目录

图表目录

引言

近几年来，考证成为了大学生的热门话题。面对日益严峻的就业形式，越来越多的大学生通过考证来增加自身的砝码，“考证热”浪潮经久不衰[1]。但是在大学生群体中存在的盲目跟风、一味追求证书数量等现象使得很多人费时费力考取的证书不仅花费了大量时间成本，而且也没有实现预期的价值。我国现有的考证相关系统[2-3]虽然得到了一定的发展，但大多内容简单，侧重的方向主要为考证信息的发布以及考证相关培训机构的宣传等，并不能解决大学生目前最亟需解决的问题。因此，考证系统应当利用何种方式帮助大学生高效备考以及如何减少大学生盲目考证是解决大学生考证问题的关键所在。

针对上述问题，我们设计并开发了大学生考证信息咨询与决策系统，该系统为大学生提供“考证一对一”的个性化服务，即提供相对全面的考证信息咨询、考证经验交流、考证真题下载等功能来帮助其获取有效信息。本系统还采用证书参数比对与用户问卷评估的方式来减少大学生考证的盲目性：通过调查大学生的考证目的与考证需求来评定证书比对的关键参数以及通过对大学生多种考证因素的分析来建立标准化评定模型以辅助大学生做出相对正确的考证决策。大学生考证咨询与决策系统不仅能极大程度地帮助大学生做好考证准备，而且该系统的使用及维护成本低，能更好的节约人力物力，为广大有考证需求的大学生服务。

系统详细设计与系统代码实现是系统开发过程中的重要环节。系统详细设计与系统总体设计结合，并称为系统设计。系统设计是系统开发的第二阶段，主要是为了解决“怎么做”的问题[4]。系统详细设计是以系统总体设计为依据，对系统的编码、输入输出、界面及系统功能模块进行设计，为系统实现提供详细设计依据。本组的详细设计部分省略了输入输出设计，由于本系统操作简单，输入方式仅为键盘输入；输出方式仅为报表输出，故省略该部分。系统代码实现是系统实施的关键环节，是以系统设计文档与数据库文档为依据通过编程实现系统各功能模块并通过测试不断完善系统内容的过程。本组在开发中参考了大量与本系统相似的现有系统[5-7]以及系统实现的相关书籍[8-11]，通过分析现有系统的优缺点、开发方法与开发策略，本组决定采用 B\S 结构、MVC 架构，并利用 Python、Django、Nginx、MySQL 等技术进行开发。

本组设计并开发的 PC 端大学生考证信息咨询与决策系统覆盖人群为有证书管理、证书查询以及证书决策方面需求的大学生，提供包括证书查询、考证分享展示、个人空间管理、考证决策查询等服务，从而达到有效帮助大学生科学备考、辅助大学生进行考证决策的目的。

1 PC 端系统详细设计

1.1 编码设计

本系统的编码设计主要有三个方面，分别是证书编码、文章编码及问卷编码。证书编码是对用户上传的证书图片信息进行管理和查找时使用的编码；文章编码是对用户发布的文章进行管理和查找时使用的编码；问卷编码是对用户考证评估问卷的各选项进行编码，以便能利用此编码快速对应问卷分值进而得出用户问卷评分。

1.1.1 证书编码

证书编码是对用户上传的证书图片信息进行编码以便对其进行管理、查询。采用数字码形式编写，共由十位数字组成。其中，前四位表示用户 ID，中间四位表示证书年号，后两位表示证书序号（即某用户发布证书图片的证书序号 ID）。证书编码设计如图 1-1 所示。

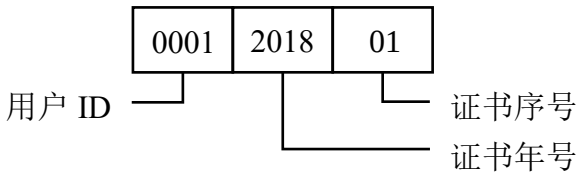


图 1-1 证书编码设计图

1.1.2 文章编码

文章编码是对用户发布的文章信息进行编码以便对其进行管理、查询。采用数字码形式编写，共由八位数字组成。其中，前四位表示用户 ID，中间两位表示栏目序号（即某用户添加文章栏目的栏目序号 ID），后两位表示文章序号（即某用户发布文章的文章序号 ID）。文章编码设计如图 1-2 所示。

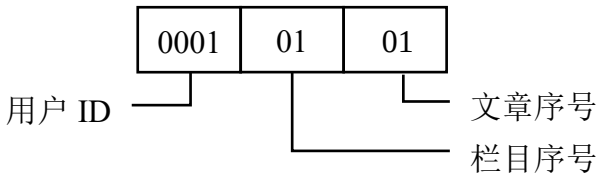


图 1-2 文章编码设计图

1.1.3 问卷编码

问卷编码是对用户考证评估问卷各选项进行编码以便能利用此编码快速对应问卷分值进而得出用户问卷评分。采用混合码形式编写，共由四位字符组成。其中，前两位表示问卷的问题序号（问卷中的问题序号 ID），中间一位表示用户将要决策的证书类型：1 为基础类证书、2 为职业类证书、3 为能力类证书，最后一位表示选项字符（若选择 A 选项，记作 A）。问卷编码设计如图 1-3 所示。

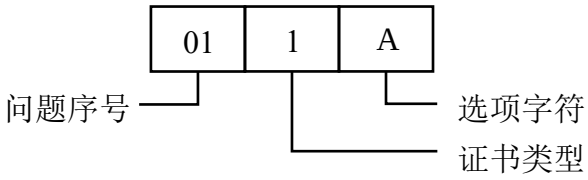


图 1-3 问卷编码设计图

1.2 用户界面设计

本系统的用户界面设计主要有六个方面，分别是用户登录界面、证书列表界面、分享展示界面、个人空间管理界面、决策查询界面、管理员管理界面。其中，用户登录界面保证用户顺利登录主页；证书列表界面允许用户通过浏览、查询的方式获取证书详情；分享展示界面整理并展示各用户发布的分享信息，供用户互动交流；个人空间管理界面使用户能够自主管理个人信息及发布信息；决策查询界面通过 VS 决策及个性决策的方式辅助用户做出考证决策；管理员管理界面帮助管理员实现管理用户、修改权限、查看反馈等功能。

1.2.1 用户登录界面

用户登录界面分为普通用户登录界面和超级用户（以下简称管理员）登录界面。普通用户和管理员须登录后方可使用系统功能，以此增加系统信息的安全性。为最大限度的方便使用人群，本系统的登录项只有用户名和密码，正确输入后即可进入系统主页，反之，系统提示相应的错误信息。

在普通用户登录界面中，除登录表单外还提供用户注册、忘记密码的链接，进一步方便普通用户操作。登录界面的异常提示预计以红色字样显示于表单内而不以弹窗提示，这样在操作上更加简单直观。由于本系统的使用者为大学生人群，因此普通用户相关界面的设计为青春、阳光且简约时尚的风格，使用户能够在轻松愉快的氛围下使用本系统。普通用户登录界面设计如图 1-4 所示。

图 1-4 普通用户登录界面设计

在管理员登录界面中，与用户登录类似，只需输入用户名和密码方可登录后台管理界面进行相关操作。由于管理员的主要任务为审核、修改、查阅等，因此，本组认为管理员相关界面应为简约、整洁的风格。管理员登录界面设计如图 1-5 所示。

图 1-5 管理员登录界面设计

1.2.2 证书列表界面

证书列表界面是本系统的主页，此界面主要由导航、证书列表和搜索框组成。导航展示系统其余功能；证书列表对分类证书（职业类、能力类、基础类）进行分类展示，用户可点击感兴趣的证书以查看其证书详情；搜索框允许用户通过输入证书名称或证书英文简称的方式来查看证书详情。证书列表界面设计如图 1-6 所示。

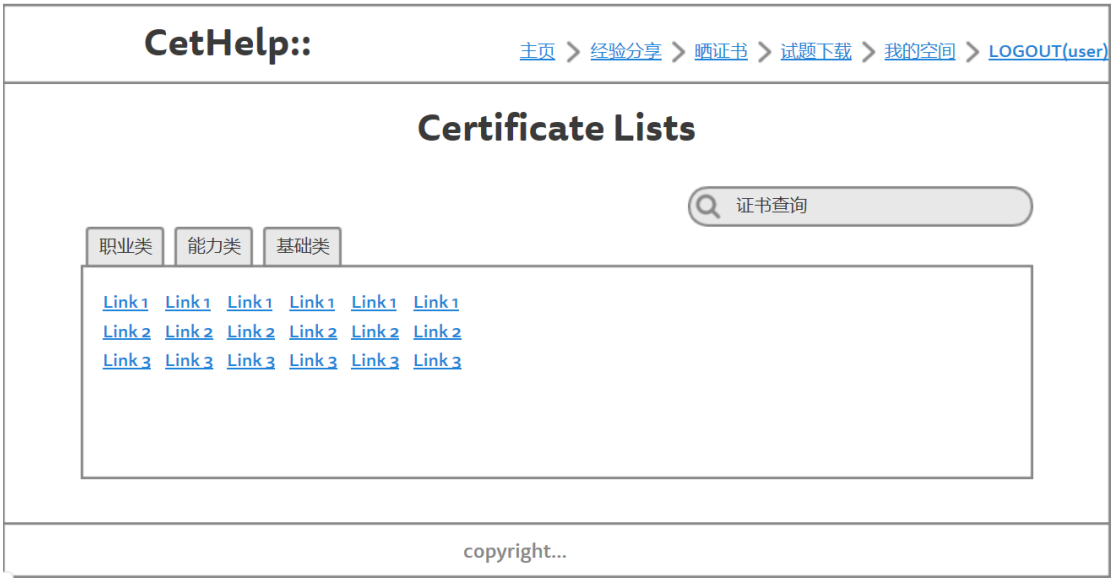


图 1-6 证书列表界面设计

1.2.3 分享展示界面

分享展示界面分为文章列表界面、证书图文展示界面、试题下载界面，分别对应主页导航中的“经验分享”、“晒证书”和“试题下载”。分享展示界面是将用户发布的分享信息进行统一展示，使用户之间能更好的交流互动。

文章列表界面展示文章标题、发布者、文章概要和文章总数信息，并采用分页处理方式以便用户分页浏览。用户可点击感兴趣的文章标题进入文章详情页，也可点击发布者链接，查看相应发布者的文章列表。分享展示界面设计如图 1-7 所示。

图 1-7 分享展示界面设计

证书图文展示界面展示用户发布的证书图文信息。以证书缩略图的形式整齐排列，供用户选择查看。用户通过点击感兴趣的缩略图，可以在弹窗中查看该用户发布的证书图文详情（证书图片、考证心得信息）。证书图文展示界面设计如图 1-8 所示。

图 1-8 证书图文展示界面设计

试题下载界面展示用户上传的试题列表，用户可通过点击列表左侧的试题名称查看试题内容，也可通过点击列表右侧的下载图标来下载试题至本地。试题下载界面设计如图 1-9 所示。

图 1-9 试题下载界面设计

1.2.4 个人空间管理界面

个人空间管理界面分为文章管理界面、图片管理界面、个人信息管理界面。在文章管理界面中，用户可以发布、修改、删除、查看文章；在图片管理界面中，用户可以发布、删除、查看图文信息；在个人信息管理界面中，用户可以对个人信息进行修改以及上传头像。个人空间界面设计如图 1-10 所示。

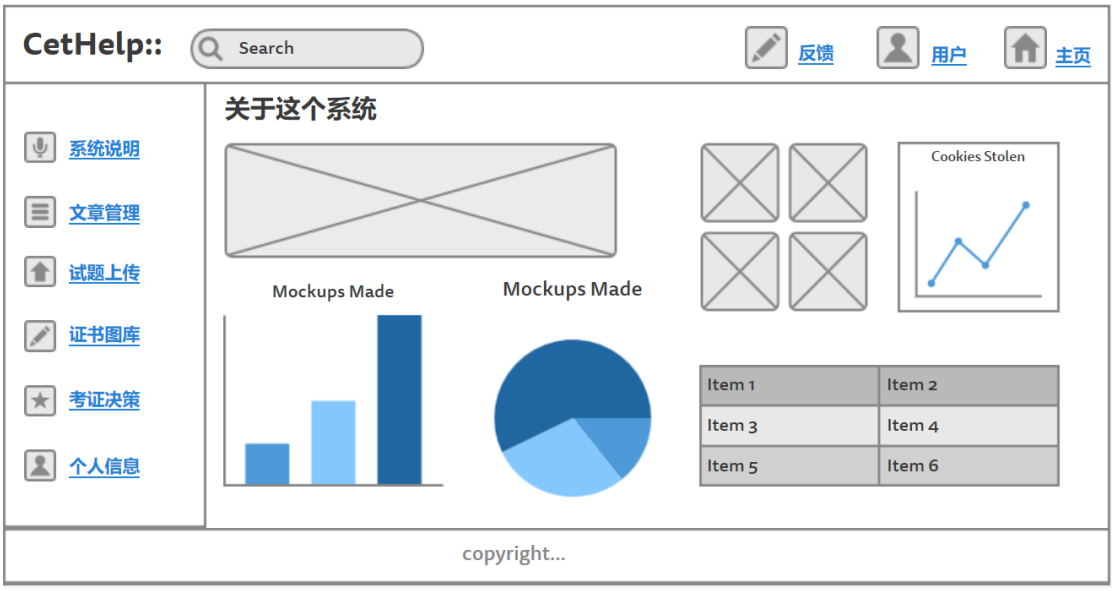


图 1-10 个人空间界面设计

文章管理界面提供用户栏目管理、文章发布以及文章列表功能。栏目管理提供栏目的增加、修改、删除功能；文章发布为用户编写考证经验分享提供便捷的平台，引入深受大学生欢迎的 Markdown 编辑器，界面简洁美观，功能丰富；文章列表提供文章信息的展示、修改、删除功能。以栏目管理界面为例，文章管理界面设计如图 1-11 所示。

图 1-11 文章管理界面设计

图片管理界面提供增加、展示、删除图文信息的功能。在个人空间界面中对应“证书图库”选项。其中，点击“添加图片”显示具体表单填写页，填写考证心得及上传证书图片后即可提交，添加成功后将在按钮下方的列表中同步更新。图片管理界面设计如图 1-12 所示。

图 1-12 图片管理界面设计

个人信息管理界面提供修改个人信息和图片上传功能。两个功能在页面中左右分布，点击“编辑个人信息”按钮即可修改个人信息；点击“上传头像”按钮即可上传头像。个人信息管理界面设计如图 1-13 所示。

图 1-13 个人信息管理界面设计

1.2.5 决策查询界面

决策查询界面分为 VS 决策查询界面和个性决策查询界面。VS 决策查询界面中，用户通过输入两个证书名称或关键词即可查询证书参数。本组成员选定了六个证书参数以辅助用户参考，分别是：证书名称、考试难度、证书性质、考证条件、考证时间、考证费用。VS 决策查询界面设计如图 1-14 所示。

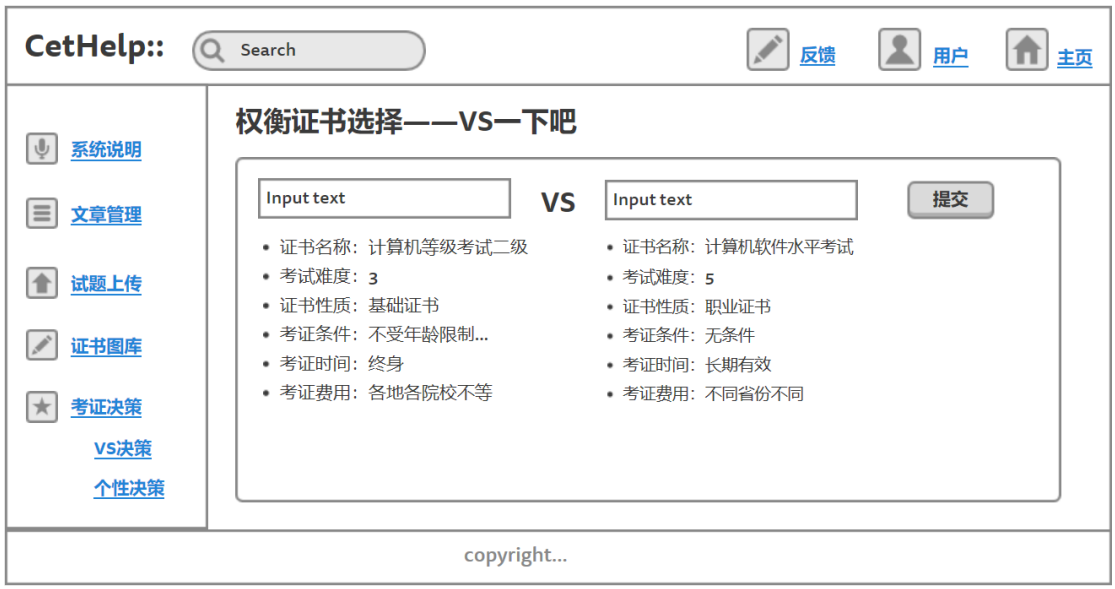


图 1-14 VS 决策查询界面设计

个性决策查询界面中，主体部分以问卷形式呈现。用户填写证书名称、证书类型和相应问题（问题分为三类：基本问题、学习能力问题、考证经验问题），点击“提交”按钮后，系统以弹窗形式反馈用户问卷评分和相应建议。个性决策查询界面设计如图 1-15 所示。

图 1-15 个性决策查询界面设计

1.2.6 管理员界面

管理员界面分为管理员列表界面和管理员管理界面。管理员列表界面中展示管理员可以查看的所有模块，管理员可以通过单击模块名称查看相应列表中用户的信息记录，并且可以通过点击“添加”、“修改”等按钮来改变模块内容。管理员列表界面设计如图 1-16 所示。

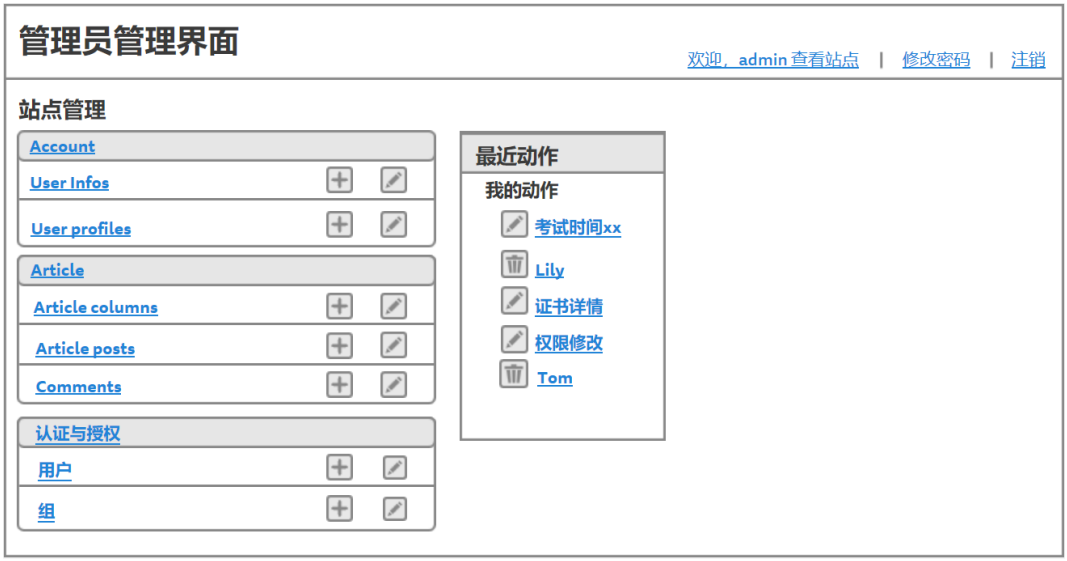


图 1-16 管理员列表界面设计

管理员管理界面显示作者发布的详细信息，方便管理员对用户发布的信息内容进行审核，管理员可通过点击相应图标实现不同功能。管理员管理界面设计如图 1-17 所示。

图 1-17 管理员管理界面设计

1.3 系统功能模块设计

本系统采用 HIPO 图进行系统功能模块设计，HIPO 图由层次化模块结构图和 IPO 图构成，前者描述了整个系统的设计结构以及各类模块之间的关系，后者描述了某个特定模块内部的处理过程和输入/输出关系[12]。

1.3.1 层次化模块结构图

根据本系统的特征，确定本系统的层次化模块结构为事务型（同时有多个事务需要处理）。输入部分为用户输入的相关信息，经审核后将合格的相关信息回送至系统，系统得到合格的相关信息后，调度中心（即处理部分）变换模块，将各类信息转换为结果信息，最后调用传出模块输出结果信息。

本系统的层次化模块结构图如图 1-18 所示，其中：a 代表相关信息，b 代表合格标志，c 代表合格信息，c1、c2、c3、c4 分别代表用户信息、发布信息、证书信息和问卷信息，对应的，d1、d2、d3、d4 分别代表各类信息的处理结果。d 代表处理结果。

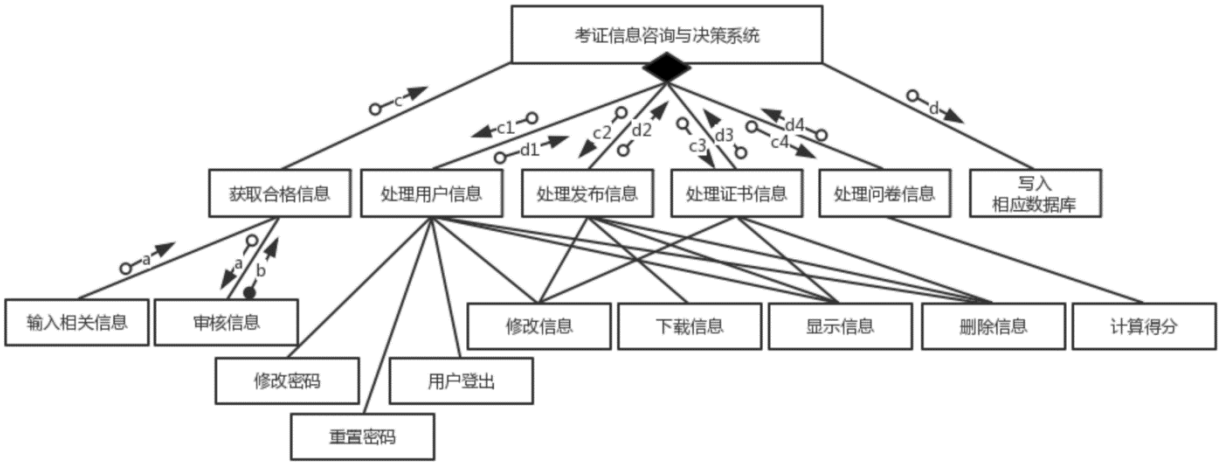


图 1-18 系统层次化模块结构图

1.3.2 IPO 图

IPO 图主要是配合层次化模块结构图详细说明每个模块的内部功能。由于 IPO 图输入、输出的设计和处理都很容易，唯独处理过程（P）描述较为困难，用自然语言很难描述清楚。因此本组采用决策树和伪代码的方式对 P 进行描述，下面就每个功能模块进行分析。

(1) 审核信息

审核信息是对系统使用者输入的信息进行审核，即检查信息是否合格。输入的信息可为用户登录信息、注册信息以及需要提交至后台的信息。检查内容为输入的信息是否存在为空、漏写、填写错误、格式错误的情况。若信息合格则将合格标志送回上一级调用模块，将检查的记录记入用户记录文件。若信息不合格则记录出错信息。审核信息 IPO 图如图 1-19 所示。

系统名称：大学生考证信息咨询与决策系统	
设计人：蒋昊瑾	设计日期：2018 年 3 月 10 日
模块名称：审核信息 上层调用模块：获取合格信息 下层被调用模块：无 模块描述：审核输入的相关信息，若信息审核合格则向上一级调用模块发送合格标志，若信息审核不合格则记录出错信息	
输入数据：相关信息	

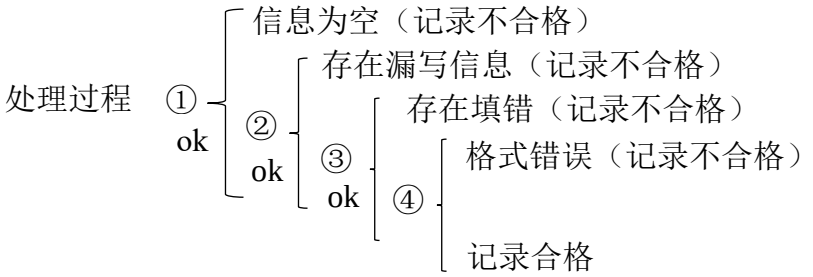
数据处理：

① 检验信息是否为空

② 检验信息是否漏写

③ 检验信息是否填错

④ 检验信息是否格式错误



输出数据：合格标志

备注：相关信息包括注册信息、登录信息、发布信息、问卷信息等

图 1-19 审核信息 IPO 图

(2) 修改密码

修改密码需要用户输入旧密码和想要更换的新密码，系统验证旧密码正确后方可更换密码。修改密码模块接收上级模块的信息参数，查询数据库，比较密码是否正确，并检验输入的新密码格式是否正确，若正确则修改数据库中的密码信息并进行页面跳转；错误则显示错误提示信息。修改密码 IPO 图如图 1-20 所示。

系统名称：大学生考证信息咨询与决策系统	
设计人：蒋昊瑾	设计日期：2018 年 3 月 10 日
模块名称：修改密码 上层调用模块：处理用户信息 下层被调用模块：无 模块描述：从上级模块接收密码信息，在数据库中比对旧密码与用户密码是否一致，若一致则检查新密码是否格式正确，若格式正确则修改数据库内容并跳转页面，否则记录错误信息	
输入数据： ① 旧密码（o_password） ② 新密码（n_password） 数据处理： ① 检验输入的旧密码是否为用户密码 ② 检验输入的新密码是否合理	

③ 修改密码 **IF** o_password equal to user's password **THEN IF** n_password is valid **THEN** user's password is n_password and do transition **ELSE** show the error **ELSE** show the error 输出数据：新密码、跳转事件
备注：普通用户和管理员修改密码模块相同

图 1-20 修改密码 IPO 图

(3) 重置密码

重置密码是用户忘记密码时可以输入邮箱验证完成密码重置。重置密码模块接收上级模块的邮箱信息，检验邮箱信息是否正确，若正确，则用户输入新密码即可重置密码，若邮箱验证失败或输入密码格式不正确则记录错误信息。重置密码 IPO 图如图 1-21 所示。

系统名称：大学生考证信息咨询与决策系统	
设计人：蒋昊瑾	设计日期：2018 年 3 月 10 日
模块名称：重置密码 上层调用模块：处理用户信息 下层被调用模块：无 模块描述：从上级模块接收邮箱信息，检验邮箱信息是否正确。若正确，则用户输入新密码即可重置密码，若邮箱验证失败或输入密码格式不正确则记录错误信息	
输入数据： ① 邮箱信息（email） ② 新密码（n_password） 数据处理： ① 检验输入的邮箱是否正确 ② 检验输入的新密码是否合理 ③ 重置密码 **IF** email is valid **THEN IF** n_password is valid **THEN** user's password is n_password and do transition **ELSE** record the error	

ELSE show the error 输出数据：新密码、跳转事件
备注：普通用户有重置密码功能

图 1-21 重置密码 IPO 图

(4) 用户登出

用户登出是用户退出系统的操作。用户登出模块接收上级模块的用户 session 信息，若注销 session 成功，则跳转至登录界面，若失败则记录失败信息。用户登出 IPO 图如图 1-22 所示。

系统名称：大学生考证信息咨询与决策系统	
设计人：蒋昊瑾	设计日期：2018 年 3 月 10 日
模块名称：用户登出 上层调用模块：处理用户信息 下层被调用模块：无 模块描述：用户登出模块接收上级模块的用户 session 信息，若注销 session 成功，则跳转至登录界面，若失败则记录失败信息	
输入数据： 用户 session 信息 数据处理： ① 是否注销 session 成功 ② 用户登出 **IF** session is invalidated **THEN** do transition **ELSE** show the error 输出数据：跳转事件	
备注：	

图 1-22 用户登出 IPO 图

(5) 修改信息

修改信息是指用户对个人信息、发布信息、证书信息、问卷信息进行修改。修改信息模块接收上级的修改信息，检验修改信息是否合理（信息是否格式正确，是否必填项都已填写，是否以空表单提交）。若合理，则将修改信息写入数据库并保存，执行跳转提示。若不合理则显示不合理提示。修改信息 IPO 图如图 1-23 所示。

系统名称：大学生考证信息咨询与决策系统	
设计人：蒋昊瑾	设计日期：2018 年 3 月 10 日
模块名称：修改信息 上层调用模块：处理用户信息、处理发布信息、处理证书信息 下层被调用模块：无 模块描述：修改信息模块接收上级的修改信息，检验修改信息是否合理，若信息合理，则将修改信息写入数据库，执行后跳转提示，否则显示不合理提示	
输入数据： 用户修改信息（m_info） 数据处理： ① 检查输入的修改信息是否合理 ② 修改该信息 **IF** m_info is valid **THEN** related information is m_info and return 1 **ELSE** return 0 输出数据：修改后的信息、跳转事件	
备注：需要修改信息模块处理的信息有用户个人信息、发布的图文信息、证书信息（只由管理员处理）	

图 1-23 修改信息 IPO 图

(6) 下载信息

下载信息是针对发布（上传）的试题信息进行下载操作。下载信息模块接收上级的下载信息，若下载成功则执行下载提示并将服务器上的文件写入硬盘，若失败则记录失败信息。下载信息 IPO 图如图 1-24 所示。

系统名称：大学生考证信息咨询与决策系统	
设计人：蒋昊瑾	设计日期：2018 年 3 月 10 日
模块名称：下载信息 上层调用模块：处理发布信息 下层被调用模块：无 模块描述：接收上级模块的下载信息，若下载成功则执行下载提示，若失败则记录失败信息	
输入数据： 下载信息（d_info） 数据处理：	

① 检查是否正常下载 ② 下载该信息对应的文件 **IF** d_info is valid **THEN** download the file from the server to the local and do transition **ELSE** show the error 输出数据：修改后的信息、跳转事件
备注：

图 1-24 下载信息 IPO 图

(7) 显示信息

显示信息是用户通过输入信息 ID 或信息名称来显示信息详情。显示信息模块接收用户输入的信息 ID 和信息名称，根据输入内容查询数据库中信息，返回查询结果。显示信息主要包括：显示证书详情信息、图文信息、评论及点赞信息、上传信息、反馈信息、个人信息。显示信息 IPO 图如图 1-25 所示。

系统名称：大学生考证信息咨询与决策系统	
设计人：蒋昊瑾	设计日期：2018 年 3 月 10 日
模块名称：显示信息 上层调用模块：处理用户信息、处理发布信息、处理证书信息 下层被调用模块：无 模块描述：接收上级模块的信息 ID 和信息名称，根据输入内容查询数据库中信息，若存在该信息则执行跳转显示，若不存在则提示错误	
输入数据： ① 信息 ID（i_id） ② 信息名称（i_name） 数据处理： ① 检查在数据库中是否存在该信息 ② 显示该信息 ID 或信息名称的数据 **IF** i_id is valid or i_name is valid **THEN** get specific information and do transition **ELSE** show the error 输出数据：具体信息、跳转事件	
备注：	

图 1-25 显示信息 IPO 图

(8) 删除信息

删除信息是通过信息 ID 和用户 session 将信息删除的操作。删除信息模块从上级模块接收信息 ID 和用户 session，验证用户是否为信息所有者，验证成功则删除该信息，失败则返回失败信息。删除信息 IPO 图如图 1-26 所示。

系统名称：大学生考证信息咨询与决策系统	
设计人：蒋昊瑾	设计日期：2018 年 3 月 10 日
模块名称：删除信息 上层调用模块：处理用户信息、处理发布信息、处理证书信息 下层被调用模块：无 模块描述：删除信息模块从上级模块接收信息 ID 和用户 session，验证用户是否为信息所有者，验证成功则删除该信息，失败则返回失败信息	
输入数据： ① 信息 ID（i_id） 数据处理： ① 验证用户是否为信息所有者 ② 删除该信息 ID 对应的数据 **IF** i_id is valid and i_id belongs to the user **THEN** delete the information and do transition **ELSE** record the error 输出数据：跳转事件	
备注：	

图 1-26 删除信息 IPO 图

(9) 计算得分

计算得分是用户提交问卷后系统根据用户所选选项来计算问卷得分。计算得分模块接收上级模块的提交信息，对应评分表中每个问题的分值来计算问卷得分。并将得分返回给用户，若用户的提交信息有误，则返回错误信息。计算得分 IPO 图如图 1-27 所示。

系统名称：大学生考证信息咨询与决策系统	
设计人：蒋昊瑾	设计日期：2018 年 3 月 10 日
模块名称：计算得分 上层调用模块：处理问卷信息 下层被调用模块：无	

模块描述：计算得分模块接收上级模块的提交信息，对应评分表中每个问题的分值来计算问卷得分。并将得分返回给用户，若用户的提交信息有误，则返回错误信息
输入数据： ① 提交信息（s_info） 数据处理： ① 验证提交信息 ② 对提交信息进行评分 **IF** s_info is valid **FOR** s in range s_info **THEN** get the score for each s and accumulate the score **THEN** return score **ELSE** show error 输出数据：分数信息
备注：

图 1-27 计算得分 IPO 图

2 PC 端系统代码实现

2.1 开发方法

大学生考证信息咨询与决策系统的需求较为稳定，功能模块数量不多，项目小组讨论后决定采用结构化系统开发方法。结构化系统开发方法最为成熟，开发工作的阶段性评估可以减少开发工作重复性和提高开发的成功率，开发工作的顺序性、阶段性降低了对开发人员的要求，适合项目小组成员参与系统开发。

2.2 开发策略

2.2.1 B/S 结构

因为考证信息咨询和决策的需求频率有限，如果开发一个独立的客户端程序，程序使用频率不会很高，而且用户成本过高、开发流程复杂且开发成本高，所以项目小组经过讨论后决定采用 B/S 结构进行系统开发。最大程度的降低系统对用户的要求，并且减少了系统在开发、维护和升级方面的支出成本以及用户的总体成本。

2.2.2 MVC 架构

在系统采用 B/S 结构进行开发的基础上，MVC 架构前后端分离的架构模式能提高开发效率，项目小组成员分工后可以同时进行开发，前端后端可以专注于

各自开发任务，最大程度的降低成员之间的耦合。除此之外，MVC 架构在开发后期也能降低系统修改完善成本，提高系统可维护性、稳定性。

2.2.3 开发工具

本组经讨论后决定利用 Python+Django+MySQL+bootstrap 组合进行编程。Python+Django 的开发组合是当今主流的 web 开发方法之一，灵活的 MVC 架构，Django 自带的丰富插件能够提高系统开发效率，降低开发成本。MySQL 作为最常见的关系型数据库管理软件，能够满足系统开发要求，且项目小组都有 MySQL 开发经验。bootstrap 能够规范系统前端页面样式，提高前端开发的效率。

2.2.4 Nginx 服务器

Nginx 是一款轻量级的 Web/反向代理服务器及电子邮件代理服务器，具有占有内存少，并发能力强，运行稳定的特点。它既可作为 http 服务器，又可作为虚拟主机。本组可以采用 Nginx+flup 或 Nginx+uwsgi 的方式部署项目，以 Nginx 作为 Web 服务器，接收 Web 的所有请求并对请求进行统一管理。

2.2.5 PyCharm 开发平台

PyCharm 是一款广受好评的 Python 开发平台，具有完善的调试、Project 管理、智能提示、单元测试等功能以便高效帮助用户完成开发工作。并且，此平台可用于支持 Django 框架下的专业 Web 开发。本系统使用的开发组合为 Python+Django+MySQL，在 PyCharm 平台中可方便的下载、管理 Python 模型库及项目文件并可便捷的连接后台数据库，为开发工作提供便利。

2.3 系统开发基础配置

2.3.1 Django 配置

① 系统正式开发之前，需要为系统创建一个 project(mysite)，project 的功能类似于一个基础的开发平台，包含数据库配置、安装 App、日志配置、中间件以及一些基本的目录配置。

```
django-admin startproject mysite
```

② project 创建完毕后，还需要创建特定 App 以实现特定的功能，以 account App 为例：

```
python manage.py startapp account
```

③ 在 project 目录 settings.py 文件的 Installed_apps 部分加入相关的 App 配置信息，以确保基础的开发环境 project(mysite)都能正常读取各个 App 的内容，其具体内容如下：

```
INSTALLED_APPS = (
    'django.contrib.admin',
```

```
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'account',  # 将 account app 加入配置项
    )
```

2.3.2 数据库连接配置

修改默认的数据库引擎为 MySQL，同时在 MySQL 中创建数据库（create database certificate），确定数据库用户名、密码以及 MySQL 的端口号，测试连接是否正常，其具体配置如下所示：

```
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.mysql',
            'NAME': 'certificate',
            'USER': 'root',
            'PASSWORD': 'root',
            'HOST': 'localhost',
            'PORT': '3306',
        }
    }
```

2.3.3 语言设置

```
    LANGUAGE_CODE= 'zh-hans'
```

一般情况下可以不用修改，但是本系统的输入信息大多为汉语输入，因此需要设置 LANGUAGE_CODE 为中文。

2.3.4 时区设置

```
    TIME_ZONE = 'Asia/Shanghai'
```

开发过程中信息发布时间、更新时间等重要时间节点都是通过 datetime.now()函数设置，因此需要设置正确的时区保证 datetime.now()函数返回正常值。

2.3.5 系统静态文件目录配置

静态文件是指在系统开发运行过程中不会出现较大改变的系统文件，包括 css 样式表、javascript 文件、图片等。将系统静态文件放入一个固定目录，用户访问系统时，服务器会直接返回静态文件，由于不需要经过服务器的处理，不会调用数据库，也不会调用脚本，静态文件能加快服务器处理速度，即使在系统访

问量很大的情况下，静态文件也能很快的加载。

① 在 project(mysite)目录的 settings.py 文件中添加以下设置，确保 project 能够正常读取静态文件，如下所示：

```
STATIC_URL = '/static/'
```

③ 继续编辑 mysite 目录下的 settings.py 文件，在上述代码之后增加如下代码：

```
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, "static"),
)
```

③ 创建 static、templates 两个目录，分别用来保存系统静态文件和系统 html 模板文件，为了避免不同 App 之间产生重名冲突，在 static 目录下分别创建 css、fonts、images、js 文件；在 templates 目录下创建与 app 名称相同的目录，对应的文件将会保存在这两个文件内,如图 2-1 所示。

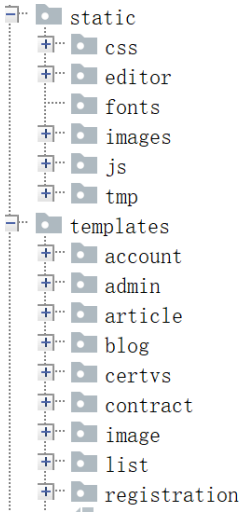


图 2-1 static、templates 目录文件

2.3.6 url.py 设置

同一个 project 下面可能会有很多不同的 App，为了保证每个 App 都能正常访问，在 project 的 url.py 文件中需要设置一个固定的路由，将系统开发工作固定到 mysite App 内部，系统开发修改将不会对整个 project 产生影响，确保整个 project 的正常运行。如,在 mysite 目录的 urls.py 中为 account App 进行路由设置：

```
urlpatterns = [
    url(r'^account/', include('account.urls', namespace='account',
app_name='account')),
]
```

接下来就需要配置./account/urls.py 中的 URL，App 文件夹中没有 urls.py 这个文件，需要自己创建。如，在 account App 中编写登录与登出功能，其页面设

置如下：

```
urlpatterns = [
    url(r'^login/$', views.user_login, name='user_login'),
    url(r'^logout/$', auth_views.logout, {"template_name":
"account/logout.html"}, name='user_logout'),
]
```

2.4 系统基础功能实现

2.4.1 数据库实现

根据数据库设计的结果修改各 App 目录内 models.py 文件，以证书库为例，证书库是 MySQL 中 certificate 数据库中已有的数据表，可利用 Django 的 inspectdb 工具自省配置文件指向的数据库，针对数据表生成 Django 模型：

```
python manage.py inspectdb
python manage.py inspectdb > models.py
```

新建 list App，其功能是展示证书信息列表和证书详情。把上述生成的 Django 模型导入 list App 中，这样便能够在 list App 的 models.py 文件中查看 class 信息。其具体操作如下所示：

```
django-admin.py startapp list
python manage.py inspectdb > list/models.py
```

对于直接在 Django 中编写数据模型的情况，只需在各 App 下的 models.py 文件中编写相应的 class 即可，这个类与数据库中的数据表具有对应关系。每一张数据表对应一个 class，每一个数据库字段对应 class 类中的一个属性值。以 account App 的 UserProfile 类为例，其关键代码如下：

```
class UserProfile(models.Model):
    user = models.OneToOneField(User, unique=True)
    stunamber = models.CharField(max_length=20, null=True)
    major = models.CharField(max_length=20, null=True)

    def __str__(self):
        return 'user {}'.format(self.user.username)
```

其中，class Userprofile(models.Model)所定义的数据模型类对应着要在数据库中建立名为 account_userprofile 的数据表；user 字段中 OneToOneField()的含义是通过 user 这个字段声明 UserProfile 类与 User 类之间的关系是“一对一”的。

2.4.2 后台数据绑定

管理员需要在系统后台对用户、物品等进行管理，为了实现这一需求，系统需要将数据模型绑定到管理员模型，从而管理员设置可以直接读取数据模型中数据。创建管理员须输入以下代码：

```
python manage.py createsuperuser
```

之后按要求输入用户名和密码即可，该用户名和密码用于登录管理员界面。一般情况下，普通用户无法登录管理员界面，普通用户得到管理员权限后方可登录。

管理员界面的样式大多情况下是 Django 的默认样式，只需通过代码增加页面功能和调整页面布局。以./account/admin.py 为例，将 UserProfile 所表示的表的内容在后台可视化，其关键代码如下：

```
from django.contrib import admin
from .models import UserProfile

class UserProfileAdmin(admin.ModelAdmin):
    list_display = ('user', 'stunumber', 'major')
    list_filter = ('major',)

admin.site.register(UserProfile, UserProfileAdmin)
```

其中，在 UserProfile 管理区域中展示 user（用户名）、stunumber（学号）、major（专业）三个属性，并可按照 major 属性对各字段进行排序；在 UserInfo 管理区域中展示上述七个属性，并通过 school、major、getcert（所获证书）属性对各字段进行排序。以 UserProfile 的管理页面为例，其效果如图 2-2 所示。

图 2-2 UserProfile 管理页面

2.4.3 服务器搭建及项目部署

系统功能基本实现后，需要将项目放入服务器以便外网访问。首先，利用Nginx 服务器在本地进行搭建，需要两个组件：Nignix 服务器和 flup（Python 的FastCGI 组件），修改 Ngnix 文件夹中的 nginx.conf 配置文件，在 server 中加入项目路径并设置静态资源等。在项目文件夹下运行 python manage.py runfcgi method=threaded host=127.0.0.1 port=8051，打开浏览器，在网址栏中输入 http://localhost 即可访问系统。之后，需要外网能够访问到本系统。本组使用腾讯云来提供云主机，在腾讯云的控制台重装系统并将电脑远程连接到服务器，在控制台下导入项目文件并利用上述方法搭建 Ngnix 服务器，然后启动 Ngnix 并利用FastCGI 运行项目。之后即可通过腾讯云外网地址访问该系统。

2.5 系统功能模块实现

2.5.1 获取合格信息

获取合格信息包括提交登录信息、提交注册信息、发布文章信息、发布图片信息、发布证书信息、提交问卷信息、提交上传信息、提交反馈信息。

(1) 提交登录信息

用户登录功能的实现使用了 Django 的内置方法，其具体代码在 Django 目录的./contrib/auth/views.py 文件中。因此，只需在 account App 中加入其路由描述便可实现登录功能。本小组没有使用默认登录模板，而是给函数 template_name 传值，传递一个新的模板文件：

```
url(r'^login/$', views.user_login, name='user_login')
```

之后，在 account App 的 views.py 文件中对用户提交的登录信息进行判断，并显示相应的结果。首先利用 request.method == "POST" (或"GET") 判断提交方式，如果是 POST 提交则验证登录信息，如果是 GET 提交则显示空表单并刷新页面。

当利用 POST 提交方式提交后，首先判断提交是否合理（是否存在漏填现象），若合理则进一步判断其用户名和密码是否填写正确，若正确则获取证书列表信息并显示系统首页，若不正确则在登录界面显示相应错误信息（若填写不合理，则提示 Invalid login，若用户名或密码填写错误，则提示 Username or Password Wrong）。其关键代码如下所示：

```
if login_form.is_valid():
    cd = login_form.cleaned_data
    user = authenticate(username=cd['username'], password=cd['password'])
        if user:
```

```
                login(request, user)
                lists = CertificateTbl.objects.all()
                return render(request, "list/list.html", {"lists":lists})
                else:
                    error = "Username or Password Wrong"
                    return render(request, "account/login.html", {"form": form,
                "error":error}
    else:
        error = "Invalid login"
        return render(request, "account/login.html", {"form": form, "error": error})
```

若用户登录信息正确，则进入证书查询列表页。本组根据用户是否登录进行不同的显示。即用户已经登录时，导航栏显示系统所有功能；用户尚未登录时，导航栏显示 LOGIN 字样，提示用户登录。其关键前端代码如下：

```
<ul class="res">
{% if user.is_authenticated %}
    <li><a href="{% url 'account:user_logout' %}">
            LOGOUT({{ user.username }})</a></li>
{% else %}
    <li style="float:right"><a href="{% url 'account:user_login' %}">
            LOGIN</a></li>
{% endif %}
</ul>
```

(2) 提交注册信息

提交注册信息和提交登录信息在功能上相似。在设置好登录的 URL 信息后须在 account App 的 forms.py 文件中进行表单设置，表单内容须根据注册表中的内容进行设置，其中最重要的是需要判断“密码”项和“确认密码”项是否输入一致。若两次输入不一致，则报错，否则返回确认密码数据。判断方法如下所示：

```
def clean_password2(self):
    cd = self.cleaned_data
    if cd['password'] != cd['password2']:
        raise forms.ValidationError("passwords do not match.")
        return cd['password2']
```

之后，在 account App 的 views.py 文件中对用户注册信息进行判断。与登录提交判断类似，首先判断提交方式为 POST 还是 GET，若为 POST 则进入具体判

断，进一步判断所提交的表单内容是否合理，若表单内容合理则创建该注册用户对象 UserInfo.objects.create(user=new_user)并返回注册成功提示，若不合理则显示核对表单内容提示。

(3) 发布文章信息

在发布文章之前，需要用户新增文章栏目信息。这里使用 layer.js 插件来实现弹窗效果，实现弹窗需引入必要的 jQuery 和 layer。使用一个按钮来触发 add_column()函数。利用 ajax 提交栏目信息，若成功则显示弹窗内容为“good”的提示并增加此栏目，若失败则提示更换栏目信息。其关键代码如下所示：

```
$.ajax({
    url:'{% url "article:article_column" %}',
    type:'POST',
    data:{"column":column_name},
    success:function(e){
        if(e=="1"){
            parent.location.reload();
            layer.msg("good");
        }else{
            layer.msg("此栏目已有，请更换名称")
        }
        },
})
```

对于发布文章功能，后台设置与添加栏目类似。在前台页面中，考虑使用深受大学生欢迎的 Markdown 编辑器。Markdown 是一种便捷的标记语言，可以完全可视化的操作，本组使用 editor.md 插件来实现编辑器功能。

在编写前台文章发布表单时需要注意，在<form>的开始标签后要添加{% csrf_token %}标签，这是一种 CSRF（跨站点伪造请求）防护机制。即 Django 第一次响应客户端发来的请求时，则会在服务器端随机生成一个 token，这个 token 会被放入 cookie 中，之后的每次 POST 请求都会带上这个 token。之后，在模板文件的编辑文章位置加入 Markdown 编辑器的内容，其关键代码如下所示：

```
<div id="editormd" class="col-md-12 text-left" style="margin-top:30px">
    <textarea style="display:none;" id="id_body"></textarea>
</div>
```

部署好 editor.md 的相关文件后，在模板文件尾部加入 JavaScript 文件和相应的脚本代码如下：

```
$(function() {
    var editor = editormd("editormd", {
        width: "70%",
        height: 640,
        syncScrolling: "single",
        path: "{% static 'editor/lib/' %}",
        taskList: true,
})
```

(4) 发布图片信息

图片信息发布是指上传证书图片以及发布相关文字信息（标题、简述），因此在编写 image App 的 views.py 文件时需要注意，当提交方式为 POST 时，接收上传的图片文件内容 FILES 以及以 POST 方式进行提交的文字内容，即 form = ImageForm(request.POST,request.FILES)。

检验方式与上文中的检验方式类似。值得注意的是，由于在 image 表中 user 属性作为外键存在，即 user = models.ForeignKey(User, related_name="images")，因此在表单中用户名项可省略。当表单以 POST 方式提交后需要先预存（这时并没有保存到后台数据库），即 new = form.save(commit=False)，然后获取用户名信息 new.user = request.user，之后即可通过 save()方法保存信息至后台。

(5) 发布证书信息

只有管理员才可发布证书信息。管理员相关操作须编写 admin.py 文件。首先编写大致的证书列表界面和相关功能，其关键代码如下所示：

```
class CertificateTblAdmin(admin.ModelAdmin):
    list_display = ('cert_id', 'cert_class', 'cert_name', 'cert_time', 'cert_link')
    list_filter = ('cert_class', 'cert_name')
    search_fields = ('cert_name', 'cert_ename')
    raw_id_field = ('cert_name')
    ordering = ['cert_id','cert_name']
admin.site.register(CertificateTbl, CertificateTblAdmin)
```

其中，list_display 定义了展示内容，list_filter 定义了筛选项，search_fields 定义了查询项，raw_id_field 定义了外键详细信息，ordering 定义了排序依据。

(6) 提交问卷信息

问卷相关数据表为 Question 表和 Choice 表，question 属性在 Choice 表中作为外键存在（question = models.ForeignKey(Question)）由于问题、选项、选项分值等信息均由管理员输入，因此只需将问卷相关表的内容显示至前台，并通过提

交按钮显示不同结果即可。

首先，获取证书的全部列表信息，即 latest_question_list = Question.objects.all()。之后在前台页面中，将选项分值(choice.value)信息作为每一个选项中 value 的值，将选项编码(choice.class)作为每一个选项中 id 的值（这样可直接利用 javascript 计算分值）。其关键前端代码如下所示：

```
{% if latest_question_list %}
<form action="." method="post">{% csrf_token %}
    {% for question in latest_question_list %}
        <h1>{{ question.question_text }}</h1>
        {% for choice in question.choice_set.all %}
        <input type="radio" name="radio{{question.id}}"
        id="{{choice.choice_class}}" value="{{ choice.value}}" />
        <label for="choice{{ forloop.counter }}">
        {{ choice.choice_text }}</label><br />
        {% endfor %}
    {% endfor %}
    <input type="submit" value="Submit" />
</form>
{% endif %}
```

(7) 提交上传信息

提交上传的文件信息，若为 POST 提交方式，则获取文件对象以及文件对象的名称。如果该文件存在，则保存对象并显示成功提示。若不存在则显示出错提示。其视图函数关键代码如下：

```
if request.method == 'POST':
    new_file = File(
        file=request.FILES.get('file'),
        name=request.FILES.get('file').name
    )
    if not new_file:
        return HttpResponse("sorry")
    else:
        new_file.save()
        return render(request, 'account/success.html')
```

(8) 发布反馈信息

与上文中的发布信息写法类似，即当提交方式为 POST 时，获取提交内容并检查发布内容是否合理，若合理则保存该内容并显示发布成功提示，若不合理则显示发布失败提示。其关键代码如下所示：

```
if request.method == "POST":
    form = ContractForm(request.POST)
        if form.is_valid():
            form.save()
            return render(request, "contract/ok.html")
        else:
            return HttpResponse("Sorry")
```

2.5.2 处理用户信息

处理用户信息包括修改密码、重置密码、用户登出、修改用户信息、显示用户信息、删除用户信息。

(1) 修改密码

用户登录后，可在个人空间中对密码进行修改。在 Django 中默认有修改密码的方法，其使用方法与前述的用户登录的内置方法类似。设定修改密码的 URL，其关键代码如下：

```
url(r'^password-change/$',auth_views.password_change,{"post_change_redirect":
"/account/password-change-done"}, name='password_change'),
url(r'^password-change-done/$',auth_views.password_change_done,
name='password_change_done'),
```

其中，"post_change_redirect": "/account/password-change-done"表示将冒号后的值传给冒号前的参数，这样即可用自定义的模板样式替换默认模板样式。

之后只需编写修改密码界面并给出此界面的入口即可。

(2) 重置密码

由于用户忘记密码后是无法找回的（系统对用户密码的加密方法不可逆），因此，重置密码功能对用户来说非常重要。Django 提供了重置密码的默认功能。首先，为与本系统风格保持一致，需重写密码重置界面。并将这个页面模板发送至用户的邮件内容。写好相应的重置密码界面后，无需再写视图函数，可直接进行 URL 配置，其具体形式可参考上文 URL 配置内容。

由于我们的意图是在一个服务器上向另一个服务器发送邮件。这里使用的默认发送邮箱是开发者的 QQ 邮箱，此邮箱需开启权限获取校验码后方可使用。输入每一种邮箱的特定端口号即可向其他邮箱发送信息。因此，要实现邮件发送功

能，还须在./mysite 下的 settings 文件中配置邮件服务器，其具体内容如下：

EMAIL_HOST = 'smtp.qq.com'

EMAIL_HOST_USER = '1316918731@qq.com'

EMAIL_HOST_PASSWORD = 'hjzeqlzmrardfhbc'

EMAIL_PORT = 587

EMAIL_USE_TLS = True

DEFAULT_FROM_EMAIL = '1316918731@qq.com'

其中，EMAIL_HOST 用来设置服务器，EMAIL_HOST_USER 用来设置用户名，EMAIL_HOST_PASSWORD 用来设置口令（这里须对发送方邮箱账号开启权限，并将权限验证码赋值过来），EMAIL_PORT 用来设置端口号，EMAIL_USE_TLS 表示安全连接。DEFAULT_FROM_EMAIL 为默认发送方邮箱。

用户忘记密码时，只需点击登录界面表单中的 Forget Password 选项即可进入忘记密码页面，用户提交自己的邮箱地址，系统将发送一封邮件至用户邮箱，收到邮件后，点开邮件中的链接，将会跳转至修改密码界面，之后的操作可参考上文密码修改一节。

(3) 用户登出

用户登出采用 Django 内置方法实现。与登录类似，首先配置 URL，即：

url(r'^logout/$', auth_views.logout, {"template_name": "account/logout.html"}, name='user_logout')

配置好后，在相应前端界面中加入登出界面的入口即可：

LOGOUT({{ user.username }})

其前端界面为用户登出提示信息，并给出登录连接，点击即可再次登录。

(4) 修改用户信息

用户可以查看和完善注册信息，用户编辑信息后点击提交按钮，将返回查看个人信息界面。本组将在视图函数中先把已有用户的信息读出来，然后判断用户的请求是 POST 还是 GET。如果是 GET，则显示表单，并且将用户已有信息页显示在其中；如果是 POST，则接收用户提交的表单信息，然后更新各个数据模型实例属性的值并保存。值得注意的是，用户信息与三个数据表相关（User 表、UserProfile 表，UserInfo 表），后两个表与 User 表的关系是一一对应关系（OneToOne）,因此通过用户名获取三个表中的信息的操作如下所示：

user = User.objects.get(username=request.user.username)

userprofile = UserProfile.objects.get(user=request.user)

userinfo = UserInfo.objects.get(user=request.user)

视图中还引入 HttpResponseRedirect 库，其作用是实现 URL 的转向，编辑时

的 url 为./edit-my-information/（这个路径在 urls.py 中配置），当用户提交了个人信息并被后端程序验证通过和保存后，就通过之前的引入项将页面转到./my-information/，用户能够看到自己的修改结果。

修改用户信息还包括上传头像功能。为方便起见，用户只允许上传一次头像（即不对用户上传的多个头像进行管理）。其上传后的判断与上文类似，获取 POST 信息，判断头像信息是否存在，若存在则提示头像已存在信息，若不存在则保存头像至后台并在前台显示头像。其关键部分代码如下所示：

```
if userinfo.img:
    error="你已经上传过头像，不可二次上传"
    return render(request, 'account/uploadimg.html',{'error': error})
else:
    userinfo.img = img
    userinfo.save()
    return render(request, 'account/myself.html',{'user': user, 'userinfo': userinfo, 'userprofile': userprofile})
```

(5) 显示用户信息

显示用户信息是将用户注册信息或用户修改后的用户信息显示出来。其具体操作与修改用户信息类似，只是 views.py 文件中的函数有所不同，展示用户信息只需根据用户名属性获取数据库中用户的详细信息并在前端展示即可:

```
def myself(request):
    user = User.objects.get(username=request.user.username)
    userprofile = UserProfile.objects.get(user=user)
    userinfo = UserInfo.objects.get(user=user)
    return render(request, 'account/myself.html', {'user': user, 'userinfo': userinfo, 'userprofile': userprofile})
```

(6) 删除用户信息

删除用户信息的操作者是管理员。由于后台界面的功能函数都是默认的，只需调用即可。通过编写 admin.py 文件设置用户信息展示页面的功能和布局展示后台界面。管理员只需点击想要删除的行，并在动作栏中选择删除项即可，值得注意的是，后台中，User 表、UserProfile 表、UserInfo 表的内容是分开管理的（不像前台页面中是显示在一起的），因此要删除某用户的信息需要在这三个表的管理界面中分别进行。

2.5.3 处理发布信息

处理发布信息包括修改发布信息、下载发布信息、显示发布信息、删除发布

信息。

(1) 修改发布信息

修改发布信息指的是修改栏目信息和修改文章信息。首先，修改栏目信息是点击图标后以弹框形式显示栏目，并在弹框中进行修改。点击铅笔图标，触发修改函数，其关键前端代码如下所示：

```
<a name="edit" href="javascript:"
onclick="edit_column(this, {{ column.id }})"><i class="lnr lnr-pencil"></i></a>
```

在修改发布信息的视图函数中，需要通过栏目 ID 获取具体对象，并将用户修改的栏目进行保存，成功则返回 1，若存在异常，则返回 0。其关键代码如下：

```
try:
    line = ArticleColumn.objects.get(id=column_id)
    line.column = column_name
    line.save()
    return HttpResponse("1")
except:
    return HttpResponse("0")
```

在 edit_column 函数中利用 ajax 提交弹框中用户输入的新栏目内容，并对视图函数返回的结果进行显示，其关键操作代码如下：

```
success: function(e){
    if(e=="1"){
        parent.location.reload();
        layer.msg("good");
        }else{
            layer.msg("新的名称没有保存,修改失败。")
        }
    }
```

修改文章信息与此类似，即单击修改文章的图标，将该文章的 ID 传给相应的视图函数（article = ArticlePost.objects.get(id=article_id)）。视图函数从数据库中读取该文章 ID 的相关内容。将文章相关内容呈现在界面上，并且处于编辑状态（<textarea>{{ article.body }}</textarea>），要求依然使用 Markdown 编辑器（在 javascript 代码中加入该插件的文件）。用户编辑之后，单击“提交”按钮，再次将页面中编辑的内容保存在上述 ID 所对应的数据库记录中。

(2) 下载发布信息

文件下载功能需要一个迭代器来处理文件，然后把这个迭代器作为参数传递

给 StreamingHttpResponse 对象，为避免文件流以乱码形式显示到浏览器的问题，我们采用将文件写入硬盘的方式，即给 StreamingHttpResponse 对象的 Content-Type 和 Content-Disposition 字段进行赋值，其关键代码如下：

```
try:
    def file_iterator(file, chunk_size=512):
        with open(file) as f:
            while True:
                c = f.read(chunk_size)
                if c:
                    yield c
                else:
                    break
        response = StreamingHttpResponse(file_iterator(file))
        response['Content-Type'] = 'application/octet-stream'
        response['Content-Disposition'] ='attachment;filename="{0}"'.format(file)
        return HttpResponse("1")
except:
        return HttpResponse("2")
```

(3) 显示发布信息

显示发布信息包括显示显示文章信息、显示图文信息、显示下载信息。其具体编写方式相似，这里以显示文章信息为例。

文章列表将所有用户发布的文字分享信息以列表形式展示出来，用户点击标题链接即可查看分享具体内容。其主要功能为切分功能、分页功能和点击进入文章详情。其前端页面如图 2-3 所示。

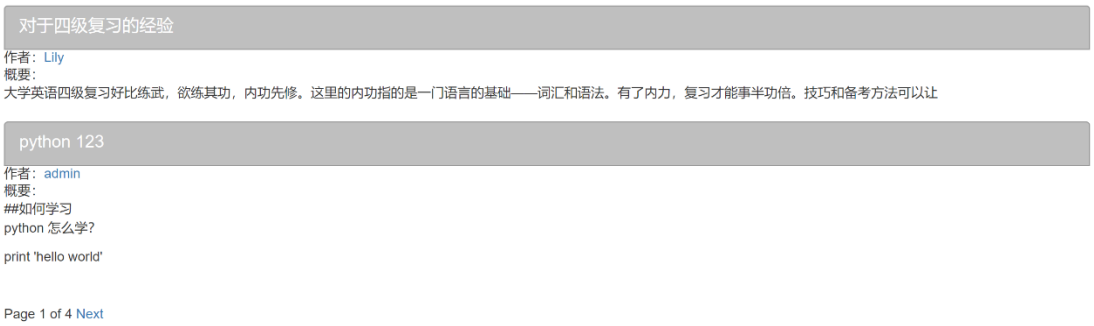


图 2-3 文章列表页面

① 系统为文章列表提供切分功能。以{{ article.body }}显示文章对象的所有文章内容，用管道符“|”对显示的内容做限制，“slice:’70’的含义是将前面的

变量内容“切下”前面 70 个字符。具体前端关键代码如下：

```
<p class="list-group-item-text">概要：{{article.body|slice:'70'| linebreaks}}</p>
```

② 系统为文章列表提供分页功能。分页是 Django 中的常见操作，因此，提供了内置的分页方法。这里对 article_list()视图函数进行重写，其关键代码如下：

```
paginator = Paginator(articles_list, 4)
    page = request.GET.get('page')
    try:
        current_page = paginator.page(page)
        articles = current_page.object_list
    except PageNotAnInteger:
        current_page = paginator.page(1)
        articles = current_page.object_list
    except EmptyPage:
        current_page = paginator.page(paginator.num_pages)
        articles = current_page.object_list
```

其中，第一句根据所查询到的文章对象 articles_list 创建分页的实例对象，并且规定每页最多 4 个。获得当前浏览器 GET 请求的参数 page 的值，也就是当前浏览器所请求的页码数值。代码中 page()是 Paginator 对象的一个方法，其作用在于得到指定页面内容，其参数必须是大于或等于 1 的整数。代码中 object_list 是 Page 对象的属性，能够得到该页面所有的对象列表。

③ 点击文章列表中的文章标题，可进入文章具体页。由于./article/models.py 文件中的 ArticlePost 类已经实现，因此只需更改文章列表中相应的超链接，即可进入文章具体页。文章具体页包括点赞、浏览次数、评论和侧栏列表（最受欢迎文章、最新文章、最多评论文章）。其前端页面如图 2-4 所示。

图 2-4 文章具体页

④ 点赞功能中，如果用户点击“like”/“unlike”图标，则首先获取此文章的 id 以及相应的 action(like 或 unlike)。如果用户点赞了该文章，则用户 like（数据表中字段）的计数加 1，并返回 1，若用户选择了不喜欢该文章，则用户 like 计数减 1，返回 2。其视图函数关键代码如下图所示：

```
try:
    article = ArticlePost.objects.get(id=article_id)
    if action=="like":
        article.users_like.add(request.user)
        return HttpResponse("1")
    else:
        article.users_like.remove(request.user)
        return HttpResponse("2")
```

在前端文件中编写 like_article 函数，其作用是将 id 和 action 传给视图函数，并对视图函数返回结果进行处理。其具体代码如下所示：

```
$.ajax({
    url: "{% url 'article:like_article' %}",
    type: "POST",
    data: {"id":id, "action":action},
    success: function(e){
        if(e=="1"){
            layer.msg("感谢点赞");
            window.location.reload();
        }else{
            layer.msg("我会继续努力");
            window.location.reload();
        }
    },
})
```

⑤ 浏览次数功能需要使用 Redis 数据库来实现。安装好 Redis 后（在 python 下也须安装 redis 库），保持 Redis 服务在启动状态下，方可实现记录浏览次数功能。

首先在./mysite/settings.py 文件中做好数据库的配置，其变量对应于 Redis 中的初步设置，即在文件中增加如下变量：

```
REDIS_HOST = 'localhost'
```

```
REDIS_PORT = 6379
REDIS_DB = 0
```

之后，编辑本文的 article_detail()视图函数，对访问文章的次数进行记录。这里用到 incr 函数，它的作用是能够递增当前键值，并且返回递增结果。其关键代码如下所示：

```
total_views = r.incr("article:{}:views".format(article.id))
```

并在前台代码中写入访问次数增加的代码，其具体内容如下所示：

```
<span style="margin-left: 15px">
<i class="lnr lnr-eye"></i>
{{ total_views }}view{{ total_views | pluralize }}</span>
```

⑥ 评论功能需要在 aricle_detail 视图函数中加入具体操作代码，即接收 POST 请求的数据，并判断数据内容是否合理，若合理则预存评论数据并将所评论文章赋值给该评论的文章属性。其关键代码如下所示：

```
 if request.method == "POST":
        comment_form = CommentForm(data=request.POST)
        if comment_form.is_valid():
            new_comment = comment_form.save(commit=False)
            new_comment.article = article
            new_comment.save()
 else:
        comment_form = CommentForm()
```

⑦ 最受欢迎文章是在文章访问统计的基础上建立的，需要编辑 article_detail()函数，相关代码如下所示：

```
r.zincrby('article_ranking', article.id, 1)
article_ranking = r.zrange('article_ranking', 0, -1, desc=True)[:10]
article_ranking_ids = [int(id) for id in article_ranking]
```

其中，第一个语句使用了 Redis 连接对象的 zincrby 方法，其作用是根据 amount 所设置的步长值增加有序集合（name）中的 value 的数值，即在 article_ranking 中的 article.id 以步长 1 自增，也就是文章被访问一次，article_ranking 就将该文章 id 的值增加 1。

其前端以列表形式展示，关键代码如下所示：

```
<ol>
{% for article_rank in most_viewed %}
    <li>
```

```
<a href="{{article_rank.get_url_path}}">{{ article_rank.title }}</a>
</li>
{% endfor %}
</ol>
```

⑧ 最新文章功能也可以通过自定义标签实现，可使用 inclusion_tag 实现。在./article/templatetags/article_tags.py 中增加自定义标签的函数，其代码如下：

```
def latest_articles(n=5):
    latest_articles = ArticlePost.objects.order_by("-created")[:n]
    return {"latest_articles": latest_articles}
```

其中，语句一定义了函数名称和参数，在具体使用此标签时会向这里传入参数。语句二中的 order_by()用于实现按照文章发布的时间倒序查询所有文章对象，并且根据参数(n)的值，获取排在前面的若干个。这里得到的值是一个以文章对象为元素组成的列表。

显示“最新文章”栏目，使用自定义标签 latest_articles，后面的数字是向这个自定义标签函数传入的参数值，即显示最新的 3 篇文章，其具体代码如下所示：

```
<p class="text-center"><h3>最新文章</h3></p>
<div>{% latest_articles 3 %}</div>
```

⑨ 最多评论文章使用 assignment_tag 自定义标签实现，其关键代码如下所示：

```
def most_commented_articles(n=3):
return ArticlePost.objects.annotate(total_comments=Count('comments'))
.order_by("-total_comments")[:n]
```

其中语句一中以参数的形式说明显示评论最多的文章数量，这里默认显示评论数量最多的前 3 篇。语句二利用 annotate()函数给 QuerySet 中的每个对象增加注释（用 Count('comments')进行标注，Count'comments'的作用是给每篇文章的评论计数）。“最多评论文章”栏目的显示如下所示：

```
<p class="text-center"><h3>最多评论文章</h3></p>
 {% most_commented_articles as most_comments %}
 <div><ul>
    {% for comment_article in most_comments %}
    <li>
    <a href="{{comment_article.get_url_path}}">{{ comment_article.title }}</a>
     </li>
```

```
        {% endfor %}
    </ul></div>
```

点击文章列表中的作者名称，可以进入作者文章列表页，其功能有列表展示、显示用户个人信息、分页、点击文章标题显示具体页。这四个功能在上文中都已写明具体实现过程，因此在此不进行重复说明。

(4) 删除发布信息

删除发布文章包括删除栏目、删除文章、删除图片。这三种具体功能的写法类似，这里以删除文章为例。

在前台点击删除图标，并在删除函数中通过 ajax 将需要删除的文章的 id 传递到相应视图函数，视图函数根据文章 id 找到具体文章信息并删除此文章。并将结果信息返回，删除函数根据返回的结果信息进行相应操作，即若删除成功则显示成功信息弹窗，若删除失败则显示失败信息弹窗。其删除函数关键代码如下图所示：

```
$.ajax({
    url: '{% url "article:del_article" %}',
    type:"POST",
    data: {"article_id":article_id},
    success: function(e){
        if(e=="1"){
            parent.location.reload();
            layer.msg("has been deleted.");
        }else{
            layer.msg("删除失败");
        }
    },
})
```

2.5.4 处理证书信息

处理证书信息包括修改证书信息、显示证书信息、删除证书信息。其中修改和删除证书信息只有管理员可操作。

(1) 修改证书信息

根据上文中对于证书信息在管理员界面的显示，可直接点击证书标题进入证书修改页对证书信息进行修改。修改函数是 Django 内置函数，不必编写代码，直接使用即可。证书表一共有两种，分别是基础证书表（Certificate tbls）和 VS 证书表（Certificate vs）。对于后台列表的功能展示可直接在相应 App 中的

admin.py 文件中添加即可。

(2) 显示证书信息

显示证书信息包括显示证书表中的信息和显示 VS 证书表中的信息，其操作是相同的。这里以显示证书表中的信息为例，用户可以通过证书分类的列表浏览证书名称，并可点击感兴趣的证书名称进入相关证书详情页。其实现方法非常简单，只需调用后台数据呈现至前台进行排版即可，其关键代码如下：

```
def cert_list(request):
    lists = CertificateTbl.objects.all()
    return render(request, "list/list.html", {"lists": lists})
def cert_result(request, result_id):
    result = CertificateTbl.objects.get(cert_id=result_id)
    return render(request, "list/content.html", {"result":result})
```

对于证书列表与证书详情页的对应关系，利用正则表达式“?P<result _id>\d”的方式表示，这里匹配的是一个数字字符，并且该数字字符赋给 article_id 参数，传给 views.cert_result(),这样 views.cert_result()函数除第一个参数 request 外，还有一个参数接收 result_id 传入的值。URL 配置部分的具体代码如下：

```
url(r'(?P<result_id>\d+)/$', views.cert_result, name="cert_result")
```

证书除了列表显示外，还可以通过搜索进行显示。若用户输入的证书关键词和列表中若干证书的名称相符，则返回的详情页为搜索所匹配的第一个证书的详情页。搜索视图关键代码如下：

```
q = request.GET.get('q')
if q:
    return render(request, 'list/errors.html', {'error_msg': error_msg})
result = CertificateTbl.objects.filter
(Q(cert_name__contains=q) | Q(cert_ename__icontains=q)).first()
```

(3) 删除证书信息

删除证书信息只有管理员可操作。此功能是 Django 默认函数，因此不必编写代码，直接使用即可。其删除证书页面如图 2-5 所示。

图 2-5 删除证书页面

2.5.5 处理问卷信息

(1) 计算得分

个性决策查询是给用户一份考证评估问卷，并通过本组成员根据建模评价模型所得出的问卷分数进行评价。估测得分在系统前端完成，对于不同的证书类别有不同的分值，因此需要利用后台证书编号来查找对应的分值，这里对于不同类型的不同分值以键值对的形式表示。由于前五道题目与证书类型相关所以在编码上相对繁琐。以前五道题的编码为例，其具体代码如下：

```
var type = document.getElementById("test").value;
for(var j = 1; j<=5; j++){
    var item = document.getElementsByName('optionsRadios'+j.toString());
    for (var i = 0; i < item.length; i++) {
        if (item[i].checked == true) {
            if (type == 'a'){
                score = score + parseFloat(item[i].value);}
            else{
                k = i+1;
                str=j.toString() + k.toString() + type.toString();
                score = score + parseFloat(differ[str]);}
        }
    }
}
```

其中，type 是取得证书类型列表的值（a 为基础证书、b 为职业证书、c 为能力证书），for 循环是因为前五道题对于不同类型的证书其每个选项的分值是不同的。因此需要通过选项编码来判断使用哪一种分值，如果选中选项的类型为基础证书类，则 score 为其 value 项的值（需要强制类型转换成浮点型），若类型为其他两种，则需要根据选项编码找出对应的分值。之后的题目则均根据选项的 value 值进行累加即可。

计算得分后，在前台会显示得分情况以及相关建议内容，即根据不同分数段，显示不同建议。为了能够找出大致的分数段并对不同分数段给出合适的建议，本组成员抽查了 50 名大学生志愿者填写问卷并记录其评分。通过分析志愿者的问卷结果数据，本组成员拟定用户得分在 0~55 之间，建议为证书难度较大，不建议近期报考；得分在 56~69 之间，建议为多加复习，合理利用时间；得分在 70 及以上的，建议为努力备考，成功很有希望。

3 总结与展望

本文介绍了大学生考证信息咨询与决策系统的 PC 端详细设计与系统实现。在 PC 端系统详细设计部分，本文分别介绍了编码设计、用户界面设计以及系统功能模块设计的详细设计方案，即在编码设计中利用数字码与混合码形式编码；在用户界面设计中本文分别对用户登录界面、证书列表界面、分享展示界面等界面进行界面设计图展示与阐述；在系统功能模块设计中采用层次化模块结构图对系统总体功能模块进行规划，并且以 IPO 图对每个细分的功能模块进行表述。在 PC 端系统代码实现部分，本文分别介绍了本系统的开发方法、开发工具、系统开发基础配置、系统基础功能实现及系统功能模块实现，即本文对结构化开发方法、B/S 结构、MVC 架构、编程工具、Web 服务器及开发平台进行了简单描述，并详细描述了系统的基础功能实现，包括数据库实现、后台数据绑定、服务器搭建及项目部署，最后本文结合 HIPO 图的模块划分对系统的功能模块实现进行了重点阐述，包括实现过程描述、关键代码展示及部分前端截图展示。

本文对大学生考证信息咨询与决策系统的实现提供了新的思路与方法，具有较强应用价值。由于本系统目前处于初步实现阶段，且受开发周期的限制，使得一些功能模块的实现并不完善，一些技术环节也未能深入，因此仍有部分问题需要进一步研究。

（1）本文在个性决策部分的问卷评估得分中采用层次分析法获取权重，进而通过权重百分比估算选项分值来获得问卷选项的具体分值。在以后的研究中，我们可以考虑利用卷积神经网络来建立机器学习模型，利用用户调查数据训练模型，从而智能化的做出考证辅助决策。

（2）本文在个人空间模块中应该加入适量的数据可视化展示，如利用直方图展示用户进行考证决策分析后的成功率；利用折线图展示用户每月的文章发送及浏览信息的情况；利用饼图展示用户的社区行为（以便用户找到与自己行为类似的用户）等。

（3）本文的考证信息咨询部分需要进一步完善，如可利用爬虫技术广泛收集证书信息与真题信息；增设日历标注和时间提醒功能，以便用户能合理规划备考时间；增设证书“加关注”功能，以便用户能准时收到与该证书相关的经验分享信息与图文信息。

结束语

通过小组六位同学的不懈努力与朱老师的悉心指导，本小组设计与开发的大学生考证信息咨询与决策系统得以顺利完成。毕业设计作为我们在大学四年中学习能力与实践能力的一次重要考核，它要求我们利用大学所学的知识来解决实际问题或者对现有成品进行创新。我们对现实中存在的“考证热”问题进行思考，针对大学生在考证中普遍存在的问题，设计出“考证一对一”个性化服务的大学生考证咨询与决策系统。在设计开发过程中，小组成员之间分工明确，具有较强的团队意识，并能够运用我们所学的知识共同分析、设计、讨论、改进本系统，使我们的系统不断地完善，达到了预计设计的要求。

目前系统已经具备了初步的结构及功能，并达到了预期的目标。但是由于我们系统开发经验不足，水平能力有限，所以开发的系统还存在较多的问题与不足。这也说明我们今后还都需要继续学习不断积累经验来提升自己的能力。

在本项目中，我主要负责的是 PC 端的系统详细设计与系统实现这两个部分，期间我回顾了计算机相关教程，查阅了大量资料，并通过总结别人的经验来不断地完善我们的系统功能，令我们的系统更加贴近实际。在实现系统功能的过程中我遇到了很多困难，不过通过小组成员间的相互鼓励和自己静下心来钻研使得问题都得以解决。在这次项目中，我们领悟了团队间分工合作的重要性，每一个重要的项目节点都离不开小组成员的沟通交流，以便使每个人的任务都能够不偏离预期的顺利进行。并且，这次项目让我们明白了实践的重要性，明白了想法不能只浮于理论，要用实践检验真知。唯有动手实践了，才能在实际问题中不断提升自我。

最后，再次感谢在系统开发过程中给予我们大力支持及宝贵意见的朱红灿老师。朱老师在我们开发系统中给了我们重要的帮助，使得我们少走了许多弯路。同时朱老师在我们开发系统当中还教会了我们许多做人处事的道理。无论是在知识方面，还是在理论实践方面都让我们受益匪浅。

参考文献

[1] 赵玲玲.大学生“考证热”现象的理性思考[J].宿州教育学院学报,2017,20(2):72-73.

[2] 余秀园,张琳,田雅丽,郭彤.职业技能鉴定考试及证书的在线管理系统开发[J].教育教学论坛,2016(23):236-237.

[3] 邢计亮.基于B/S模式的远程教育考试系统设计与实现[D].河北:河北科技大学,2014.

[4] 黄梯云.管理信息系统(第6版)[M].北京:高等教育出版社,2016:135-136.

[5] 肖红玉,贺辉,陈红顺.在线评测教学辅助系统设计[J].计算机技术与发展,2017(11):141-145.

[6] 江柳.基于Django的博客系统开发研究[J].电脑编程技巧与维护,2016(13):14-15.

[7] 李文俊.基于Web的法院车辆管理系统的设计与实现[D].福建:厦门大学,2015.

[8] 宫薇薇,齐向春,裴世廉.Python与R语言混合编程方法的研究和应用[J].计算机应用与软件,2018(1):28-31.

[9] 董伟明.Python Web开发实战[M].北京:电子工业出版社,2016:127-135.

[10]仇小花,秦栓栓,邱果.基于WEB开发中的XML与JSON数据传输格式研究[J].信息技术与信息化,2017(4):123-125.

[11]张康.基于Nginx的在线教育平台架构优化研究[D].北京:北京工业大学,2016.

[12]薛华成.管理信息系统(第6版)[M].北京:清华大学出版社,2012:405-406.