[](http://pusc.cn/)

《面向对象技术与UML》课程设计

**题 目： 人脸识别登录系统**

**项目成员： 高阳、张法伟、葛其凡**

**专业年级： 软件工程专业2016级**

**指导教师： 景海峰**

**院 系： 基础科学学院信息技术系**

**2018年 10 月 02 日**

**目录**

[第一章 需求分析 3](#_Toc533529545)

[1. 1需求分析 3](#_Toc533529546)

[1.1.1系统角色分析 3](#_Toc533529547)

[1.2 UML用例建模 3](#_Toc533529548)

[1.2.1用例 3](#_Toc533529549)

[1.2.2用例图 4](#_Toc533529550)

[1.2.3用例规约 4](#_Toc533529551)

[1.3用例模型 5](#_Toc533529552)

[1.4分析类 5](#_Toc533529553)

[1.4.1 用户注册模块 5](#_Toc533529554)

[1.4.2用户登录模块 6](#_Toc533529555)

[1.4.3用户管理模块 8](#_Toc533529556)

[1.5功能性需求 9](#_Toc533529557)

[1.5.1 系统用例图 10](#_Toc533529558)

[1.5.2 系统模块介绍 10](#_Toc533529559)

[1.6 非功能型需求 11](#_Toc533529560)

[第二章 概要设计 11](#_Toc533529561)

[2.1系统体系结构设计 11](#_Toc533529562)

[2.2系统架构设计 11](#_Toc533529563)

[2.3系统结构设计 11](#_Toc533529564)

[2.4概念类图设计 12](#_Toc533529565)

[2.5 数据库设计 12](#_Toc533529566)

[2.5.1数据库概念设计 12](#_Toc533529567)

[2.5.2 数据库逻辑设计 13](#_Toc533529568)

[2.6 本章小结 15](#_Toc533529569)

[第三章 系统详细设计及实现 15](#_Toc533529570)

[3.1系统流程图 15](#_Toc533529571)

[3.2用户注册页面实现 16](#_Toc533529572)

[3.2.1网站页面 16](#_Toc533529573)

[3.2.2 人脸识别注册 20](#_Toc533529574)

[3.2.3 账号密码注册 22](#_Toc533529575)

[3.3用户登录界面实现 23](#_Toc533529576)

[3.3.1网站页面 23](#_Toc533529577)

[3.3.2人脸识别登录 24](#_Toc533529578)

[3.3.3 账号密码登录 26](#_Toc533529579)

[3.4 后台管理界面 27](#_Toc533529580)

[3.5退出系统 28](#_Toc533529581)

[3.6本章小结 28](#_Toc533529582)

[第四章 软件测试 28](#_Toc533529583)

[4.1软件测试概述 28](#_Toc533529584)

[4.2 测试方法 29](#_Toc533529585)

[4.3主要测试用例 29](#_Toc533529586)

[4.3.1 登录模块测试用例 29](#_Toc533529587)

[4.4 本章小结 30](#_Toc533529588)

# 第一章 需求分析

本系统主要实现用户登录时账号与人脸识别两种登陆方式，根据个人权限的不同可以修改个人信息或查看他人信息。

1. 1需求分析

人脸识别登录系统需要完成的主要功能有用户用过账号密码或者人脸识别两种方式登录进入系统，用户可以对于系统中的个人信息进行管理，而管理员可以对于所有用户的信息进行管理操作。

1.1.1系统角色分析

小组成员根据甲方的需求，归结总纳出两种角色：用户、管理员。用户可以看到自己的相关信息，用户可以对于自己的部分信息进行新增、查询、修改和删除操作。但对于有些功能用户只能进行查询功能。

管理员可以看到所有用户的相关信息，并可以对于这些信息进行新增、查询、修改流程图如下图所示。

* 1. UML用例建模

1.2.1用例

（1）Use Case: 人脸识别登录

参与者：用户

主事件流：

1.用户选择人脸识别登录选项并将面部移至摄像头前。

2.摄像头采集用户面部信息。

3将面部信息与数据库中的面部信息相匹配。

4若匹配成功，则提示登陆成功，用户登陆到系统内，反之。

（2）Use Case:账号密码登录

参与者：用户

主事件流：

1.用户选择账号登录选项并输入账号密码。

2.将用户输入的user和passwd与数据库中相应的字段进行匹配。

3.若匹配成功，则提示登陆成功，用户登陆到系统内，反之。

（3）Use Case:注册

参与者：用户

1.用户选择注册选项，并输入想要注册的账号密码。

2.将用户输入的user和passwd传入数据库中。

3.。；显示注册成功界面，并跳转至登录界面。

1.2.2用例图

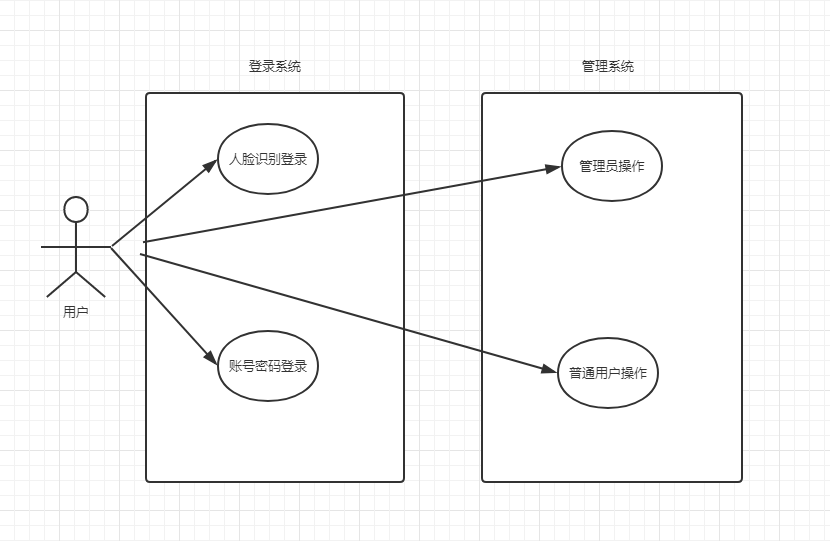


图1-1 总用例图

1.2.3用例规约

1. 用户登陆用例如表1-1：

表1-1用户登录用例

|  |  |
| --- | --- |
| 用例名称： | 用户登录 |
| 角色： | 用户 |
| 用例说明: | 该用例主要实现的功能是用户的登录 |
| 前置条件： | 用户必须输入正确的账号密码或面部无遮挡且完整 |
| 基本事件流： | 输入账号密码或面部识别 |
| 后置条件： | 用户手动选择登录方式 |

1. 用户管理用例如表1-2：

表1-2用户管理用例

|  |  |
| --- | --- |
| 用例名称： | 用户管理 |
| 角色： | 用户 |
| 用例说明: | 该用例主要实现的功能是对个人信息进行增删改查 |
| 前置条件：  基本时间流： | 只可以对自己本人的信息进行操作  用户修改信息后，数据库实时更新 |
| 后置条件： | 修改后保存才可生效 |

1.3用例模型

1.4分析类

1.4.1 用户注册模块

（1）用户注册用例图

如图1-1所示，表示用户注册的用例图

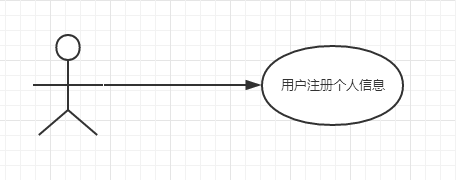


图1-1 用户注册用例图

（2）用户注册时序图

如图1-2，表示用户注册的时序图

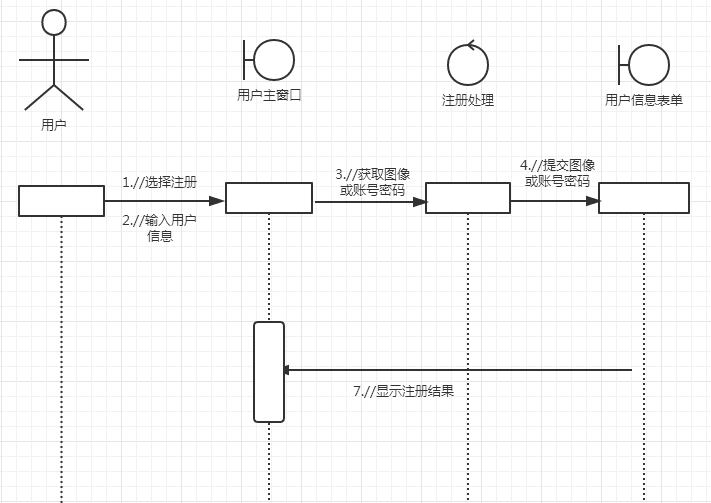


图1-2 用户注册时序图

1. 用户注册协作图

如图1-3，表示用户注册的协作图。

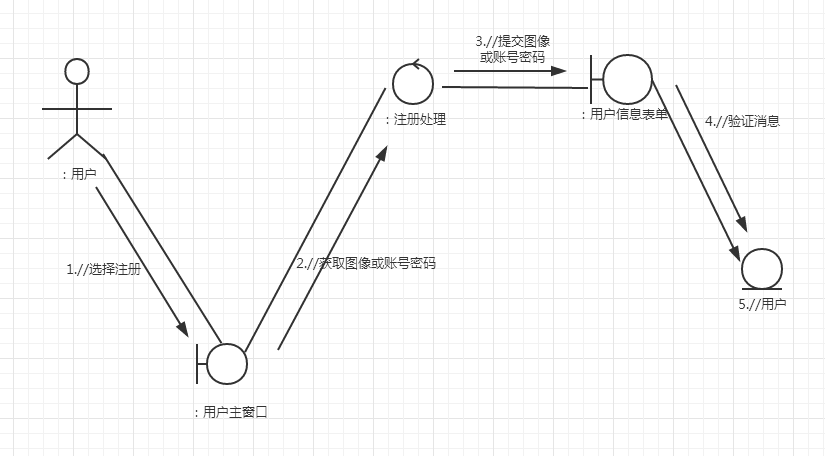


图1-3 用户注册协作图

1.4.2用户登录模块

（1）用户登录用例图

用户登录的用例图，如图1-4所示：

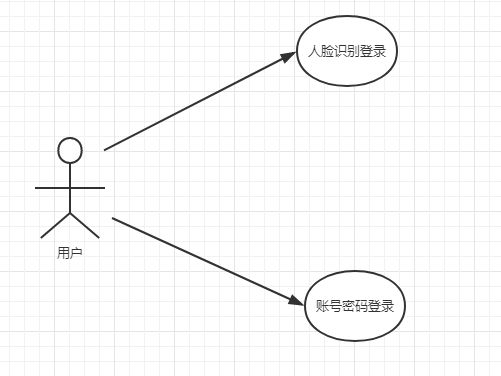


图1-4 用户登录用例图

（2）用户登录时序图

如图1-5所示，表示用户登录的时序图

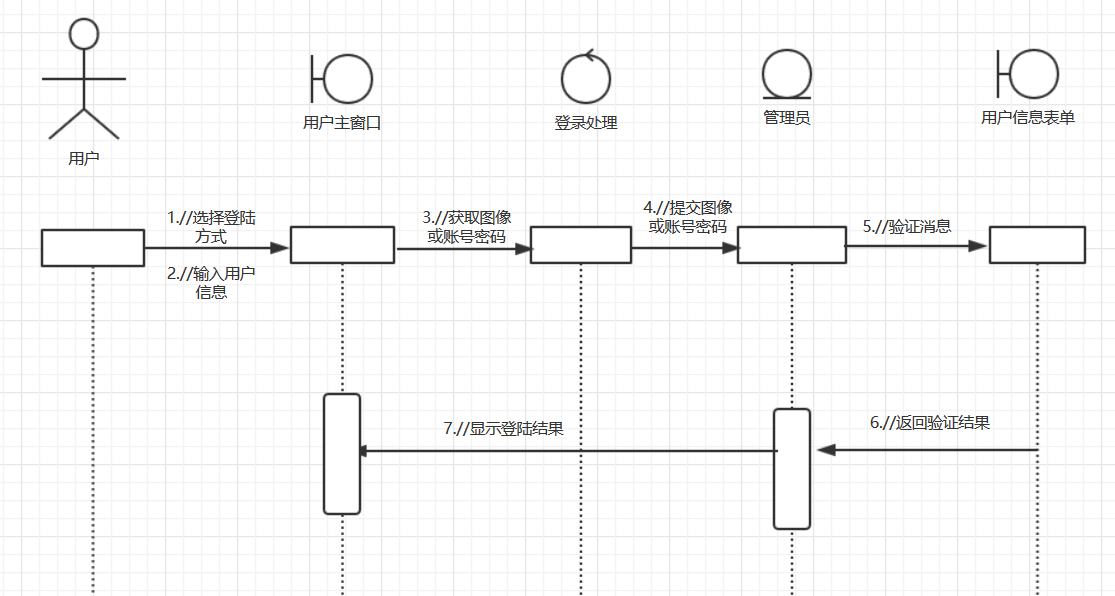


图1-5 用户登录时序图

（3）用户登录分析类图（协作图）

如图1-6所示，表示用户登录的协作图

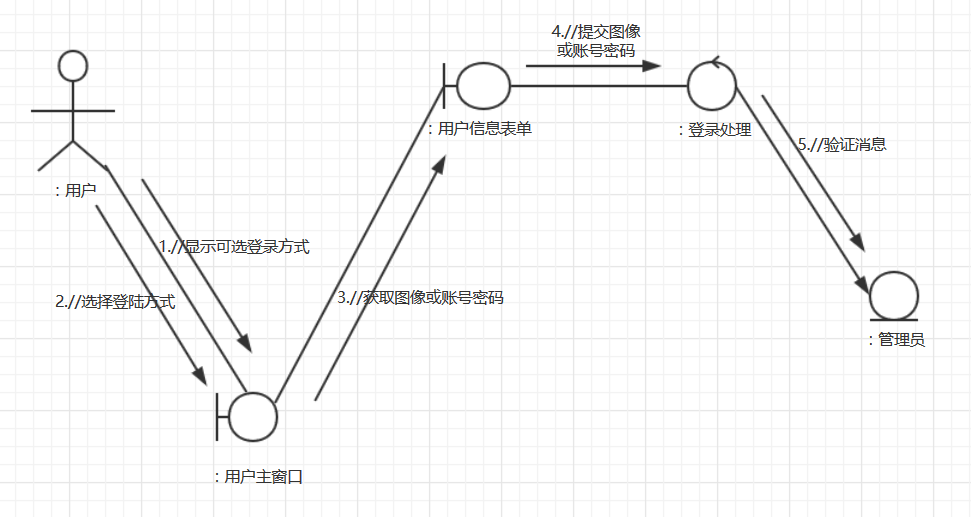


图1-6 用户登录协作图

1.4.3用户管理模块

（1）用户管理用例图

如图1-7所示，表示用户管理的用例图

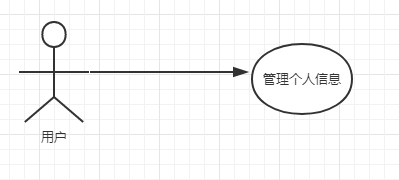


图1-7用户管理个人信息用例图

（2）用户管理时序图

如图1-8所示，表示用户管理的时序图

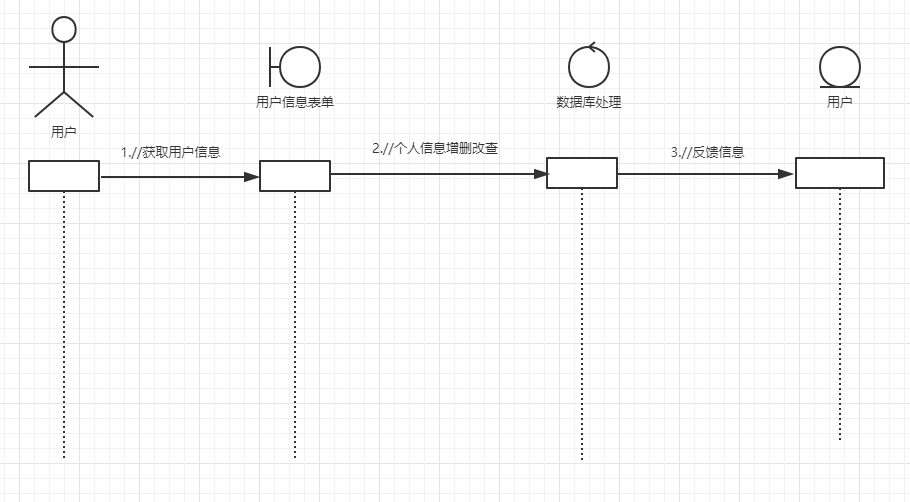


图1-8 用户管理时序图

（3）图片文字提取分析类图（协作图）

如图1-9所示，表示用户管理的协作图

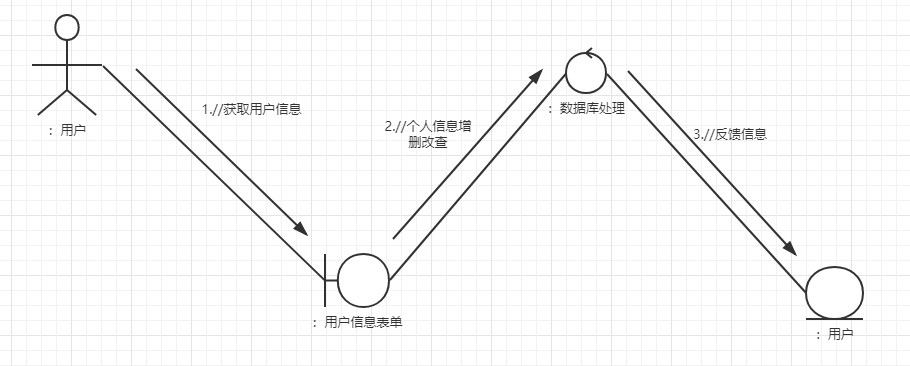


图1-9 用户管理协作图

## 1.5功能性需求

根据系统业务流程分析可知，系统主要可分为用户和管理员两方面的角色，具体可包括用户注册、账号密码登录、人脸识别登录等功能。采用面向对象的方法对整个系统进行功能分析，重点在于人脸识别登录模块。

### 1.5.1 系统用例图

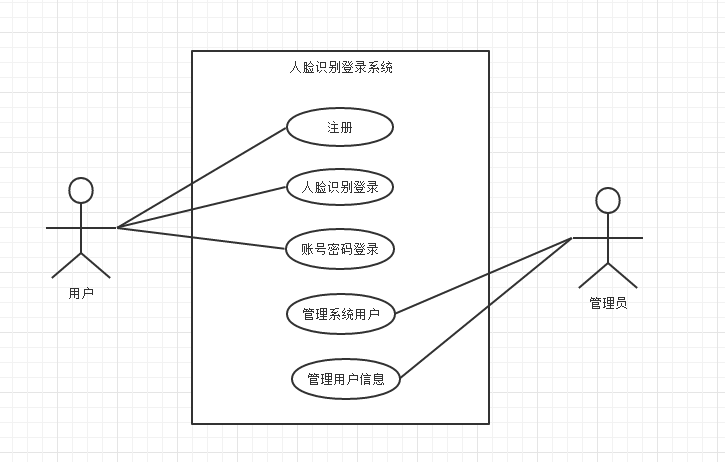
该系统主要是针对管理员和用户两类角色，为教学提供服务系统,如图1-8：

图1-8 系统用例图

### 1.5.2 系统模块介绍

1．用户注册

进入登录界面之后，对于第一次登录的用户来说，首先需要注册，点击去注册链接，就会跳转到注册页面，填写账号密码信息，然后点击在“注册”按钮就可进入注册界面，注册完成后返回登录界面。

用户账号密码登录

进入登录界面之后，用户选择账号密码登录方式，输入账号密码信息，点击登录按钮，就会跳转界面，显示登录成功。

用户人脸识别登录

进入登录界面之后，用户选择人脸识别登录方式，用户看向摄像头，程序就会采集用户 面部的信息并与存储的信息相匹配，如匹配成功，则跳转页面，显示登录成功，若匹配不成功，就会一直识别。

## 1.6 非功能型需求

本系统界面设计简单明晰，用户以易操作，对于用户的信息进行管理和保护，安全性较高，本系统中各个模块之间耦合性较低，模块内部的内聚性较高，故该系统的可维护性较高。并且对于本系统各个功能进行严密的测试，尽量解决潜在的bug问题,提高系统的质量。

# 第二章 概要设计

2.1系统体系结构设计

2.2系统架构设计

系统的总体架构采用B/S模式，前端用的是Bootstrap框架和Jquery技术，后端用Python技术，符合MVC设计模式。

基于B/S模式开发的核心优势就是能够很好的实现分层框架结构，从而提高系统的重用性，灵活性和扩展性。系统分层架构的设计满足了高内聚、低偶合的要求，各层内的逻辑及处理则进行完全的封装，封装体的内部保持紧密的联系，各层之间只需要相互通信即可。

2.3系统结构设计

通过对系统的分析可知，系统主要分为管理员和用户两个角色，系统则主要功能模块设计的依据主要从这两个方面角色的用力模型进行设计，具体可以包括主页面的登录方式，分别是账号密码登录和人脸识别登录两种方式，以及对于后台用户信息的管理，该系统的主要功能结构设计模块设计如下图所示。

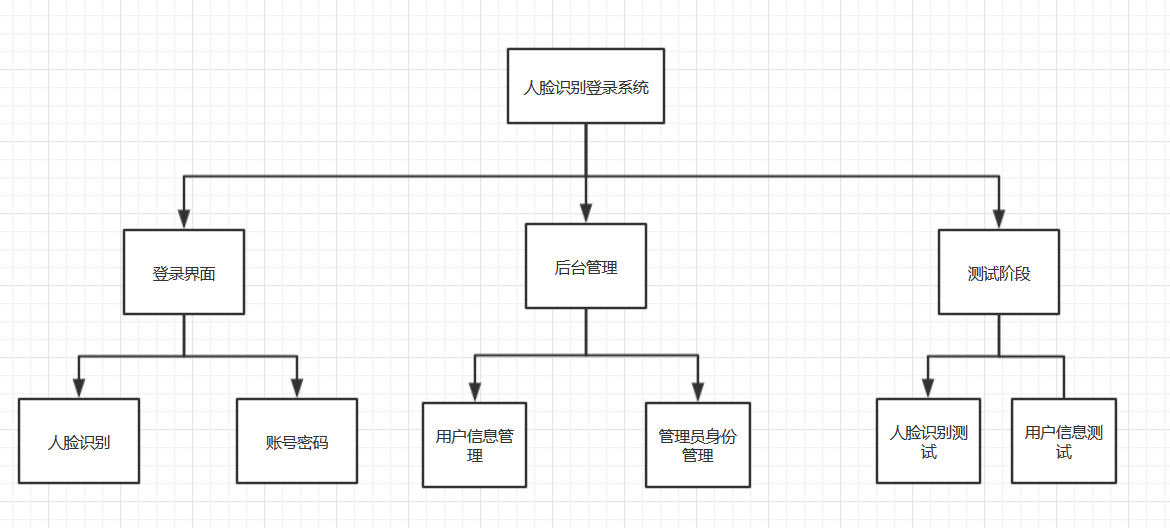


图2-1 主要功能结构设计图

2.4概念类图设计

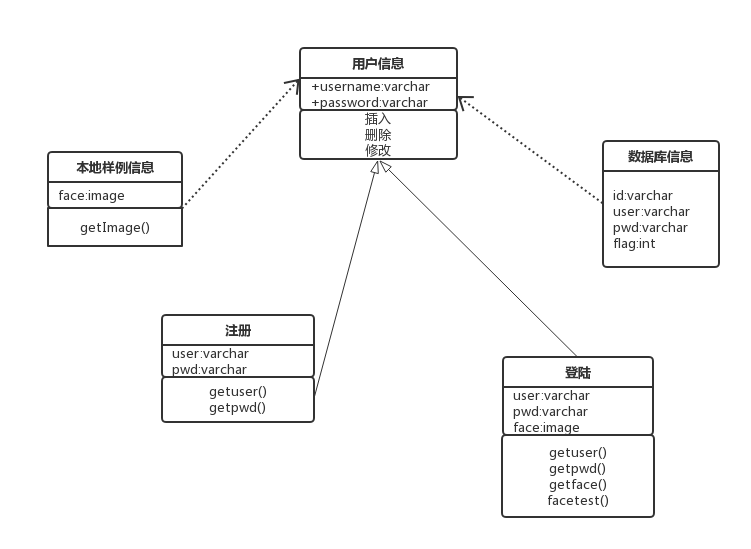


图2-2

2.5 数据库设计

2.5.1数据库概念设计

通过对于系统的需求分析，整套喜用可以设计出个实体，他们分别是本专业实体、非本专业用户实体、管理员实体。

系统的使用者涉及到多种用户，多以需要一个用户表来保存登陆账号和登陆密码。用户表E-R图如图2-3。本专业实体图片信息已被保存，即登陆时即可用数据库中的账号密码登陆，亦可用人脸识别登陆，人脸识别登陆本地人脸被保存到本地，如图2-4

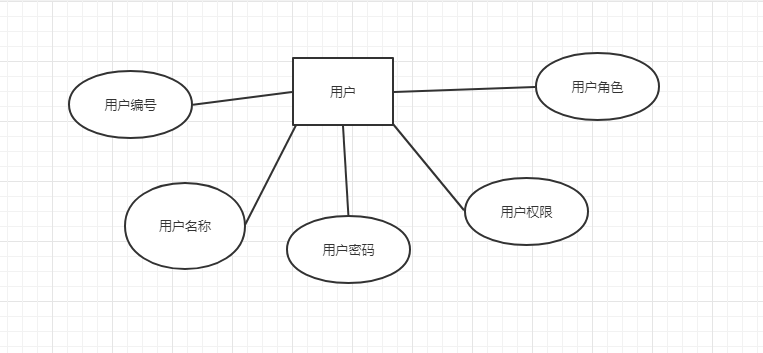


图2-3

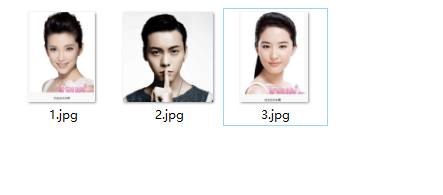


图2-4

2.5.2 数据库逻辑设计

根据系统的需求分析，本系统涉及到的模块需要1张表来存储信息，其中角色信息表如表2-5所示。

表2-5 用户信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 字段长度 | 是否允许为空 | 是否为主键 | 描述 |
| id | int | 11 | 是 | 是 | 用户编号 |
| username | varchar | 150 | 否 | 否 | 用户名称 |
| password | varchar | 128 | 否 | 否 | 用户密码 |
| last\_login | datetime | 6 | 是 | 否 | 用户最后登录时间 |
| ·is\_superuser | tinyint | 1 | 否 | 否 | 用户是否是管理员 |
| first\_name | varchar | 30 | 否 | 否 | 用户姓 |
| last\_name | varchar | 150 | 否 | 否 | 用户名 |

Python Django创建应用后连接数据库后需要迁移数据库，与文件链接，迁移数据库表如图2-6所示。

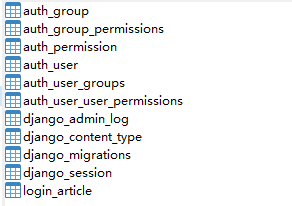


图2-6数据库迁移图

用户注册需输入账号密码，账号重复会提示用户，密码会以MD5加密后保存到数据库中，格式如图2-7所示。

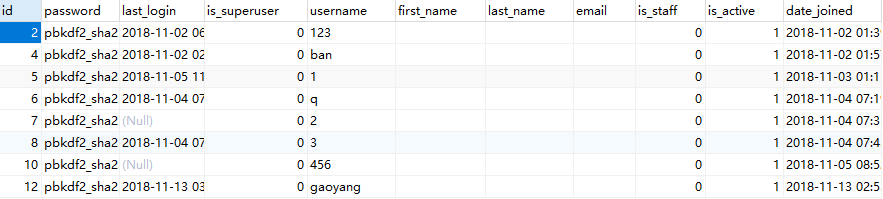


图2-7用户数据库表

2.6 本章小结

本章主要介绍了人脸识别登录系统的系统架构设计，总体功能模块设计以及数据库的设计，系统架构中前端主要是用Bootstrap框架和Jquery技术，后端用了Python Django技术，符合MVC设计模式，数据库从概念设计和逻辑设计两方面进行设计。

# 第三章 系统详细设计及实现

3.1系统流程图

当用户使用本系统时，可以选择登录方式，既可以通过账号密码登录，也可以通过人脸识别登录，在登录的时候，会进行数据比对，如果验证成功，则成功登录，否则重新登录，如下图所示

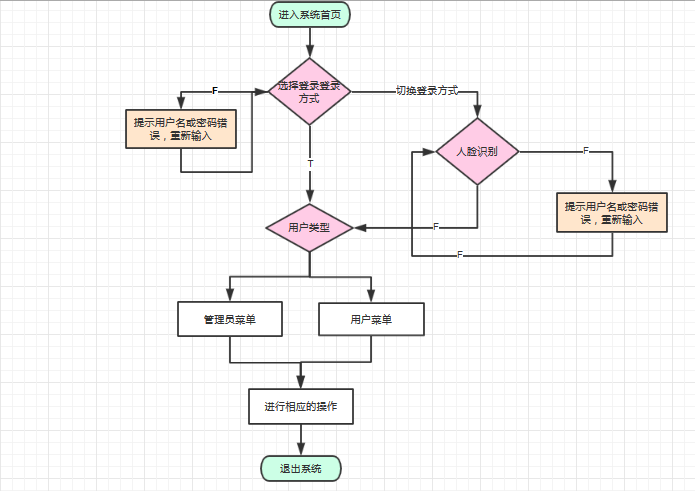


图3-2 系统流程图

3.2用户注册页面实现

3.2.1网站页面

初始页面简介明了如图3-3

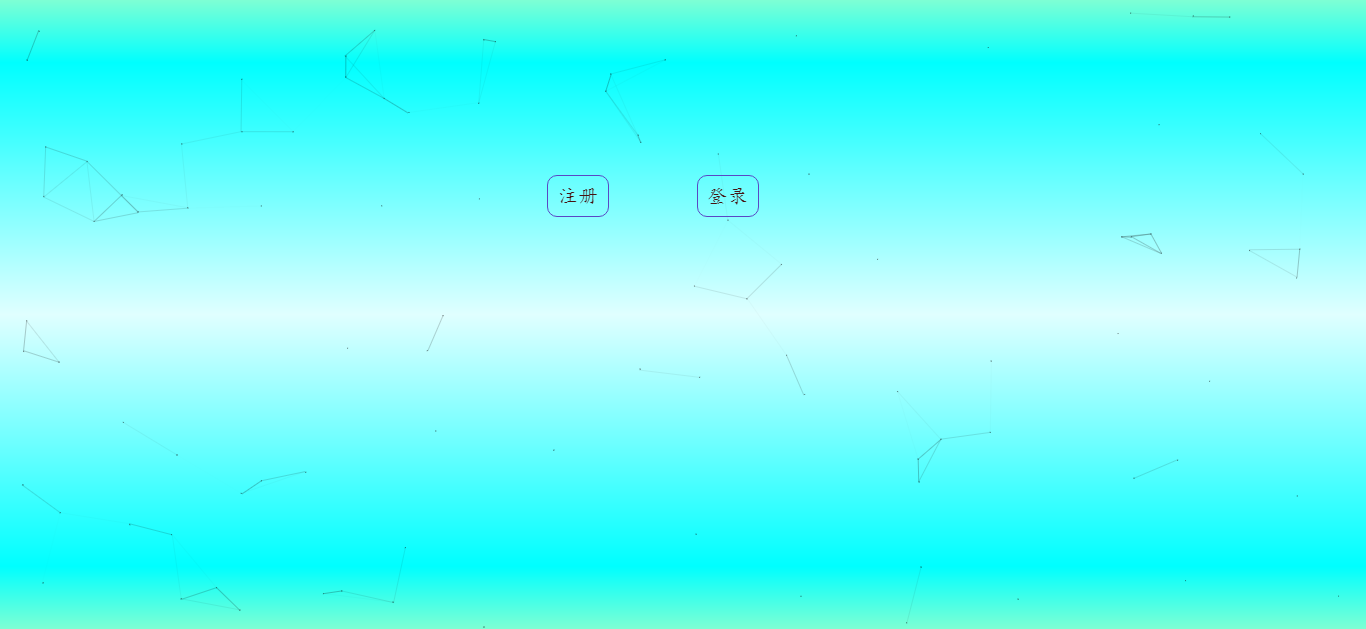


图3-3

代码示例：

<**html lang="en"**>  
<**head**>  
 <**meta charset="utf-8"**>  
 <**style**>  
 **li**,  
 **a** {  
 **text-decoration**: **none**;  
 **list-style**: **none**;  
 **font-family**: **"楷体"**;  
 **font-size**: 35**px**;  
 **color**: **rgb**(71, 18, 18);  
 }  
   
 **body** {  
 **background**: **#ccc**;  
 **background**: **linear-gradient**(**#7fffd4**, **#00ffff** 10%, **#e0ffff** 50%, **#00ffff** 90%, **#7fffd4**);  
 **position**: **relative**;  
 ;  
 }  
   
 .**nav-item** {  
 **width**: 350**px**;  
 **padding**: 10**px**;  
 */\* border: solid 1px rgb(112, 35, 189); \*/* **position**: **absolute**;  
 **top**: 25%;  
 **left**: 34%;  
 }  
   
 .**nav-item a** {  
 **font-size**: 20**px**;  
 **border**: **rgb**(90, 71, 197) **solid** 1**px**;  
 **padding**: 10**px**;  
 **border-radius**: 10**px**;  
 **margin-left**: 70**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<**div id="layout" class="pure-g"**>  
 <**div class="yoyoyo"**>  
 <**div class="header"**>  
 <**nav class="nav"**>  
 <**ul class="nav-list"**>  
 <**li class="nav-item"**>  
 {% **if user**.**is\_authenticated** %}  
 <**a class="button-success pure-button" href="/index/"**>{{ **user**.**username** }}</**a**>  
 <**a class="button-success pure-button" href="/logout/"**>退出</**a**>  
 {% **else** %}  
 <**a class="button-success pure-button" href="/regist/"**>注册</**a**>  
 <**a class="button-success pure-button" href="/login1/"**>登录</**a**>  
 {% **endif** %}  
 </**li**>  
 </**ul**>  
 </**nav**>  
 </**div**>  
 </**div**>  
 <**div class="content pure-u-1 pure-u-md-3-4"**>  
 <**div**>  
 {% **block content** %}  
 {% **endblock** %}  
 </**div**>  
 </**div**>  
</**div**>  
</**body**>  
<**script**>  
 ! **function**() {  
 **function** *n*(n, e, t) {  
 **return** n.getAttribute(e) || t  
 }  
  
 **function** *e*(n) {  
 **return *document***.getElementsByTagName(n)  
 }  
  
 **function** *t*() {  
 **var** t = *e*(**"script"**),  
 o = t.**length**,  
 i = t[o - 1];  
 **return** {  
 **l**: o,  
 **z**: *n*(i, **"zIndex"**, -1),  
 **o**: *n*(i, **"opacity"**, .5),  
 **c**: *n*(i, **"color"**, **"0,0,0"**),  
 **n**: *n*(i, **"count"**, 99)  
 }  
 }  
  
 **function** *o*() {  
 a = m.width = ***window***.innerWidth || ***document***.documentElement.clientWidth || ***document***.body.clientWidth, c = m.height = ***window***.innerHeight || ***document***.documentElement.clientHeight || ***document***.body.clientHeight  
 }  
  
 **function** *i*() {  
 r.clearRect(0, 0, a, c);  
 **var** n, e, t, o, m, l;  
 s.forEach(**function**(i, x) {  
 **for** (i.**x** += i.**xa**, i.**y** += i.**ya**, i.**xa** \*= i.**x** > a || i.**x** < 0 ? -1 : 1, i.**ya** \*= i.**y** > c || i.**y** < 0 ? -1 : 1, r.fillRect(i.**x** - .5, i.**y** - .5, 1, 1), e = x + 1; e < u.length; e++) n = u[e], **null** !== n.**x** && **null** !== n.**y** && (o = i.**x** - n.**x**, m = i.**y** - n.**y**, l = o \* o + m \* m, l < n.**max** && (n === y && l >= n.**max** / 2 && (i.**x** -= .03 \* o, i.**y** -= .03 \* m), t = (n.**max** - l) / n.**max**, r.beginPath(), r.lineWidth = t / 2, r.strokeStyle = **"rgba("** + d.**c** + **","** + (t + .2) + **")"**, r.moveTo(i.**x**, i.**y**), r.lineTo(n.**x**, n.**y**), r.stroke()))  
 }), x(*i*)  
 }  
 **var** a, c, u, m = ***document***.createElement(**"canvas"**),  
 d = *t*(),  
 l = **"c\_n"** + d.l,  
 r = m.getContext(**"2d"**),  
 x = ***window***.requestAnimationFrame || ***window***.webkitRequestAnimationFrame || ***window***.mozRequestAnimationFrame || ***window***.oRequestAnimationFrame || ***window***.msRequestAnimationFrame || **function**(n) {  
 ***window***.setTimeout(n, 1e3 / 45)  
 },  
 w = ***Math***.random,  
 y = {  
 **x**: **null**,  
 **y**: **null**,  
 **max**: 2e4  
 };  
 m.id = l, m.style.cssText = **"position:fixed;top:0;left:0;z-index:"** + d.**z** + **";opacity:"** + d.**o**, *e*(**"body"**)[0].appendChild(m), *o*(), ***window***.onresize = *o*, ***window***.onmousemove = **function**(n) {  
 n = n || ***window***.event, y.**x** = n.clientX, y.**y** = n.clientY  
 }, ***window***.onmouseout = **function**() {  
 y.**x** = **null**, y.**y** = **null** };  
 **for** (**var** s = [], f = 0; d.**n** > f; f++) {  
 **var** h = w() \* a,  
 g = w() \* c,  
 v = 2 \* w() - 1,  
 p = 2 \* w() - 1;  
 s.push({  
 **x**: h,  
 **y**: g,  
 **xa**: v,  
 **ya**: p,  
 **max**: 6e3  
 })  
 }  
 u = s.concat([y]), *setTimeout*(**function**() {  
 *i*()  
 }, 100)  
 }();  
</**script**>  
</**html**>

3.2.2 人脸识别注册

注册界面清晰简洁，易于操作，注册界面如图3-4：



图3-4

代码示例：

**def** loadImage(filePath):  
  
 inputImg = ASVLOFFSCREEN()  
 **if** bUseBGRToEngine:  
 bufferInfo = ImageLoader.getBGRFromFile(filePath)  
 inputImg.u32PixelArrayFormat = ASVL\_COLOR\_FORMAT.ASVL\_PAF\_RGB24\_B8G8R8  
 inputImg.i32Width = bufferInfo.width  
 inputImg.i32Height = bufferInfo.height  
 inputImg.pi32Pitch[0] = bufferInfo.width\*3  
 inputImg.ppu8Plane[0] = cast(bufferInfo.buffer, c\_ubyte\_p)  
 inputImg.ppu8Plane[1] = cast(0, c\_ubyte\_p)  
 inputImg.ppu8Plane[2] = cast(0, c\_ubyte\_p)  
 inputImg.ppu8Plane[3] = cast(0, c\_ubyte\_p)  
 **else**:  
 bufferInfo = ImageLoader.getI420FromFile(filePath)  
 inputImg.u32PixelArrayFormat = ASVL\_COLOR\_FORMAT.ASVL\_PAF\_I420  
 inputImg.i32Width = bufferInfo.width  
 inputImg.i32Height = bufferInfo.height  
 inputImg.pi32Pitch[0] = inputImg.i32Width  
 inputImg.pi32Pitch[1] = inputImg.i32Width // 2  
 inputImg.pi32Pitch[2] = inputImg.i32Width // 2  
 inputImg.ppu8Plane[0] = cast(bufferInfo.buffer, c\_ubyte\_p)  
 inputImg.ppu8Plane[1] = cast(addressof(inputImg.ppu8Plane[0].contents) + (inputImg.pi32Pitch[0] \* inputImg.i32Height), c\_ubyte\_p)  
 inputImg.ppu8Plane[2] = cast(addressof(inputImg.ppu8Plane[1].contents) + (inputImg.pi32Pitch[1] \* inputImg.i32Height // 2), c\_ubyte\_p)  
 inputImg.ppu8Plane[3] = cast(0, c\_ubyte\_p)  
 inputImg.gc\_ppu8Plane0 = bufferInfo.buffer  
  
 **return** inputImg

*# init Engine*pFDWorkMem = CLibrary.malloc(c\_size\_t(FD\_WORKBUF\_SIZE))  
pFRWorkMem = CLibrary.malloc(c\_size\_t(FR\_WORKBUF\_SIZE))  
  
hFDEngine = c\_void\_p()  
ret = AFD\_FSDK\_InitialFaceEngine(APPID, FD\_SDKKEY, pFDWorkMem, c\_int32(FD\_WORKBUF\_SIZE), byref(hFDEngine),  
 AFD\_FSDK\_OPF\_0\_HIGHER\_EXT, 32, MAX\_FACE\_NUM)  
**if** ret != 0:  
 CLibrary.free(pFDWorkMem)  
 exit(0)  
  
  
versionFD = AFD\_FSDK\_GetVersion(hFDEngine)  
  
  
hFREngine = c\_void\_p()  
ret = AFR\_FSDK\_InitialEngine(APPID, FR\_SDKKEY, pFRWorkMem, c\_int32(FR\_WORKBUF\_SIZE), byref(hFREngine))  
**if** ret != 0:  
 AFD\_FSDKLibrary.AFD\_FSDK\_UninitialFaceEngine(hFDEngine)  
 CLibrary.free(pFDWorkMem)  
 CLibrary.free(pFRWorkMem)  
 exit(0)  
  
*# print FREngine version*versionFR = AFR\_FSDK\_GetVersion(hFREngine)  
  
*# load Image Data***if** bUseYUVFile:  
 filePathA = **u'001\_640x480\_I420.YUV'** yuv\_widthA = 640  
 yuv\_heightA = 480  
 yuv\_formatA = ASVL\_COLOR\_FORMAT.ASVL\_PAF\_I420  
  
 filePathB = **u'003\_640x480\_I420.YUV'** yuv\_widthB = 640  
 yuv\_heightB = 480  
 yuv\_formatB = ASVL\_COLOR\_FORMAT.ASVL\_PAF\_I420  
  
 inputImgA = loadYUVImage(filePathA, yuv\_widthA, yuv\_heightA, yuv\_formatA)  
 inputImgB = loadYUVImage(filePathB, yuv\_widthB, yuv\_heightB, yuv\_formatB)  
 inputImgC = loadYUVImage(filePathC, yuv\_widthA, yuv\_heightA, yuv\_formatA)  
 inputImgD = loadYUVImage(filePathD, yuv\_widthB, yuv\_heightB, yuv\_formatB)  
**else**:  
 filePathA = path0  
 filePathB = path1  
  
 inputImgA = loadImage(filePathA)  
 inputImgB = loadImage(filePathB)  
 inputImgC = loadImage(filePathC)  
 inputImgD = loadImage(filePathD)  
  
res=str(compareFaceSimilarity(hFDEngine, hFREngine, inputImgA, inputImgB))[8:11]  
**return** float(res)  
  
*# release Engine*AFD\_FSDK\_UninitialFaceEngine(hFDEngine)  
AFR\_FSDK\_UninitialEngine(hFREngine)  
  
CLibrary.free(pFDWorkMem)  
CLibrary.free(pFRWorkMem)

3.2.3 账号密码注册

清晰简洁的注册界面，用户依次输入账号、密码等信息进行注册，将表单的数据存入数据库，实现用户的注册功能。

代码示例：

**def** regist(request):  
 **if** request.method == **'POST'**:  
 uf = test(request.POST)  
 **if** uf.is\_valid():  
 username = uf.cleaned\_data[**'username'**]  
 password = uf.cleaned\_data[**'password'**]  
 registAdd = User.objects.create\_user(username=username, password=password)  
 **if** registAdd == **False**:  
 **return** render(request, **'share1.html'**, {**'registAdd'**: registAdd, **'username'**: username})  
  
 **else**:  
 **return** render(request, **'share1.html'**, {**'registAdd'**: registAdd})  
 **else**:  
 uf = test()  
 **return** render(request, **'regist1.html'**, {**'uf'**: uf})

3.3用户登录界面实现

3.3.1网站页面

初始页面简洁，如图3-5

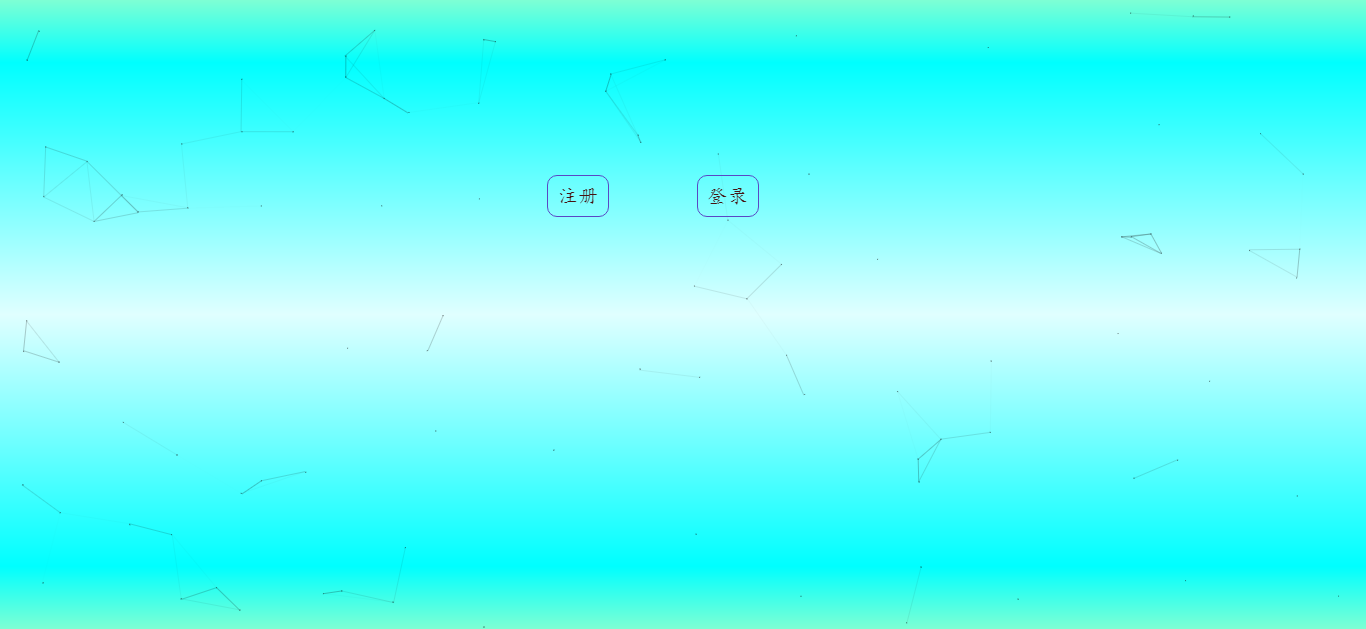


图3-5

代码示例：

<**div id="layout" class="pure-g"**>  
 <**div class="yoyoyo"**>  
 <**div class="header"**>  
 <**nav class="nav"**>  
 <**ul class="nav-list"**>  
 <**li class="nav-item"**>  
 {% **if user**.**is\_authenticated** %}  
 <**a class="button-success pure-button" href="/index/"**>{{ **user**.**username** }}</**a**>  
 <**a class="button-success pure-button" href="/logout/"**>退出</**a**>  
 {% **else** %}  
 <**a class="button-success pure-button" href="/regist/"**>注册</**a**>  
 <**a class="button-success pure-button" href="/login1/"**>登录</**a**>  
 {% **endif** %}  
 </**li**>  
 </**ul**>  
 </**nav**>  
 </**div**>  
 </**div**>

3.3.2人脸识别登录

清晰简洁的登录界面，可以让人心旷神怡，系统的界面在于设计的时候，着眼于用户的需求，登录界面如图3-6所示



图3-6

代码示例：

<**script src="/static/js/app.js"**></**script**>  
<**script**>  
 **var *video*** = ***document***.getElementById(**'video'**);  
 **var *canvas*** = ***document***.getElementById(**'canvas'**);  
 **var *context*** = ***canvas***.getContext(**'2d'**);  
 **function** *getUserMediaToPhoto*(constraints,success,error) {  
 **if**(***navigator***.mediaDevices.getUserMedia){  
 ***navigator***.mediaDevices.getUserMedia(constraints).then(success).catch(error);  
 }**else if** (***navigator***.webkitGetUserMedia) {  
 ***navigator***.**webkitGetUserMedia**(constraints,success,error);  
 }**else if**(***navigator***.mozGetUserMedia){  
 ***navigator***.**mozGetUserMedia**(constraints,success,error);  
 }**else if**(***navigator***.getUserMedia){  
 ***navigator***.getUserMedia(constraints,success,error);  
 }  
 }  
 **function** *success*(stream){  
 **var** CompatibleURL = ***window***.URL || ***window***.webkitURL;  
 ***video***.**src** = CompatibleURL.*createObjectURL*(stream);  
 ***video***.play();  
 *postFace*()  
 }  
 **function** *error*(error) {  
 ***console***.log(**'访问用户媒体失败：'**,error.**name**,error.**message**);  
 }  
 **function** *postFace*() {  
 *setTimeout*(**function** () {  
 ***context***.drawImage(***video***,0,0,480,320);  
 **img**=***canvas***.toDataURL(**'image/jpg'**)  
 **img**=**img**.split(**','**)[1]  
 **$**.post({  
 **url**:**'/getface'**,  
 **data**:{  
 **message**:**img** },  
 success:**function** (callback) {  
 **if**(callback==**'no'**){  
 *postFace*()  
 }**else** {  
 ***window***.location.href=callback  
 }  
 },  
 error:**function** (callback) {  
 *postFace*()  
 }  
 })  
 },300)  
 }  
 **if**(***navigator***.mediaDevices.getUserMedia || ***navigator***.webkitGetUserMedia || ***navigator***.mozGetUserMedia || ***navigator***.getUserMedia){  
 *getUserMediaToPhoto*({**video**:{**width**:480,**height**:320}},*success*,*error*);  
 }**else**{  
 *alert*(**'你的浏览器不支持访问用户媒体设备'**);  
 }

3.3.3 账号密码登录

清晰简洁的登录界面，用户依次输入账号、密码等信息进行登录，将表单的数据与数据库中的数据进行校对，实现用户的登录功能，注册界面如下图3-7所示。



图3-7

代码示例：

**def** login1(request):  
 **if** request.method == **'POST'**:  
 username = request.POST.get(**'username'**)  
 password = request.POST.get(**'password'**)  
 re = auth.authenticate(username=username, password=password)  
 **if** re **is not None**:  
 auth.login(request, re)  
 **return** redirect(**'/'**, {**'user'**: re})  
 **else**:  
 **return** render(request, **'login1.html'**, {**'login\_error'**: **'用户名或密码错误'**})  
 **return** render(request, **'login1.html'**)

3.4 后台管理界面

和删除操作。管理员通过对于用户信息的管理，使得用户的信息合理协调的呈现给用户，后台管理的管理员登录到系统，管理员可以看到所有用户的相关信息，并可以对于这些信息进行新增、查询、修改流程图如下图3-8所示。

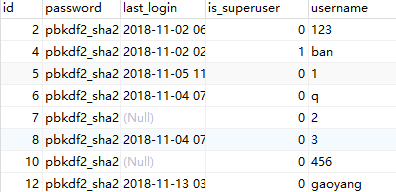


图3-8

3.5退出系统

如果用户在完成相应的操作之后，要退出当前的当前的系统，可以点击右上角的“退出”按钮选项，可以安全的退出管理系统。

代码示例：

**def** logout(request):  
 auth.logout(request)  
 **return** render(request, **'index.html'**)

3.6本章小结

本章主要介绍了人脸识别登录系统的详细设计和系统实现，功能包括系统流程与的世界，用户注册功能的实现，以及进行系统管理等操作，用户在使用完系统后可以点击按钮退出当前系统。

# 第四章 软件测试

4.1软件测试概述

在开发软件的过程中，人们使用了许多保证软件质量的方法分析、设计和实现软件，但难免在工作中犯错误。这样，在软件产品中就会隐藏许多错误和缺陷。对于规模大、复杂性高的软件更是如此。在这些错误中，有些是致命的错误，如果不排除，就会导致财产的重大损失。这种情况迫使人们必须认真计划、彻底的进行软件测试。

为了保证软件的质量和可靠性，应力求在分析、设计等各个开发阶段结束前，对软件进行严格的技术评审。但由于人们能力的局限性，审查不能发现所有的错误。而且在编码阶段还会引进大量的错误。

软件测试就是在软件投入运行前，对软件需求分析、设计规格说明和编码的最终审查，是软件质量保证的关键步骤。如果给软件测试下定义，可以这样讲：软件测试是为了发现错误而执行程序的过程。或者说，软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计一批测试用例（即输入数据和预期的结果），并利用这些测试用例去运行程序，以发现错误的过程。

4.2 测试方法

黑盒测试Black-box Testing，又称功能测试、数据驱动测试或基于规格说明的测试，是一种从用户观点出发的测试。被测程序被当作一个黑盒，不考虑程序内容结构和内容特性，测试着只知道该程序输入和输出之间的关系或程序的功能，依靠能够反映这一关系的程序功能的需求规格说明书考虑确定测试用例盒推断测试结果的正确性。软件的黑盒测试被用来证明软件功能的正确性盒可操作性。

白盒测试White-box Testing，又称结构测试、逻辑驱动测试或基于程序的测试。白盒测试依赖于对程序细节的严密检查，针对特定条件和循环集设计测试用例，对软件的逻辑路径进行测试。在程序的不同点检验“程序状态”，一盘定期实际情况是否和预期的状态相一致。软件的白盒测试用来分析程序的内部结构。白盒测试要求对某些程序的结构特性做到一定程度的覆盖，或者说是“基于覆盖的测试”。

4.3主要测试用例

4.3.1 登录模块测试用例

表4-1 登录模块测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 输入数据 | 期望结果 | 实际结果 | 结果 |
| 角色：普通用户 | 提示“用户名或密码错误” | 提示“用户名或密码” 错误 | 通过 |
| 账号：123123 |  |  |  |
| 密码：123123 |  |  |  |
| 角色：本专业学生  密码登陆  账号：1  密码：1 | 登录成功 | 登录成功 | 通过 |
| 角色：本专业学生  密码登陆  账号：aaa  密码：123 | 提示“用户名或密码错误” | 提示“用户名或密码错误” | 通过 |
| 账号：123123 |  |  |  |
| 密码：123123 |  |  |  |
| 角色：本专业学生  人脸登陆 | 登陆成功 | 登陆成功 | 通过 |
|  |  |  |

4.4 本章小结

本章主要介绍了人脸识别系统的系统测试部分，通过对登录和注册模块，航班信息添加、查询、删除、修改操作进行测试。测试结果符合预期结果，功能一切正常。