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：2020.12.16

一、实验名称：移动客户端界面

二、实验学时：4

三、实验目的：掌握移动界面设计有关的知识

四、实验原理：

1) 使用 flutter 移动应用开发框架。

2) Flutter 框架的界面 UI 设计开发。

3) Dart 语言实现的移动端 UI 界面设计与开发。

五、实验内容：

1) 安装并使用 flutter 与安卓虚拟机；

2) 设计移动应用的登录界面；

3) 创建 Login 和 Home 页面布局；

4) 编写登录按钮处理事件；

5) 实现定制的 ListView 界面（RecyclerView）；

6) 编译、调试和查看程序运行结果。

六、实验器材（设备、元器件）：

装有 Windows 10 系统的电脑一台。

七、实验步骤：

1) 安装并启动 Android Studio+flutter+Android Emulator 开发环境。
2) 编写启动文件 main.dart 代码，实现全局用户 token 验证函数。
3) 设计并编写首页（Home.dart）布局界面。
4) 实现定制的 ListView 界面（RecyclerView）。
5) 设计并编写登录页（Login.dart）布局界面。
6) 实现前端输入验证功能。
7) 编写登录按钮处理事件，与后端相连接。
8) 编译生成目标程序，在真机和模拟器中调试移动应用程序。

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) 安装并使用 Android Studio、安卓虚拟机调试环境与 flutter&&dart 编程环境，并创建该 flutter 项目（lab_code_deployment_system）。
2) 编写 flutter 应用入口文件代码 main.dart，添加全局函数，实现用户与后端 token 的校验并获得用户信息。如代码 2-1 和 2-2 所示：

代码 2-1 flutter 项目整体入口文件

```
import 'package:flutter/material.dart';
import 'package:lab_code_deployment_system/http/users.dart' as UsersApi;
import 'package:lab_code_deployment_system/pages/home.dart';
import 'package:lab_code_deployment_system/pages/login.dart';


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

```
void main() {
  AMapLocationClient.setApiKey("f7a4d569eeaf0f2fbbbdcb91530b390a");

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

class SplashScreen extends StatefulWidget {
  @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    readFromSharedPreferences();
  }
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
          child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          FlutterLogo(
            size: 100.0,
          ),
        ],
      )),
    );
  }
}
```

代码 2-2 与后端交互验证用户 token 的全局函数

```
void readFromSharedPreferences() async {
  if (await UsersApi.checkUserToken()) {
    // 校验用户 token 是否有效
    Navigator.of(context)
        .pushReplacement(MaterialPageRoute(builder: (context) => Home()));
```

```
  } else {
    // 用户 token 无效或未获取到 userInfo
    Navigator.of(context)
        .pushReplacement(MaterialPageRoute(builder: (context) => Login()));
  }
}
```

3) 首先设计并编写主界面。实现主界面框架，下方显示两个 tab 栏切换按钮以及显示图标，上方显示当前 tab 栏页面的名称，在上下栏中间显示该 tab 栏页面的内容，分别是首页 tab 栏的模型 model 类型列表（如图 3-1 所示），以及我的 tab 栏的个人信息列表，以及账户登出、GPS 定位等功能按钮（如图 3-2 所示）。如代码 3-1 和 3-2 所示：

代码 3-1 首页 tab 栏页面中间列表部分

```
import 'package:flutter/material.dart';
import
'package:lab_code_deployment_system/pages/home_routes/model_list_route.dar
t';
import
'package:lab_code_deployment_system/pages/home_routes/my_account_route.d
art';
import 'package:lab_code_deployment_system/pages/styles/color.dart';
import 'package:lab_code_deployment_system/pages/styles/widget_style.dart';
```

```
class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primaryColor: primaryColor,
        primaryColorDark: primaryColorDark,
        primaryColorBrightness: primaryColorBrightness,
        primaryTextTheme: TextTheme(
          headline1: TextStyle(color: primaryTextColorDark),
          headline2: TextStyle(color: primaryTextColorDark),
          headline3: TextStyle(color: primaryTextColorDark),
          headline4: TextStyle(color: primaryTextColorDark),
          headline5: TextStyle(color: primaryTextColorDark),
          headline6: TextStyle(color: primaryTextColorDark),
          subtitle1: TextStyle(color: primaryTextColorLight),
          subtitle2: TextStyle(color: primaryTextColorLight),
          bodyText1: TextStyle(color: primaryTextColor),
          bodyText2: TextStyle(color: primaryTextColor),
          button: TextStyle(color: primaryTextColor),
          caption: TextStyle(color: primaryTextColorLight),
```

```
            overline: TextStyle(color: primaryTextColorLight),
          ),
          accentColor: accentColor,
          accentColorBrightness: accentColorBrightness,
          highlightColor: highlightColor,
          splashColor: splashColor,
          scaffoldBackgroundColor: backgroundColor,
          visualDensity: VisualDensity.adaptivePlatformDensity,
        ),
        home: HomePage(),
      );
    }
}
```

代码 3-2 首页页面框架 tab 栏以及切换

```
class HomePage extends StatefulWidget {
  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int _currentIndex = 0;
```

```
  final List<Widget> _children = [HomeModelListRoute(),
HomeMyAccountRoute()];


  final List<BottomNavigationBarItem> _list = <BottomNavigationBarItem>[
    BottomNavigationBarItem(
      icon: Icon(Icons.home),
      label: '首页',
      activeIcon: Icon(Icons.home),
    ),
    BottomNavigationBarItem(
      icon: Icon(Icons.account_circle),
      label: '我的',
      activeIcon: Icon(Icons.account_circle),
    ),
  ];


  @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('退出',
                    style: TextStyle(color: primaryTextColorDark)),
                content: Text('您要退出实验室代码部署系统吗？',
                    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(
              bottomNavigationBar: BottomNavigationBar(
                backgroundColor: primaryColor,
                type: BottomNavigationBarType.fixed,
                onTap: onTabTapped,
                currentIndex: _currentIndex,
                fixedColor: accentColor,
                items: _list,
              ),
              body: _children[_currentIndex],
            ));
  }


  void onTabTapped(int index) {
    setState(() {
      _currentIndex = index;
    });
  }
}
```

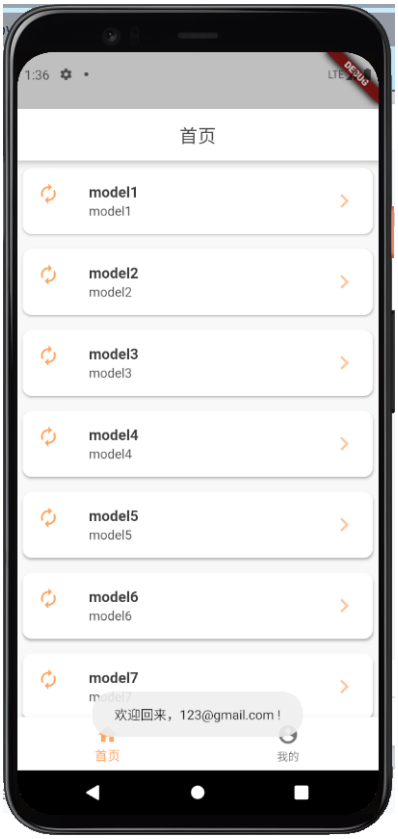

图 3-1 首页页面布局与框架

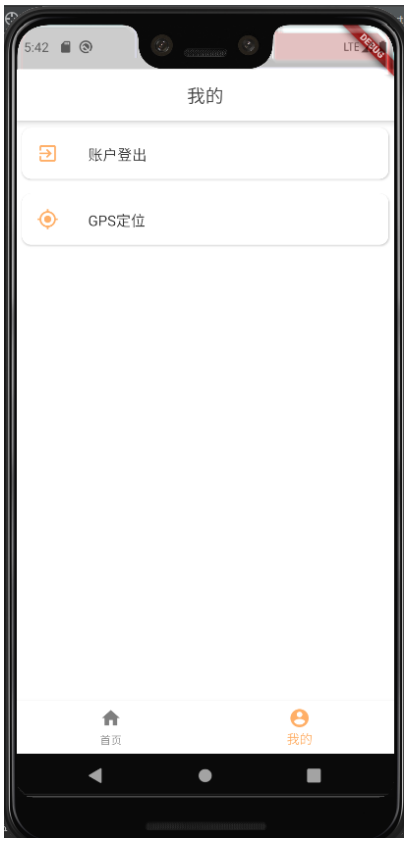

图 3-2 首页我的 tab 栏页面布局

4) 设计并编写登录页面，实现登录用户名输入框和登录密码输入框，以及两个输入框的输入文字提示，用户输入内容的前端正确性验证以及相应的错误提示，以及密码输入框的密码输入特效和其右边的明文显示按钮及其功能，如图 4-1 所示。代码如 4-1 至 4-4 所示：

代码 4-1 设置登录页面布局与框架

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:lab_code_deployment_system/http/users.dart' as UsersApi;
import 'package:lab_code_deployment_system/pages/home.dart';
import 'package:lab_code_deployment_system/pages/styles/color.dart';
import 'package:lab_code_deployment_system/settings.dart';
import
'package:lab_code_deployment_system/utils/shared_preferences_utils.dart';


class Login extends StatefulWidget {
  @override
  _Login createState() => _Login();
}


class _Login extends State<Login> {
```

```
  final String _pageTitle = 'Lab Code System';
  final String _usernameLabel = 'Email /  登录邮箱';
  final String _usernameNeedError = '请输入账户名';
  final String _usernameFormatError = '请输入正确的邮箱地址';
  final String _passwordLabel = 'Password /  密码';
  final String _passwordNeedError = '请输入登录密码';
  final String _loginText = 'Log in';
  final String _loginNeedUsernameAndPasswordError = '请输入您的账户名或密码！
';
  final String _loginErrorText = '登录失败，您的账户名或密码可能输入错误！';

  final _formKey = GlobalKey<FormState>();
  final TextEditingController _emailController = new TextEditingController();
  final TextEditingController _passwordController = new TextEditingController();
  final FocusNode _emailFocusNode = new FocusNode();
  final FocusNode _passwordFocusNode = new FocusNode();

  String _username, _password;
  bool _isObscure = true;
  bool _isLogin = false;
  Color _eyeColor;
```

```
  var _emailReg = RegExp(

r"[\w!#$%&'*+/=?^_`{|}~-]+(?:\.[\w!#$%&'*+/=?^_`{|}~-]+)*@(?:[\w](?:[\w-]*[\w])?\.)+[\
w](?:[\w-]*[\w])?");

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        theme: ThemeData(
          primaryColor: accentColorDark,
          primaryColorBrightness: accentColorBrightness,
        ),
        home: Scaffold(
          body: Form(
            key: _formKey,
            child: ListView(
              padding: EdgeInsets.symmetric(horizontal: 22.0),
              children: <Widget>[
                SizedBox(
                  height: kToolbarHeight,
                ),
                buildTitle(),
```

```
                    buildTitleLine(),
                    SizedBox(height: 70.0),
                    buildEmailTextField(),
                    SizedBox(height: 30.0),
                    buildPasswordTextField(context),
                    SizedBox(height: 60.0),
                    buildLoginButton(context),
                    SizedBox(height: 30.0),
                  ],
                ),
              ),
            ));
  }
```

代码 4-2 设置输入框等具体组件样式

```
Padding buildTitle() {
  return Padding(
    padding: EdgeInsets.all(8.0),
    child: Text(
      _pageTitle,
      style: TextStyle(color: accentColor, fontSize: 30.0),
    ),
```

```
    );
}

Padding buildTitleLine() {
    return Padding(
        padding: EdgeInsets.only(left: 12.0, top: 4.0),
        child: Align(
            alignment: Alignment.bottomLeft,
            child: Container(
                color: accentColorLight,
                width: 40.0,
                height: 2.0,
            ),
        ),
    );
}

TextFormField buildEmailTextField() {
    return TextFormField(
        autofocus: true,
        controller: _emailController,
        focusNode: _emailFocusNode,
```

```
      cursorColor: primaryTextColor,
      decoration: InputDecoration(
        labelText: _usernameLabel,
      ),
      validator: (String value) {
        if (value.isEmpty) {
          return _usernameNeedError;
        }

        if (!_emailReg.hasMatch(value)) {
          return _usernameFormatError;
        } else {
          return null;
        }
      },
      onChanged: (String value) => _username = value,
      onEditingComplete: () =>
          FocusScope.of(context).requestFocus(_passwordFocusNode),
    );
}

TextFormField buildPasswordTextField(BuildContext context) {
```

```
return TextFormField(
  cursorColor: primaryTextColor,
  controller: _passwordController,
  focusNode: _passwordFocusNode,
  onChanged: (String value) => _password = value,
  onEditingComplete: () => loginSystem(),
  obscureText: _isObscure,
  validator: (String value) {
    if (value.isEmpty) {
      return _passwordNeedError;
    } else {
      return null;
    }
  },
  decoration: InputDecoration(
      labelText: _passwordLabel,
      suffixIcon: IconButton(
        icon: Icon(
          Icons.remove_red_eye,
          color: _eyeColor,
        ),
        onPressed: () {
```

```
                setState(() {
                  _isObscure = !_isObscure;
                  _eyeColor = _isObscure ? primaryTextColor : accentColor;
                });
              },
            )),
    );
}
```

代码 4-3 设置登录按钮与后端登录验证

```
/// 登录到系统方法
void loginSystem() async {
  if (_formKey.currentState.validate()) {
    /// 只有输入的内容符合要求通过才会到达此处
    _formKey.currentState.save();

    setState(() {
      _isLogin = true;
    });

    /// 执行登录方法
    String token;
```

```
if (isDevMode) {
    token = 'thisIsToken';
} else {
    token = await UsersApi.getLoginResponse(_username, _password);
}

if (token != null) {
    /// 登录成功
    Map<String, dynamic> userInfo = {'username': _username, 'token': token};

    /// 保存用户信息
    SharedPreferencesUtils.instance.saveUserInfo(userInfo);
    Navigator.of(context)
        .pushReplacement(MaterialPageRoute(builder: (context) => Home()));
    Fluttertoast.showToast(msg: '欢迎回来，$_username !');
} else {
    Fluttertoast.showToast(msg: _loginErrorText);
    _passwordController.clear();
    setState(() {
      _isLogin = false;
    });
}
```

```
  } else {
    Fluttertoast.showToast(msg: _loginNeedUsernameAndPasswordError);
  }
}
```

代码 4-4 创建登录按钮并监听

```
  /// 监听登录按钮
  /// 如果正在请求登录信息，则禁用按钮
  Function getLoginButtonListener() {
    if (_isLogin) {
      return null;
    } else {
      return () {
        loginSystem();
      };
    }
  }

  /// 登录按钮的创建方法
  Align buildLoginButton(BuildContext context) {
    return Align(
      child: SizedBox(
```

```
            height: 45,
            width: 270,
            child: RaisedButton(
              child: Text(
                _loginText,
                style: TextStyle(color: accentTextColor, fontSize: 20.0),
              ),
              color: accentColorDark,
              splashColor: splashColor,
              disabledColor: disabledColor,
              onPressed: getLoginButtonListener(),
              shape: StadiumBorder(side: BorderSide(color: accentColor)),
            ),
          ),
        );
    }
}
```

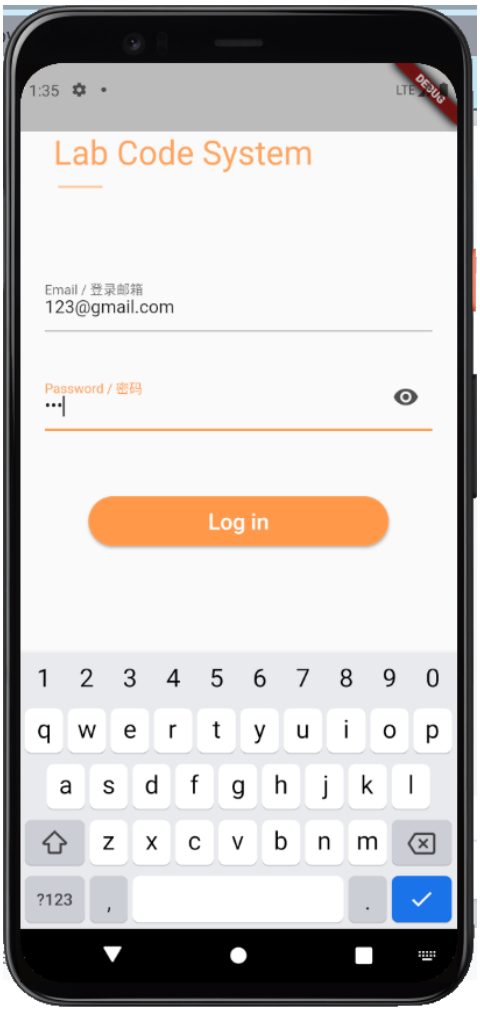

图 4-1 登录页面布局与框架

九、总结及心得体会：

本次实验实现了使用移动端跨平台框架 flutter 实现了应用主页、登录页、整体框架页面的设计开发与编写。

本次对于 flutter 应用界面的编写，发现 flutter 框架本身语言对 UI 界面设计的简易性有所提高，并且 flutter 框架开发应用所使用的 dart 语言既可以用来设计开发移动应用的 UI 界面，又可以用来编写移动应用的逻辑交互。此外，基于 flutter 实现的应用在性能上并不输于原生的安卓或者 IOS 应用，因为它是使 dart 语言直接建立在底层 C/C++层上的，替代了原来的 Java/Swift/Objective-C 的作用。

在 Flutter 框架应用的开发中，万物皆 widget，所以能设置为无状态的组件尽量设置为无状态的组件，因为这可以减少内存消耗，提高界面渲染性能。

十、对本实验过程及方法、手段的改进建议：

Flutter 中的一些主题库并不是在安卓和 IOS 中通用的，所以在跨平台之前，必须要确定是否所用的 UI 组件库是两个平台通用的。

可以在 Home 中将几个没有状态变化的组件设置为 Stateless（无状态）组件以提高移动应用的渲染性能。

报告评分：

指导教师签字：