
                icon: Icon(Icons.close),
                tooltip: '清空已选择的文件',
                onPressed: () => {
                  if ((_selectedFile == null || _selectedFile.isEmpty) &&
(_pickedFile == null))
                    {Fluttertoast.showToast(msg: '您还未选择文件喔！')}
                  else {
                    showAlertDialog(context, '清空', '您要清空已选择的文件吗？',
                        confirmFunction: () {
                      setState(() {
                        _selectedFile = null;
                        _pickedFile = null;
                      });
                    })
                  }
```

```
                },
              ),
            )
          ],
        ),
      );
  }
```

⑦在 base_model_page.dart 中编写 build(BuildContext context)方法，进行页面设计。如果未选择文件，则显示_getSpeedDial()方法返回的 SpeedDial 组件；如果已选择文件，则显示_getFloatActionButton()方法返回的 FloatingActionButton 组件，如代码 8-15 所示。

表 8-15 界面设计代码

```
@override
Widget build(BuildContext context) {
  return WillPopScope(
      onWillPop: () async => showDialog(
          context: context,
          builder: (context) => AlertDialog(
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.all(
                        Radius.circular(borderRadiusDialog))),
                title: Text('返回',
                    style: TextStyle(color: primaryTextColorDark)),
                content: Text(
                    '您要退出实验 $_modelName 页面吗？' +
                        '正在进行的' +
                        (_onHandleProcess == true
                            ? '实验处理操作'
                            : _onDownloadProcess
                                ? '下载实验结果'
```

```
                                        : '操作') +
                                '将会结束。',
                            style: TextStyle(color: primaryTextColor)),
                        actions: <Widget>[
                          RaisedButton(
                              color: accentColorDark,
                              textColor: accentTextColor,
                              splashColor: splashColor,
                              shape: RoundedRectangleBorder(
                                  borderRadius: BorderRadius.all(
                                      Radius.circular(borderRadiusButton))),
                              child: Text('确定'),
                              onPressed: () => Navigator.of(context).pop(true)),
                          RaisedButton(
                              shape: RoundedRectangleBorder(
                                  borderRadius: BorderRadius.all(
                                      Radius.circular(borderRadiusButton))),
                              child: Text('取消'),
                              onPressed: () => Navigator.of(context).pop(false)),
                        ])),
                child: Scaffold(
                  appBar: AppBar(
                    elevation: elevationAppBar,
                    leading: IconButton(
                      icon: Icon(Icons.keyboard_arrow_left),
                      onPressed: () => Navigator.pop(context),
                      tooltip: '快速返回上一页面',
                    ),
                    title: Text(this._modelName),
                    actions: [
                      IconButton(
                        icon: Icon(Icons.code),
                        onPressed: () => _launchURL(),
                        tooltip: '打开此实验代码的相关链接',
```

```
                    )
                ],
                centerTitle: true,
            ),
            bottomNavigationBar: (_selectedFile == null || _selectedFile.isEmpty)
&& (_pickedFile == null)
                ? null
                : BottomAppBar(
                    shape: const CircularNotchedRectangle(),
                    child: Container(height: iconSizeFloatButton,),
                ),
            body: ListView(
                children: [
                    Card(
                        elevation: elevationCard,
                        margin: EdgeInsets.symmetric(
                            vertical: marginVerticalCard,
                            horizontal: marginHorizontalCard),
                        shape: RoundedRectangleBorder(
                            borderRadius:
                                BorderRadius.all(Radius.circular(borderRadiusCard))),
                        child: Column(
                            children: [
                                _getReadMeListTile(),
                                _getTipCard('upload'),
                                _getTipCard('result')
                            ],
                        ),
                    ),
                    _getUploadSourceSectionCard(),
                    _getResultSourceSectionCard(),
                ],
            ),
            floatingActionButton: (_selectedFile == null || _selectedFile.isEmpty)
```

```
&& (_pickedFile == null)
                    ? _getSpeedDial()
                    : _getFloatActionButton(),
              floatingActionButtonLocation: (_selectedFile == null ||
_selectedFile.isEmpty) && (_pickedFile == null)
                    ? null
                    : FloatingActionButtonLocation.centerDocked,
            ));
  }
}
```

3. 调用摄像头

(1)导入 image_picke 的 dart 包，修改 pubspec.yaml，增加依赖。

(2)在调用摄像头并拍摄图片功能所在的页面 base_model_page.dart 中，导入 image_picke 的 dart 包，如代码 8-16 所示。

代码 8-16 导入 dart 包代码

```
import 'package:image_picker/image_picker.dart';
```

(3)在 base_model_page.dart 中编写_takePhotos()方法，该方法用于调用摄像头拍摄。在该方法中使用 ImagePicker.getImage(source: ImageSource.camera)方法调用摄像头，如代码 8-17 所示。

表 8-17 调用摄像头_takePhotos()代码

```
PickedFile _pickedFile; // 拍照
String _selectedFileCount;

// 拍照
void _takePhotos() async {
  final picker = ImagePicker();
  _pickedFile = await picker.getImage(source: ImageSource.camera);

  setState(() {
    if (_pickedFile != null) {
      _selectedFileCount = "1";
      Fluttertoast.showToast(msg: '点击按钮上传文件执行实验代码吧！');
    } else {
      print('No image selected.');
    }
  });
}
```

base_model_page.dart 文件编写模型详细页面“BaseModelPage”，点击“拍摄”按钮，申请访问摄像头，调用摄像头拍摄，点击“确定”后拍摄的图片上传至该页面，显示获取图片的数量和名称，结果如图 8-6 所示。

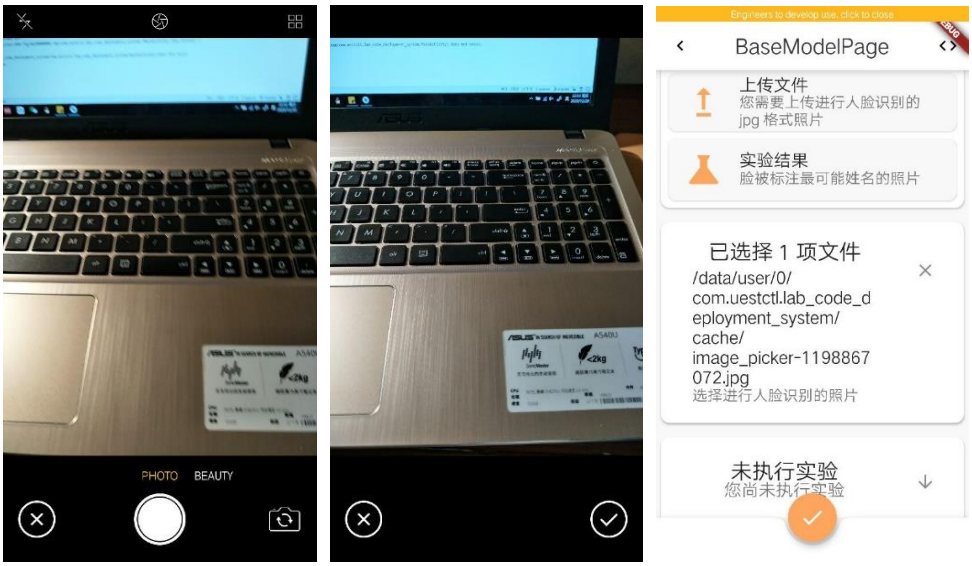

图 8-6 调用摄像头

4. 获取视频并播放

(1)获取视频

在 base_model_page.dart 中编写_playVideo()方法，该方法用于访问相册获取获取视频并播放。在该方法中使用 ImagePicker. getVideo(source: ImageSource.gallery)方法访问相册获取视频，并跳转页面至 VideoAppPage 类，同时传递视频路径参数_video，如代码 8-18 所示。

表 8-18 选择视频_playVideo()代码

```
// 选择视频
void _playVideo() async {
  final picker = ImagePicker();
  _pickedFile = await picker.getVideo(source: ImageSource.gallery);

  setState(() {
    if (_pickedFile != null) {
      _selectedFileCount = "1";
      _video = File(_pickedFile.path);
      // 传递参数
      Navigator.push(context, MaterialPageRoute(builder: (context) =>
VideoAppPage(video: _video)));
    } else {
      print('No video selected.');
    }
  });
}
```

base_model_page.dart 文件编写模型详细页面“BaseModelPage”，点击“视频”按钮，申请访问相册，选择视频，跳转至视频播放页面，播放视频，返回模型详细页面后上传该视频，在该页面的“选择照片 Card”上显示获取视

频的数量和名称，结果如图 8-7 所示。

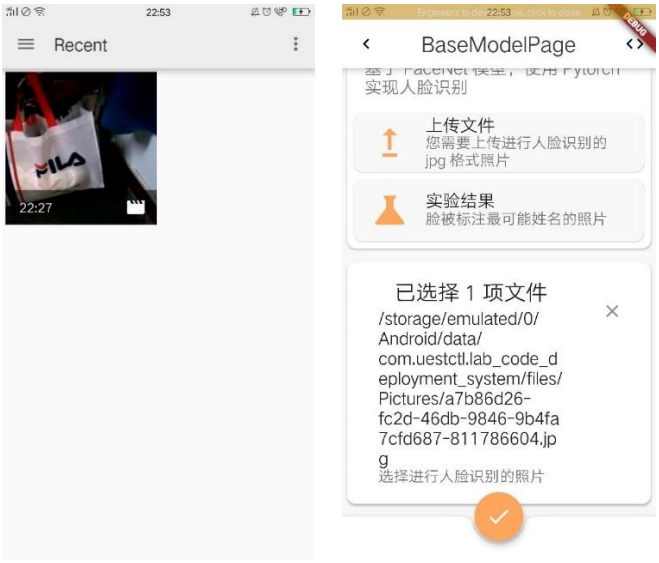

图 8-7 获取视频

(2)播放视频

①导入 video_player 的 dart 包，修改 pubspec.yaml，增加依赖。

②在播放视频功能所在的页面 play_video_route.dart 中，导入 video_player 的 dart 包。

③接收视频路径参数。

④编写播放视频代码。

使用 VideoPlayerController.file()方法加载 File 类型的文件，返回 VideoPlayerController。

使用_controller.initialize()初始化播放器；_controller.dispose()释放播放器资源；控制视频使用_controller.pause()和_controller.play()控制暂停和播放。

如代码 8-19 所示。

代码 8-19 play_video_route.dart 代码

```
import 'package:lab_code_deployment_system/pages/styles/widget_style.dart';
import 'package:flutter/material.dart';
import 'package:video_player/video_player.dart';
import 'dart:io';

class VideoAppPage extends StatefulWidget {
  // 接收参数
  final File video;
  VideoAppPage({Key key ,@required this.video}):super(key:key);
  @override
  _VideoAppState createState() => _VideoAppState();
}

class _VideoAppState extends State<VideoAppPage> {
  VideoPlayerController _controller;
  @override
  void initState() {
    super.initState();
    // 组件传参
    _controller = VideoPlayerController.file(widget.video)
      ..initialize().then((_) {
        setState(() {});
      });
  }
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Video Demo',
      home: Scaffold(
        appBar: AppBar(
          elevation: elevationAppBar,
          leading: IconButton(
            icon: Icon(Icons.keyboard_arrow_left),
```

```
            onPressed: () => Navigator.pop(context),
            tooltip: '快速返回上一页面',
          ),
          title: Text('Video Demo'),
        ),
        bottomNavigationBar: BottomAppBar(
          shape: const CircularNotchedRectangle(),
          child: Container(
            height: iconSizeFloatButton,
          ),
        ),
        body: Center(
          child: _controller.value.initialized
              ? AspectRatio(
            aspectRatio: _controller.value.aspectRatio,
            child: VideoPlayer(_controller),
          )
              : Container(),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () {
            setState(() {
              _controller.value.isPlaying
                  ? _controller.pause()
                  : _controller.play();
            });
          },
          child: Icon(
            _controller.value.isPlaying ? Icons.pause : Icons.play_arrow,
          ),
        ),
        floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
      ),
    );
```

```
  }
  @override
  void dispose() {
    super.dispose();
    _controller.dispose();
  }
}
```

play_video_route.dart 文件编写视频播放页面“VideoAppPage”，点击按钮，控制视频播放与暂停，结果如图 8-8 所示。

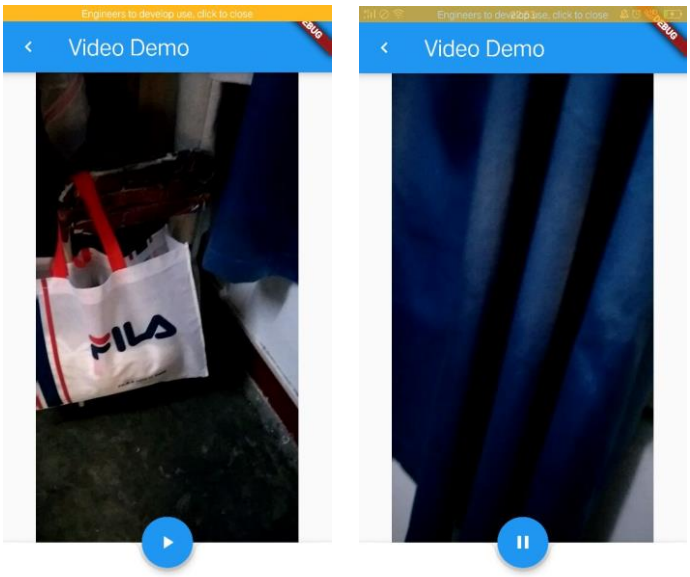

图 8-8 播放视频

九、总结及心得体会：

本次实验调用了 amap_location、file_picker、image_picker 多个 flutter 插件，实现了应用程序获取 GPS 信息、访问相册获取图片、调用摄像头、访问相册获取视频并播放等功能，通过本次实验，我掌握移动开发中信息获取的相关知识，了解了 Flutter 的混合式开发。

本次实验时通过 amap_location 高德地图定位 flutter 组件来实现 GPS 定位，调用 amap_location，首先需要在官网申请适用于 android 和 ios 的 apikey。先在官网上创建新应用，使用 Android Studio 的 Terminal 工具，输入“keytool -v -list -keystore　keystore 文件路径”命令获取应用安全码 SHA1，填写完成后方可获得该应用的 apikey，并且需要在应用的配置文件和 main()函数中配置。

本次实验时通过 permission_handler 插件实现 GPS 定位、访问相册、调用摄像头、文件存储等权限申请。

本次实验使用 flutter_speed_dial 插件，实现 SpeedDial 组件，作为浮动按钮菜单，使界面设计更合理。

十、对本实验过程及方法、手段的改进建议：

本次实验使用的一些插件暂时只支持 iOS 端和 Android 端，因此应用无法部署到 web 端。可以考虑使用其它的插件对 web 端进行原生支持。

当前视频播放页面只实现播放和暂停功能，可以多添加其他功能，例如：倍速播放、循环播放、音量大小等，充分利用 video_player 插件支持功能。

file_picker 插件可实现多种不同的默认类型过滤（媒体，图像，视频，音频或其他），在本次实验中使用文件扩展名 jpg 自定义格式过滤选择文件，可尝试过滤选择其他类型的文件。

报告评分：

指导教师签字：