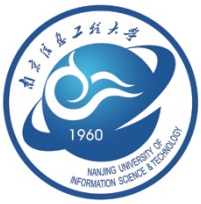
南京信息工程大学

# 本科生毕业论文(设计)




题　　目　基于 iOS 的"云类识别"移动学堂设计与开发

学生姓名　丁健

学　　号　20121308077

学　　院　计算机与软件学院

专　　业　计算机科学与技术

指导教师　王保卫

二〇一六 年 五 月 二十 日

# 声　　明

本人郑重声明：

1、 持以“求实、创新”的科学精神从事研究工作。

2、 本论文是我个人在导师指导下进行的研究工作和取得的研究成果。

3、 本论文中除引文外，所有实验、数据和有关材料均是真实的。

4、 本论文中除引文和致谢的内容外，没有抄袭其他人或其他机构已经发表或撰写过的研究成果。

5、 其他同志对本研究所做的贡献均已在论文中作了声明并表示了谢意。

作者签名：_____

日　　期：_____

# 目 录


1 绪论 ..... 1

1.1 研究背景 ..... 1
1.2 国内外研究现状 ..... 1
1.3 研究目的及意义 ..... 1

2 系统相关技术 ..... 2

2.1 iOS 操作系统 ..... 2
2.2 Xcode ..... 2
2.3 Objective-C 语言 ..... 2
2.4 设计模式 ..... 3

3 系统概要设计 ..... 3

3.1 需求分析 ..... 3
3.2 系统模块设计 ..... 4
3.3 系统流程综述 ..... 5
3.4 数据库设计 ..... 6
3.4.1 E-R 图 ..... 6
3.4.2 数据库表设计 ..... 7
3.5 Web Service 接口设计 ..... 9

4 详细设计与实现 ..... 11

4.1 UI 设计与实现 ..... 11
4.2 网络请求 ..... 12
4.3 数据库的云端同步与本地缓存策略 ..... 13
4.4 形态学分类法界面的实现 ..... 16
4.5 练习考试界面主流程的实现 ..... 17
4.6 错题的录入与界面展示 ..... 19
4.7 登录注册功能的实现 ..... 20
4.8 练习模式的设计 ..... 22

5 系统测试 ..... 23

5.1 功能测试 ..... 24
5.2 健壮性测试 ..... 24

6 总结与展望 ..... 25

参考文献： ..... 25

致谢 ..... 27

# 基于 iOS 的"云类识别"移动学堂设计与开发

丁健

南京信息工程大学计算机与软件学院，南京 210044

**摘要：**大气探测实验实习（以下简称大探实习）是大气科学本科生的主干课，大探实习中有一部分重要内容是“云的观测”，云的分类是云的观测中的难点。由于地域和天气的限制，目前相关高校对云的分类的教学和考核相对单一且不深入。为此，建设“云类识别”实验系统迫在眉睫。本文针对这一需求设计并实现了基于 iOS 的"云类识别"移动学堂。该系统通过网络实时获取云端题库数据，有效地进行“云类识别”的学习与考试，满足了高校多样化、具体化教学与考核的要求。本系统的出现大大扩充了云的资料库，为高校的课程设计积累丰富资源。该系统是采用 Xcode 作为 IDE、Objective-C 作为开发语言的 C/S 结构的 iOS 平台系统。

**关键字：**iOS；云类识别；多平台；学习软件

# The Design and Implementation of Cloud identification Experimental simulation system Based on iOS

Ding Jian

School of Computer and Software，NUIST，Nanjing 210044，China

**Abstract**：Atmospheric Exploration experiment and practice (hereinafter referred to as the major exploration practice) is the main course of atmospheric science undergraduate, great part of the major exploration practice is the “cloud observations”. Cloud classification is a difficult point in the observation of cloud. Due to geographical and weather constraints, the current cloud classification of teaching and assessment in related universities is relatively simple and not in depth. The construction of a cloud recognition experiment system is imminent in order to achieve this goal. This paper designs and implements the cloud identification mobile system based on iOS in view of this requirement. The system through the network real-time access to the cloud database data, effectively carry out the "cloud classification" of the study and examination,which meets the requirements of diversified and specific teaching and assessment in colleges and universities. The emergence of the system has greatly expanded the database of the cloud, and accumulated rich resources for the curriculum design of colleges and universities. The system is a C/S platform system based on iOS using Xcode as IDE and Object-C as the development language.

**Key words**：iOS；classification of clouds；multi platform；learning application

# 1 绪论

## 1.1 研究背景

大气探测实验实习（以下简称大探实习）是大气科学本科生的主干课,大探实习中有一部分重要内容是“云的观测”，而这部分的实习目前只局限于课堂图片介绍，和实习当天当时的云状云量及云高。且由于地域特征，有部分种类的云在本地区极少出现。“云的分类”是云的观测中的难点。在教材或试卷中，对云识别的教学和考核，也仅通过单一的文字描述。即便有图片，也由于印刷成本等原因，是黑白模糊的，不能充分体现云的外形特征。

如果通过手机显示云的外形和结构特点，云的演变过程，不仅视图能够直观清晰，对云的识别也是充分全面的。随着平时对云的案例的积累，将大大扩充云的资料库，为课程设计积累丰富资源。

## 1.2 国内外研究现状

关于这门学科，在各开设“大气科学”专业的高校一般都有授课，这门学科内的“云类识别”是重要的一项知识点，但由于知识需涉及大量高清图片，所以一般纸面知识的学习与测试并不能满足师生的需要，而实际学习又受天气，环境等因素的影响。而就此学习方法，还没有出现一个比较好的办法，即使有，也局限于小型系统，不全面。

纸质的资料已经完全不能满足老师同学对大量高清云类图的需求了，而在智能手机如此普及苹果手机盛行的年代，开发出一个能满足师生学习高清云类图的 iOS 平台的系统已经迫在眉睫。这样的系统需要充分考虑到当下天气、环境等因素对师生学习“云类识别”的影响，要保证偶尔老师不在的情况下，该系统也能带着同学们进行多样化的云类学习。

而本系统的设计就是建设一个多平台、知识全、方式科学的自主学习平台，弥补当前学习“云类识别”的客观条件的不足。

## 1.3 研究目的及意义

“云类识别”虚拟仿真实验系统是提供云类识别学习的手段和方法，为新型的大气探测实验实习的顺利进行奠定了良好的软件基础。传统的大气探测实验实习利用纸面知识进行学习，实习老师会在合适的天气带领学生去大气实习站去现场观测云，不过由于实习课程时间相对固定，而不同的云出现的时间却不固定，并且很多云只在特定的地区才会出现，因此学生的学习难度与学习成本都非常大，老师的教学质量也因为外在的原因得不到保障。本设计采用云端录入大量高清云类图及其介绍，系统通过网络请求的方式实时获取云端的最新云类图。因其可随身携带、学习效率高、云图高清、不受地理因素影响等特点，能够很好的解决当前

线下云类图学习的不足。本系统还可与相应的 web 系统协同使用，当云类图的资料或者题库需要更新时，可通过相应的 web 系统进行新题库的录入与云类图学习资料的增删改，提交修改之后，所有使用本系统的学生只需点击同步数据按钮，即可将本系统的所有资料同步成云端最新的数据，因此，在资料的更新与正确性方面，本系统完全不输纸质资料。本系统的开发，更为日益增多的 iPhone 用户提供了一个更加高效便捷的云类图学习平台。

# 2 系统相关技术

## 2.1 iOS 操作系统

iOS 操作系统是苹果公司研发并推出的基于 Unix 的移动设备操作系统。该系统处理器速度极快、效率极高。最新的 iOS 版本是 2016 年 05 月 04 日发布的 9.3.2 版本。本文采用的是 2015 年 12 月 16 日发布的 9.2.1 测试版。

## 2.2 Xcode

Xcode 是苹果公司官方推出的用户界面统一、高效的集成开发工具，从 Xcode3.1 版本开始可用于开发 iPhone 应用。

本系统的开发采用的是 Xcode7.1 版本，它拥有操场写作、App 瘦身、全新的调试模式和性能分析特性、UI testing、免费的设备调试、崩溃日志等新性能。下面简单介绍几个 Xcode 7 新特性：

（1）App 瘦身

Xcode 7 现在可以开发多达 3 个平台的应用, 设备种类很多并且屏幕分辨率等规格各不相同。针对同一个 APP, 你可以利用 Xcode7 和 App Store 只下载该设备用到的资源。

（2）全新的测试

Xcode 7 里面, XCTest 框架加入了一个主打的特性: UI testing. UI testing 以 XCTest 现存的 API 和概念的一个扩展的方式实现, 已经熟悉 Xcode 的测试功能的开发者很容易上手。全新的测试包括 UI 录制、正确性和性能测试、代码覆盖率测试以及 Xcode Server。

（3）免费的设备调试

在 Xcode7 里面, 不再需要购买开发者以及进行繁琐的设置, 你就可以在任意的设备上进行开发和调试了。只需要注册一个 Apple Id[1]。

## 2.3 Objective-C 语言

尽管苹果大力推荐开发者们采用全新的 swift 语言进行 iOS 项目的开发，但是 Objective-C 语言的地位短期内是无法被撼动的。Objective-C 是在 C 语言的基础上增加了面向对象思想的

iOS 开发基本编程语言。Objective-C 兼具了 C 语言、C++的优点，同时还具有独特的内存管理机制、消息传递等个性化的特质。Objective-C 区别于其他语言的一大特点是没有垃圾回收机制，不过其 ARC 自动内存管理机制将开发者从内存管理的深渊中完整的解放了出来[2]。

### 2.4 设计模式

本系统的主体架构采用 MVC 设计模式，对数据库的管理、用户信息的管理采用了单例设计模式、代理设计模式也广泛用于本系统各个界面逆向跳转时的传值、注册登录的代理回调。

（1）MVC 设计模式：

MVC 设计模式也是苹果官方推荐的系统架构思路，尽管大型项目复杂的业务逻辑会使控制器变得臃肿、难以维护，但是 MVC 设计模式在面对本系统这样的中小型项目时表现十分不错。

（2）代理设计模式：

代理设计模式广泛用于对象之间的交互通信。Cocoa touch 很多地方都采用了代理设计模式，比如说 UITableView 的 delegate 方法，开发者通过实现关键的代理方法来告诉 UITableView 要显示几行的数据等。本系统在注册、登录功能中也使用到了代理设计模式进行登录／注册完成时的页面跳转、逻辑处理[3]。

（3）单例设计模式：

单例模式是指某个单例类从程序开始运行到程序的生命周期结束，只有一个该类的实例。本系统的数据库管理类、用户信息管理类都采用了单例设计模式以达到全局信息一致的目的。

## 3 系统概要设计

### 3.1 需求分析

基于 iOS 的"云类识别"移动学堂设计的目的是为了让学生可以随时随地、不受天气、地域条件限制的去学习云的分类。

为了满足学生的需求，本系统至少需要做到界面友好，操作简单、功能完善这几个要求。

用户可以根据自己的实际情况，选择不同的练习方式进行云类识别的学习与考试。从大的功能上来看，基于 iOS 的"云类识别"移动学堂可以分为两大板块：云类认识学习板块和云类练习考试板块：

（1）云类认识学习板块:通过不同云的分类入口进入学习各类云，主要通过文字介绍与图片展示相结合的方式。

（2）云类练习考试板块：题型以选择判断为主，用户可以左右滑动切换上一题下一题，也可点击按钮跳转到指定题号的题目。每一题答完都会提示是否正确，如果正确则自动切换下一题，如果错误则展示试题详解，并标示正确选项。进一步细分为练习模式和考试模式。

以下是各个练考模式的功能需求概要：

（1）顺序练习：用户选择按照题库的顺序有序进行练习。

（2）章节练习：用户可根据自身学习情况，选择不同的章节进行针对性练习。

（3）随机练习：用户的学习到了后期阶段的时候，需要将题库打乱，随机进行练习，以达到更好的学习效果。

（4）错题练习：在各种模式下，用户做错的题目均会被记录到错题集中，用户可选择错题练习进行针对性的巩固。错题练习界面需要提供删除错题按钮，以便用户自行删除已经训练多次、熟记于心的错题。

（5）模拟考试：按比例从各个章节随机抽取相应比例的题目作为考卷，能够真实有效地检测出用户的学习成果。

### 3.2 系统模块设计

通过对用户需求的分析，基于 iOS 的"云类识别"移动学堂大致可以分为云学、云图、我三大模块。其中，云图可细分为顺序练习、统计、章节练习、模拟考试、收藏、随机练习、错题练习 7 个小模块。系统模块设计如图 3.1 所示：

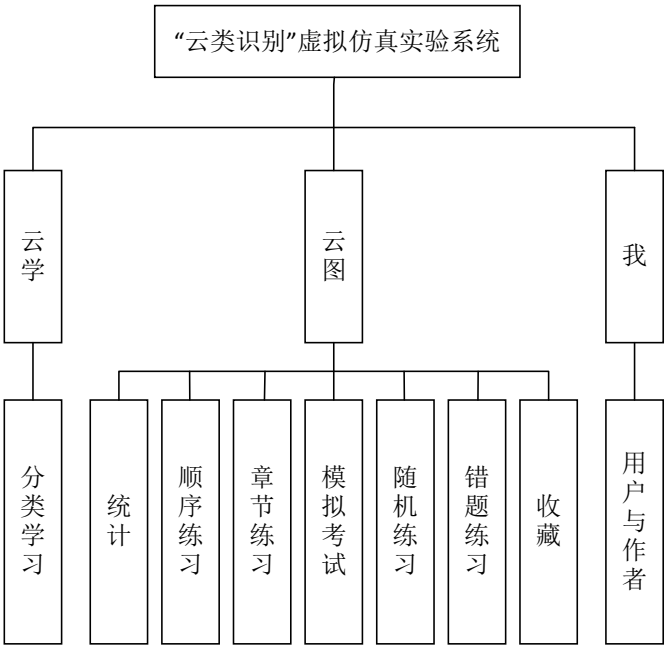



图 3.1 系统模块设计

用户点击云图应用，默认进入系统首页——云学，点击下方 tab 可任意切换至云图、我的界面。如果用户想要通过图文方式进行初步的学习，可在云学中选择相应的云来学习。如果用户想要进一步深入学习，可切换至云图 tab，自由选择不同的练习方式，或者选择模拟考试来检测自己的练习成果。我的页面主要是用户信息和开发者本人的一些信息。

云学的内容采用本地静态数据的方式进行存储，后期数据的更新可以通过版本迭代来实现。云学中的内容为图文介绍不同类型的云，数据相对稳定，属于不太会变更的科学知识，因此，采用本地存储的方式可以在保证质量的前提下更加快捷、高效地进行开发。

云图中的题库采用了网络请求＋本地缓存相结合的方式，既保证了题库的实时更新，也尽可能的降低了用户的等待时间与流量的消耗。具体做法后续会详细说明。

### 3.3 系统流程综述

针对上面的需求分析，系统最终得以完整实现，相关使用流程如下：

用户点击桌面云图 icon，启动程序，第一个页面为启动图，用于展示云图应用相关开发者信息。展示一秒后进入云学界面并自动播放云的介绍音频。

主菜单依次分为云学、云图、我三个 tab。点击可自由切换到对应的 tab。

点击云学界面，除了可以播放云的介绍的录音，还可以按照形态学分类法或者国际学分类法分别去学习云类识别。形态学分类主要是由 webView 加载服务端的 html 文件来展示相关分类云图。国际学分类法则更加细致的进行了分类，分为层积云、积云、积雨云等多个分类，点击不同的分类，还会有此分类下的更细致的云类，再点击则进入对应云的展示学习界面。此处形态学分类和国际学分类采用了不同的展示方式，一是因为国际学分类更加细致，二是希望通过不同的展现方式来让云的学习更加有趣、多样。

点击云图界面，可以选择不同的练习模式或者考试模式，点击章节练习时用户会被要求选择所要练习的章节号。错题练习的题目是用户在所有模式下答错的题目的集合，并且在错题模式下可以删除已经完全熟悉的错题。选择用户需要的答题模式进入答题，左右滑动可切换上一题下一题，点击底部的工具栏可以弹出所有的题目编号，点击题目编号可切换至任意题目。每一题是否作答以及答对与否都在弹出的视图里得以体现。用户可以对虽然答对但是不熟悉的题目进行收藏，并且可以在云图的收藏界面查看这些不熟悉的题目。答完每一题都可在工具栏实时查看答对、答错以及总答题量。答对自动右滑下一题，答错则用绿色标出正确答案，并展示出详细的解释，用户也可在未作答的情况下，点击右上角的题解按钮显示本题详解。除了练习、考试两大重要模块，云图板块下还有收藏、统计两大不可小视的功能。收藏功能如上所说，可以查看用户在每次答题过程中收藏的所有题目。统计功能主要是展示用户每次完整答题的得分情况，以便用户进行对自己的学习情况进行分析总结。

点击“我”界面，可以点击用户头像进行用户帐号的登录、退出、切换。点击关于云图可以查看本系统的相关简介。点击关于作者可以查看作者的相关信息。

本系统的相关操作流程如图 3.2 所示：

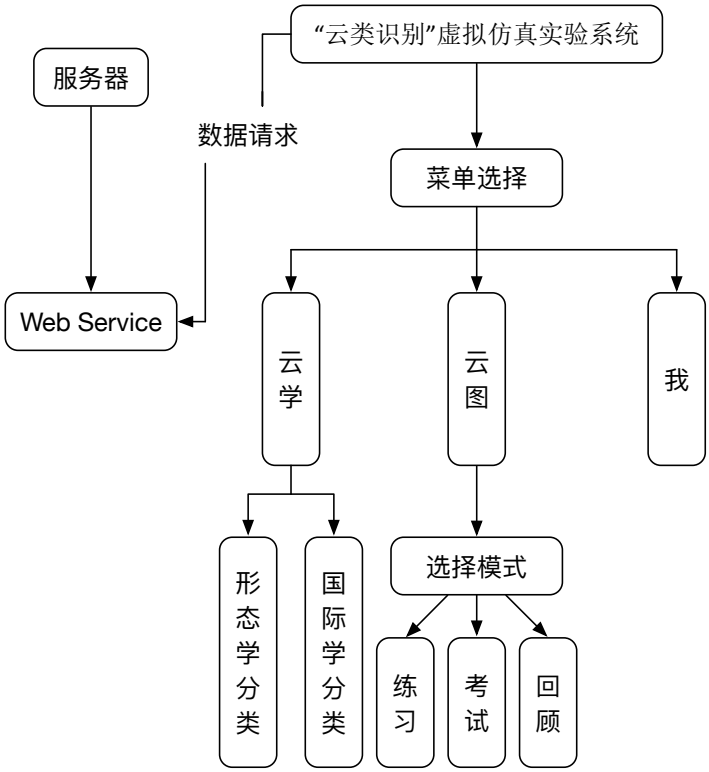



图 3.2 系统操作流程图

## 3.4 数据库设计

本系统采用了手机端常用的 sqlite3 数据库。数据库的相关表均为提前和中间层约定好格式、通过网络请求获取 json 数据后用模型对象存入本地 sqlite 数据库。以下将详细介绍数据库是如何设计的。

### 3.4.1E-R 图

本系统中的实体有：用户实体、题库实体、错题实体、收藏实体和统计实体，云图系统的 E-R 图如图 3.3 所示：

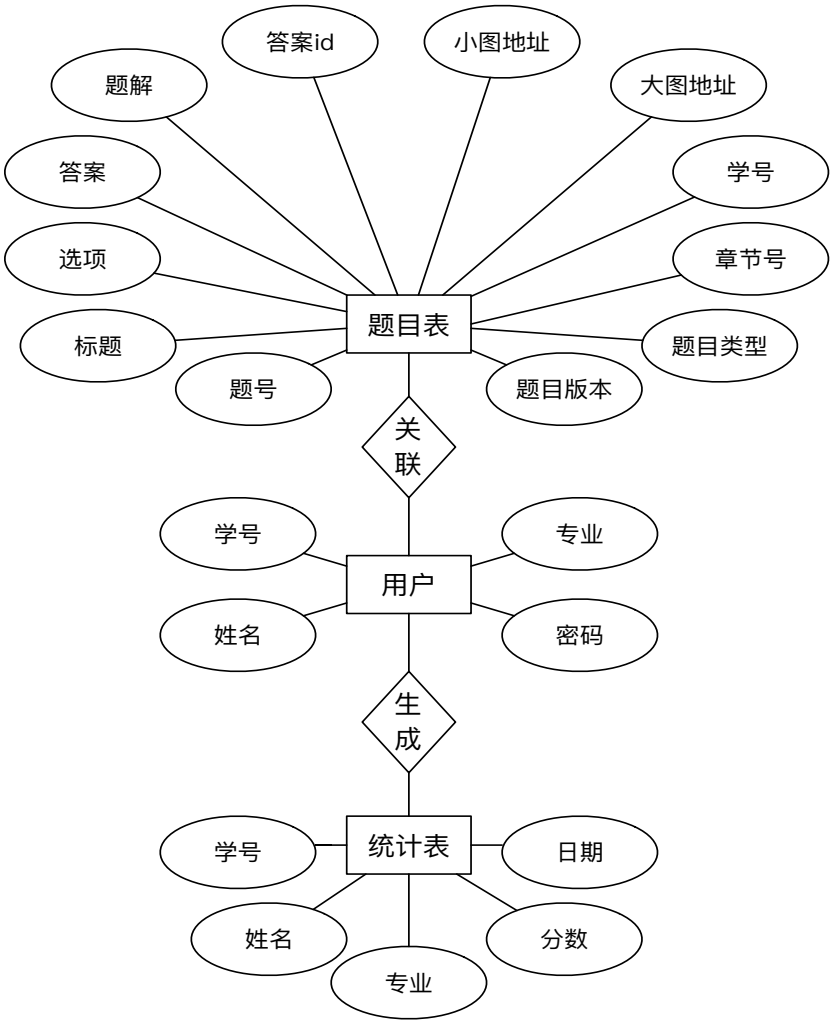



图 3.3 云图系统 E-R 图

3.4.2 数据库表设计

本系统共设有数据库表 4 张，分别是用户表、云图题库表（错题表、收藏表）、交卷记录表和题库版本号表，用户表见表 3.1，用户表中共有四个字段分别代表了用户的学号、密码、姓名、专业。

表 3.1 用户表

| 列名 | 数据类型 | 允许空 | 备注 |
|---|---|---|---|
| StuNum | varchar(50) | 否 | 学号（主键） |
| StuName | varchar(50) | 否 | 用户名 |
| StuPwd | varchar(50) | 否 | 密码 |
| StuMajor | varchar(50) | 是 | 专业 |

题库表共设有 14 个字段，由题号唯一标识每条题目，除了基本的题型、标题、选项、答案、题解之外，还特地设置了大图、小图字段用于显示题目相关的图片、章节号用于章节练习模式时区分章节使用、题目版本号用于区分不同版本的题目。题库表见表 3.2：

表 3.2 题库表

| 列名 | 数据类型 | 允许空 | 备注 |
|---|---|---|---|
| QNum | int | 否 | 题号（主键） |
| QTitle | varchar(50) | 否 | 标题 |
| QOption1 | varchar(50) | 否 | 选项 1 |
| QOption2 | varchar(50) | 否 | 选项 2 |
| QOption3 | varchar(50) | 是 | 选项 3 |
| QOption4 | varchar(50) | 是 | 选项 4 |
| QAnswer | int | 否 | 答案 |
| QExplain | int | 是 | 解析 |
| QRightNum | int | 否 | 答案 id |
| QLargeImgUrl | int | 是 | 大图片 |
| QShortImgUrl | int | 是 | 小图片 |
| QSection | int | 否 | 章节号 |
| QType | int | 否 | 题型 |
| QVersion | int | 否 | 题目版本号 |

模拟考试成绩表共设有 5 个字段，分别代表了学号、专业、姓名、日期、分数。此表的意义在于给老师以及学生查看答题情况。用户在答题时选择交卷后会将本次答题数据以 json 的格式传给服务端。考试成绩表见表 3.3：

表 3.3 考试成绩表

| 列名 | 数据类型 | 允许空 | 备注 |
|---|---|---|---|
| StuNum | varchar(50) | 否 | 学号（主键） |
| StuMajor | varchar(50) | 是 | 专业 |
| StuName | varchar(50) | 是 | 姓名 |
| Date | varchar(50) | 是 | 日期 |
| Score | varchar(50) | 是 | 分数 |

题库版本表共设有两个字段，一个是版本号，用于对比本地的数据库版本和服务端的数据库版本是否一致，不一致则更新本地数据库。二是更新方式，当数据库需要更新时，由这个字段来确定是全量更新还是增量更新。数据库版本表见表 3.4：

表 3.4 数据库版本表

| 列名 | 数据类型 | 允许空 | 备注 |
|---|---|---|---|
| VersionNum | varchar(50) | 否 | 版本号（主键） |
| UpdateType | varchar(50) | 否 | 更新方式（1 全量、2 增量） |

## 3.5 Web Service 接口设计

本系统共涉及到 4 个 Web Service 接口，分别是云图用户注册接口 StudentRegister、云图用户登录接口 StudentIsExist、云图题库更新接口 DownLoadQuestion、上传答题成绩接口 ImportScore。本系统的 Web Service 采用了轻量级的 json 进行云图系统与 service 端的数据交互。

（1）StudentRegister

云图用户注册接口采用了 POST 请求方式，学号 stuNum 为必传参数，当本系统向 web service 发起用户注册请求时，接口根据本系统所传关键注册信息 stuNum 判断服务端数据库中是否有相应学号的用户，若没有则向学生表中添加相应的用户并通知本系统注册成功，相反则通知本系统注册失败。云图用户注册请求参数信息见表 3.5：

表 3.5 云图用户注册请求参数表

| 请求参数名 | 含义 | 是否必传 | 请求方式 |
|---|---|---|---|
| stuNum | 学号 | 是 | POST |
| stuName | 姓名 | 否 | |
| stuPwd | 密码 | 否 | |
| stuMajor | 专业 | 否 | |

（2）StudentIsExist

云图用户登录接口采用了 POST 请求方式，学号 stuNum 和密码 stuPwd 均为必传参数，当本系统发起云图用户登录请求时，接口根据本系统所传登录信息 stuNum 和 stuPwd 判断服务端数据库中是否有相应学号的用户，若有则通知本系统登录成功，本系统提示用户登录成功并做相应跳转处理。云图用户登录请求参数信息见表 3.6：

表 3.6 云图用户登录请求参数表

| 参数名 | 含义 | 是否必传 | 备注 |
|---|---|---|---|
| stuNum | 学号 | 是 | POST |
| stuPwd | 密码 | 是 | |

（3）DownLoadQuestion

云图题库更新接口采用了 GET 请求方式，系统本地缓存题库版本号 versionNum 为必传的唯一的请求参数，当本系统发起云图题库更新请求时，接口根据本系统所传题库版本号与服务器数据库题库表中版本号相比，若本系统所传题库版本号较小则需要返回最新的云图题库给本系统。考虑到题库只是增加了一些数据而原先的数据没有变化的情况下，若此时采用全量覆盖更新本系统云图题库的方案，则会造成用户手机流量的不必要的浪费。经过对多个知名学习答题类 app 的研究，本系统最终采用全量更新＋增量更新相结合的方案对云图题库进行更新。更新方案如下：

DownLoadQuestion 接口设有 updateType 字段，若服务器端最新云图题库只是增加了一些新的题目，则 updateType 字段返回 2 即告知本系统进行云图题库增量更新；若服务器最新云图题库有较大的改动，则 updateType 字段返回 1 即告知本系统进行云图题库全量更新。

服务端云图题库的更新通过相应 web 系统的教师端进行。教师端提供增量更新、全量更新的单选按钮，默认选中全量更新。教师针对云图题库的更新和变动情况勾选增量更新或者全量更新按钮，教师每更新一次云图题库，云图题库版本号自动加一存入服务器数据库对应的云图题库版本表中，同时本次更新的更新方式 updateType 也会随之存入云图题库版本表。

表 3.7 云图更新题库请求参数表

| 参数名 | 含义 | 是否必传 | 备注 |
|---|---|---|---|
| versionNum | 学号 | 是 | POST |

（4）ImportScore

云图题库更新接口采用了 GET 请求方式，实现方式与原理和登录、注册类似。

表 3.8 云图答题成绩请求参数表

| 参数名 | 含义 | 是否必传 | 备注 |
|---|---|---|---|
| stuNum | 学号 | 是 | GET |
| stuName | 姓名 | 是 | |
| stuMajor | 专业 | 否 | |
| date | 日期 | 否 | |
| score | 分数 | 是 | |


图 3.4 成绩统计与个人信息界面

# 4 详细设计与实现

## 4.1 UI 设计与实现

目前 iOS 提供了三种方式用于 UI 实现，最原始的是纯代码实现，现在会有一些比较保守的、老派的开发者仍然采用这种做法[4]。第二种是 xib 实现，这是苹果公司为了降低开发者绘制 UI 的成本而提供的技术。第三种是 StoryBoard 故事板实现，storyboard 也是苹果官方大力推崇的开发方式，所有视图的逻辑关系都在一个可视化 xml 文件里得以清楚的显示，跳转逻辑关系非常清晰。不过由于商业项目大多是多人协作开发，采用 storyboard 进行开发的话，特别容易在提交代码的时候发生不必要的冲突，无意间增加了额外的开发成本，因此，尽管苹果大力推荐，storyboard 在大型商业项目里并没有特别流行。

本系统综合考虑了三种方式的优缺点

纯代码：　　优点：可控性强，健壮性强，面对产品经理频繁的要求变动能够应对自如。

　　缺点：与时代脱节，代码量大，阅读成本大。

xib：　　优点：开发效率高，多人协作方便。

缺点：xib 中的设置往往会被代码所改变，不易维护。

Storyboard：优点：逻辑清晰，开发高效

缺点：多人协作消耗大、不易被大团队技术负责人接受。

采用了 xib 加 storyboard 相结合的方法进行软件的开发。

本系统结构采用当下最流行的 UITabBarController+UINavgationController 的结构。由 UITabBarController 来控制三个主视图的切换，而每个主视图内部视图的跳转则通过 UINavgationController 来控制。本系统共涉及到如下几个视图控制器：云学控制器 ViewController、云图控制器 YTQuestionViewController、“我”控制器 YTMyViewController、形态学控制器 WebViewController、国际学控制器（YTGlobalKindViewController、YTGlobalDetailViewController）、答题控制器 YTQuestionViewController、注册控制器 RegisterViewController、登录控制器 LoginViewController、关于作者控制器 AboutAutherViewController。

界面间的层次关系如图 4.1 所示：

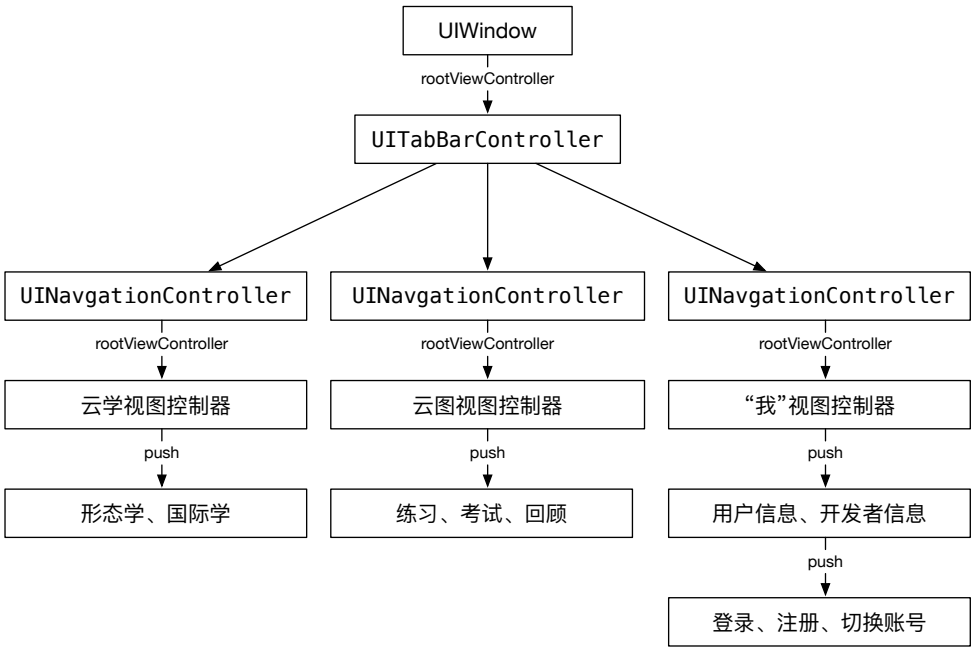



图 4.1 系统结构层次图

## 4.2 网络请求

网络请求也是本系统的一大核心功能，由于原生的 http 请求写起来非常繁琐且容易出错，实际开发中通常会采用封装好的第三方网络框架进行网络数据请求。在 iOS 开发领域，AFNetWorking 就是被广大开发者所采用的第三方网络框架[5]。此框架简单易用，以 get 请求为例，完成一个请求只需要完成如下三个步骤：

给 AFN 提供的方法传入 URL。

给 AFN 提供的方法传入 request 参数。

在 AFN 的成功、失败的回调里做相应的数据处理。

通常我们在开发过程中只会用到 GET、POST 方式的请求，不过 AFN 也提供了诸如 PUT 之类的请求方式。

### 4.3 数据库的云端同步与本地缓存策略

数据库相关的问题为本系统的核心技术问题，本系统在提交到 app store 的时候，会内置一个版本号为 0 的本地数据库 yuntu_zero.db3，在最初的时候此数据库的数据与 service 端的数据完全一致。在系统内置本地数据库的原因是，在本系统上线初期，云图题库不会有什么变动，因此，在用户第一次使用本系统的时候要尽量避免下载云图题库这样的耗时操作，以防给用户不好的第一印象，最终会导致用户的留存率大大降低[6]。

既然内置了一个本地数据库，那么在后期云端题库发生变动的时候肯定会涉及到一个题库数据同步的问题。本系统对此的解决方案为，在云图首页右上角显示一个“同步题库”的按钮，当用户点击按钮的时候，本系统会去发请求给 web service，告知 service 端本地数据库的版本号，如果 service 端数据库的版本号大于本地版本号，则 service 端会将 B/S 的教师端（本系统为多平台系统，除了本人负责的 iOS 客户端，还有其他同学负责的安卓客户端、web 端。其中手机端主要提供给学生使用，此处提到的是 web 系统提供的教师端功能，主要用于发布新的题库、查看学生相关学习情况。）选择的题库更新方式告知本系统是进行云图题库的全量更新还是增量更新。网络请求以及相关更新本地数据库代码如下：

（1）是否更新题库请求成功后的逻辑判断

```
if ([responseObject[@"versionNum"] integerValue] > [[[NSUserDefaults standardUserDefaults] objectForKey:VersionNumKey] integerValue]) {
                    //服务端版本号高于本地则更新题库
                    NSMutableArray *dicArray = [NSMutableArray array];
                    for (NSDictionary *dict in responseObject[@"questionList"]) {
                        YTQuestionItem *item = [YTQuestionItem questionWithDict:dict];
                        [dicArray addObject:item];
                    }
                    if (dicArray) {
                        self.questionArray = dicArray;
                        [UserInfo sharedInstance].isOriginalDataBase = NO;
                        [WeakSelf cacheQuestionListWithUpdateType:responseObject[@"updateType"]];
                    }
                }
                        NSMutableArray *dicArray = [NSMutableArray array];
                        for (NSDictionary *dict in responseObject[@"questionList"]) {
                            YTQuestionItem *item = [YTQuestionItem questionWithDict:dict];
                            [dicArray addObject:item];
```

```
                }
                if (dicArray) {
                    self.questionArray = dicArray;
                    [UserInfo sharedInstance].isOriginalDataBase = NO;
                    [WeakSelf
cacheQuestionListWithUpdateType:responseObject[@"updateType"]];
                    [[NSUserDefaults standardUserDefaults]
setValue:responseObject[@"versionNum"] forKey:VersionNumKey];
                }
```

本方法中的关键逻辑：将本地缓存的数据库版本号 versionNum 作为请求参数传给 service 端，service 端将所得版本号跟服务端数据库中存储的最新版本号相比较，若本系统所传版本号小于服务端版本号，则将题库以 json 的格式返回给本系统，返回 json 实例如下：

```
{
    "questionList": [
        {
            "QNum": "3",
            "QTitle": "高积云在 1000 米高空",
            "QOption1": "是",
            "QOption2": "不是",
            "QOption3": "",
            "QOption4": "",
            "QAnswer": "不是",
            "QExplain": "在 2000 米高空",
            "QRightNum": "a",
            "QLargeImgUrl": "",
            "QShortImgUrl": "",
            "QSection": "2",
            "QType": "2",
            "IsNew": "1",
            "QVersion": "3"
        }
    ],
    "versionNum": "3",
    "updateType": "2",
    "isSuccess": "1"
}
```

其中，questionList 为所需更新题目的数组，updateType 字段决定了更新的方式，1 代表全量更新，2 代表增量更新。更新完题库后也要将本地缓存的 versionNum 字段换成最新的。本系统操作数据库都要用到一个数据模型 YTQuestionItem，这个 model 即 MVC 设计模式里的 M[7]。

若网络中断或者其他原因导致请求失败，则不会进行数据库的更新操作，并会 toast 提示用户请求失败。

（2）数据库增量更新与全量更新的实现

全量更新实现方法相对简单，只需要将当前用户对应的数据库题库表删除即可实现[8]。增量更新方法如下：

```
- (void)saveQuestionListDataBaseWithArray:(NSArray *)array
{
    //删除全部数据
    [_dbQueue inDatabase:^(FMDatabase *db) {
        [db executeUpdate:@"DELETE FROM Question"];
    }];
    //插入数据
    [_dbQueue inDatabase:^(FMDatabase *db) {
        for (YTQuestionItem *item in array) {
            [db executeUpdate:@"insert into Question
(QNum,QTitle,QOption1,QOption2,QOption3, QOption4,QAnswer,
QExplain,QRightNum,QLargeImgUrl,QShortImgUrl,QSection,QType,QVersion)
values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)",item.QNum,item.QTitle,item.QOption1,item.QOption2,item.QOpti
on3,item.QOption4,item.QAnswer,item.QExplain,item.QRightNum,item.QLargeImgUrl,item.QShor
tImgUrl,item.QSection,item.QType,item.QVersion];
        }
    }];
}
```

增量更新需要在插入每一条数据的时候根据主键判断表中是否已有此条数据，若已有则删除此条数据再进行插入。

```
- (void)saveQuestionListDataBaseIncreUpdateWithArray:(NSArray *)array
{
    //queue 为新建的串行线程
    dispatch_async(queue, ^{
        for (YTQuestionItem *item in array) {
            //监测
            [_dbQueue inDatabase:^(FMDatabase *db) {
                NSString *existsSql = [NSString stringWithFormat:@"select count(QNum)
as countNum from Question where QNum = '%@'", item.QNum];
                FMResultSet *rs = [db executeQuery:existsSql];
                while ([rs next]) {
                    NSInteger count = [rs intForColumn:@"countNum"];
                    if (count == 1) {
                        NSLog(@"QNum is exist!");
                        [rs close];
                        return;
                    }
                    NSLog(@"QNum is not existed.");
                    dispatch_async(queue, ^{
                        //插入数据
```

```
                [_dbQueue inDatabase:^(FMDatabase *db) {
                    for (YTQuestionItem *item in array) {
                        [db executeUpdate:@"insert into Question
(QNum,QTitle,QOption1,QOption2,QOption3, QOption4,QAnswer,
QExplain,QRightNum,QLargeImgUrl,QShortImgUrl,QSection,QType,QVersion)
values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)",item.QNum,item.QTitle,item.QOption1,item.QOption2,item.QOpti
on3,item.QOption4,item.QAnswer,item.QExplain,item.QRightNum,item.QLargeImgUrl,item.QShor
tImgUrl,item.QSection,item.QType,item.QVersion];
                    }
                }];
            });
        }
        [rs close];
    }];
  }
});
}
```

此处需要注意的是，对数据库的多个操作要放在不同的线程里去做，否则数据可能会发生混乱，FMDB 也会适时的在控制台进行警告。

### 4.4 形态学分类法界面的实现

云学界面进入即会循环播放云类介绍的 m4a 格式的音频文件，用户若不方便听可手动暂停。考虑到形态学分类的类别比国际学分类少的多，因此形态学分类采用了加载服务端包含图文介绍的 html 文件的实现方案。这样的方式一方面保证了形态学分类知识的实时性，另一方面 html 对文字和图片的混合使用的处理能力是远远强于 native 实现的，这也是采用 html 进行形态学分类展示的重要原因之一[9]。

由于 html 文件的代码较长，在此就对本系统所用到的几个标签作一个解释。

iOS 平台想要加载 html 文件，必须要使用 UIWebView。对 UIWebView 进行相关配置后，方可以加载显示相关 html 内容。

html 相关标签

<body> </body> ：代表全文主体内容，通常会将整块内容包含在这个标签内。

<pre> </pre> ：代表文字按照既定的格式展示，不会改变原来的格式而变成一整块。

<img /> ：用于插入图片，要注意图片和文件必须完全在同一目录下，否则无法显示。


图 4.2 云学及形态学界面

## 4.5 练习考试界面主流程的实现

练习考试是本系统最核心的功能，也是本系统耗费时间最长的部分。以下为练考功能要点简介：

（1）在用户每答完一道题后，立即展示用户的选择是否正确。若正确，则自动跳转到下一题，错误则展示正确答案并显示试题详解。

（2）用户可点击右上角的收藏按钮对一些难题进行收藏，以便下次对这些不熟悉的题目进行专项练习。

（3）用户可点击右上角的题解按钮查看本题的详细解释。

（4）用户点击题目的图片可以展示大图，并且可以局部放大、移动。

（5）用户点击底部工具栏中的题号按钮可以将工具栏上移，并展示所有的题目序号以及用户答题正确情况。用户点击任意题目序号可切换至相应的题目。

（6）用户点击底部工具栏的交卷按钮，可提交本次答题情况至服务端，方便老师查看总结学生的学习情况。

由于要实现左右滑动切换题目的效果，起初的实现方案是 scrollView 内部嵌套多个 tableView，认真思考发现 scrollView 没有类似 tableView 一样的代理方法，题库的题目条数是未知且多变的，因此这个方案没有被采用，经过对 apple 技术文档的学习推敲，决定采用更加

高级的 UICollectionView[10]。整个界面上布局了一个与屏幕等宽的 UICollectionView，设置其滑动方向为仅限水平方向，设置每个 collection_cell 的大小为 UICollectionView 的大小，这样就能保证根据题目的数量确定 collection_cell 的个数（即内嵌 tableView 的个数）。嵌套相关实现如下：

```
if (collectionView.tag == MainCollectionViewTag) {//主集合视图
        static NSString *ID = @"collect_cell";
        UICollectionViewCell *cell = [collectionView dequeueReusableCellWithReuse
Identifier:ID forIndexPath:indexPath];
        UITableView *tableView = (UITableView *)[cell viewWithTag:1];
        _collectionViewRowNum = indexPath.row;
        [tableView reloadData];
        return cell;
    }
```

底部工具栏的实现相对来说比较基础，在此不详细列出初始化的代码，简单解释一下上移动画的实现[11]。apple 官方提供了 UIView 的动画实现方案，开发者只需指定动画执行的时间以及所要做的动画效果。通常这个动画效果都是通过平移、缩放、旋转等基础动画的单一或者叠加来实现的。工具栏的上移只用到了平移，并且在执行上移动画的过程中，背景的亮度是逐渐变暗而不是一个突变，这样显得动画更加顺畅、优雅。

```
- (void)didPressPageItem
{
    self.maskView.hidden = NO;
    bShowBottomView = YES;
    [UIView animateWithDuration:0.3 animations:^{
        self.maskView.alpha = 0.5;
        self.bottomView.frame = CGRectMake(0, ScreenHeight / 3, ScreenWidth,
ScreenHeight /3*2);
    } completion:^(BOOL finished) {
    }];
}
```

交卷功能需要将本次答题情况连同用户个人信息一起上传至 service 端。请求一共需要 5 个参数，学号 stuNum、姓名 stuName、专业 stuMajor 在登录成功后都会缓存在本地的 userDefault 中，日期在提交的时候实时生成即可，最关键的就是分数这个参数，需要设置一个全局变量用来记录实时的分数。提交成绩的请求参数设置如下：

```
NSMutableDictionary *paras = [NSMutableDictionary dictionary];
paras[@"stuNum"] = [UserInfo sharedInstance].stuNum;//学号必传
paras[@"stuName"] = [UserInfo sharedInstance].stuName;//姓名必传
paras[@"stuMajor"] = [UserInfo sharedInstance].stuNum;//专业
paras[@"date"] = [self fitScoreDate:[NSDate date]]; //日期
paras[@"score"] = self.testScore;//分数必传
```

testScore 在每答完一道题后，判断是否答对，若答对的话累加相应的得分。最终在提交试卷的时候，将 testScore 赋值给 score 参数进行成绩的提交。


图 4.3 云学及章节练习选项界面

## 4.6 错题的录入与界面展示

错题的录入功能是指在用户进行答题的过程中，若某一题答错了，则立即将对应题号的题目存入错题表 wrongQuestion 中。在云图界面的错题练习功能中，之前所有答错的题目都会被展示在此，并可以继续作答。录入错题的相关代码如下：

```
- (void)saveWrongQuestionListDataBaseWithItem:(YTQuestionItem *)item
{
    //queue 为新建的串行线程
    dispatch_async(queue, ^{
        [self checkTableWrongQuestion];
    });
    dispatch_async(queue, ^{
        //监测错题表中错题 QNum 是否存在
        [_dbQueue inDatabase:^(FMDatabase *db) {
            NSString *existsSql = [NSString stringWithFormat:@"select count(QNum) as countNum from wrongQuestion where QNum = '%@'", item.QNum];
            FMResultSet *rs = [db executeQuery:existsSql];
            while ([rs next]) {
                NSInteger count = [rs intForColumn:@"countNum"];
```

```
                    if (count == 1) {
                        [rs close];
                        return;
                    }
                    NSLog(@"QNum is not existed.");
                    dispatch_async(queue, ^{
                        //插入数据
                        [_dbQueue inDatabase:^(FMDatabase *db) {
                            [db executeUpdate:@"insert into wrongQuestion
(QNum,QTitle,QOption1,QOption2,QOption3, QOption4,QAnswer,
QExplain,QRightNum,QLargeImgUrl,QShortImgUrl,QSection,QType,QVersion)
values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)",item.QNum,item.QTitle,item.QOption1,item.QOption2,item.QOpti
on3,item.QOption4,item.QAnswer,item.QExplain,item.QRightNum,item.QLargeImgUrl,item.QShor
tImgUrl,item.QSection,item.QType,item.QVersion];
                        }];
                    });
                }
                [rs close];
            }];
        });
    }
```

如果对错题已经进行过进一步深入学习，并且熟记于心不需要再放在错题集里了，则可以手动点击错题练习右上角的删除错题按钮，即可从错题集中删除本题。

用枚举类型判断是否是错题练习模式，若是则在右上角增加一个删除按钮。相关判断代码如下：

```
        if (self.answerType == YTAnswerWrong) {
    //错题练习模式多个删除本题按钮
            self.navigationItem.rightBarButtonItems = @[deleteItem,storeItem,explainItem];
        } else {
            self.navigationItem.rightBarButtonItems = @[storeItem,explainItem];
        }
```

收藏功能与错题录入展示功能类似，不过收藏功能不会自动触发，需要用户手动点击收藏。同样的，在云图界面的收藏板块里也会实时的展示用户手动收藏的相关题目。

### 4.7 登录注册功能的实现

云图登录功能的实现，简单点的做法是直接本地保存一份用户表，登录的时候去验证这个表即可。这样简单粗暴的做法当然是不可取的，安全性极低且不符合设计原则，app 本地禁止保存任何敏感的用户信息、帐号信息[12]。

本系统当然也是采用 http 请求的方式让服务端去验证帐号、密码是否正确。相关登录验证代码如下：

（1）设置请求参数

```
NSMutableDictionary *paras = [NSMutableDictionary dictionary];
paras[@"stuNum"] = _txtMobile.text;
paras[@"stuPwd"] = _txtPwd.text;
```

（2）登录成功处理

```
[hud hide:YES];
[WeakSelf.navigationController popViewControllerAnimated:YES];
if (WeakSelf.delegate && [WeakSelf.delegate
respondsToSelector:@selector(didLoginDone)]) {
    [UserInfo sharedInstance].stuNum = _txtMobile.text;
    [UserInfo sharedInstance].stuName = _txtPwd.text;
    [WeakSelf.delegate didLoginDone];
}
```

在用户点击了登录操作之后，本系统会用一个遮罩提示用户正在登录，并用此遮罩盖住整个界面直到帐号验证成功或者验证失败。如果用户登录成功，则执行 LoginViewController 的代理方法 didLoginDone，用来执行登录成功后的跳转以及其他的逻辑处理。如果用户登录失败，则弹框提示用户登录失败。

注册功能需要用户如实填写自己的学号、姓名、专业、密码等，同样也是将这些参数通过网络请求传给 webService 的相应接口进行验证，若服务端数据库中不存在此学号的帐号信息，则可以进行注册。相关功能实现跟云图登录功能类似，此处不再一一贴出。


图 4.4 云图系统登录注册界面

## 4.8 练习模式的设计

云图练习模式一共有顺序练习、章节练习、错题练习、随机练习、收藏练习、模拟考试。顺序练习直接取数据库的所有题目即可，章节练习按照章节号从数据库中取对应章节的题目就能实现，错题练习跟收藏练习都是单独的错题表和收藏表。这里重点讲一下随机练习和模拟考试的设计。

（1）随机练习

```
NSMutableArray *questionList1 = [[[YTDataBaseManager sharedInstance] questionsList]
mutableCopy];
        if ([YTDataBaseManager sharedInstance].questionsList.count < 10) {
            self.questionList = [YTDataBaseManager sharedInstance].questionsList;
        } else {
            while (self.questionList.count < 10) {
                int r = arc4random() % questionList1.count;
                [self.questionList addObject:array[r]];
                [array removeObjectAtIndex:r];
            }
        }
```

随机练习是在题库总题数大于所要取出的题数的前提下，用系统自带的随机函数arc4random()依次从题库数组中进行取值，每取一次即从题库数组的副本中删除该条题目，然后再进行下一次随机取题，直到取满所需题数，这样既保证了随机性，也保证了云图随机练习不会出现重复的题目[13]。

（2）模拟考试

```
NSMutableArray * questionList1 = [[[YTDataBaseManager sharedInstance]
questionsListWithSectionNum:1] mutableCopy];
        NSMutableArray * questionList2 = [[[YTDataBaseManager sharedInstance]
questionsListWithSectionNum:2] mutableCopy];
        NSMutableArray * questionList3 = [[[YTDataBaseManager sharedInstance]
questionsListWithSectionNum:3] mutableCopy];
        while (self.questionList.count < 15) {
            if (array1.count > 0) {
                int r1 = arc4random() % questionList1.count;
                [self.questionList addObject: questionList1[r1]];
                [questionList1removeObjectAtIndex:r1];
            }
            if (self.questionList.count >=15) {
                return;
            }
            if (array2.count > 0) {
                int r2 = arc4random() % questionList2.count;
                [self.questionList addObject: questionList2 [r2]];
                [questionList2removeObjectAtIndex:r2];
```

```
        }
        if (self.questionList.count >=15) {
            return;
        }
        if (array3.count > 0) {
            int r3 = arc4random() % questionList3.count;
            [self.questionList addObject: questionList3 [r3]];
            [questionList3removeObjectAtIndex:r3];
        }
    }
```

模拟考试也是在题库总题数大于所要取出的题数的前提下，按照随机练习模式的设计依次从每个章节中随机取一题，没取完一题都会验证是否取满，若取满则退出取题。这样既保证了模拟考试的随机性，也保证了各个章节的题目分布均匀，不至于考察的知识点过于集中、片面。


图 4.5 模拟考试界面

# 5 系统测试

在软件开发过程中开发者由于开发周期、开发效率等因素的影响，经常会犯一些明显或者不太明显的错误，这些错误一旦跟随着 app 发不到了用户的手里，影响将会是相当可怕的，

所以有效全面的测试还是非常必要的。为了保证基于 iOS 的"云类识别"移动学堂的安全、稳定、良好的用户体验，本文对云图系统进行了必要的系统测试[14]。

系统测试的主要内容包括：

（1）功能测试：根据相关策划稿、原型测试系统的功能是否已经正确实现。正确性是衡量系统质量的最重要的参数，所以在进行系统测试时，功能测试是一个不可或缺的重要环节。

（2）健壮性测试：通过一些非常规数据对系统进行暴力测试，测试系统在不正常的流程中是否能够保证不崩溃、不闪退、不出现与非常规数据无关的异常。即要考察系统的容错能力与自我修复能力。

## 5.1 功能测试

功能测试也叫黑盒测试，它是指在测试过程中只考虑系统需要实现的功能而不考虑系统的逻辑结构以及代码如何实现的测试方式，一般从系统的界面、操作流程出发，按照软件需求编写测试用例，检测输入数据的预期结构与实际结构是否存在差异，进而提出修改，达到用户需求。

本系统根据需求分析设计的测试用例为：

（1）用户点击桌面云图图标打开“云类识别”虚拟仿真实验系统，点击底部云学 tab，点击进入形态学分类法界面，返回云学界面再点击进入国际学分类法界面。多次反复来回操作。

（2）从云学 tab 切换至云图 tab，点击右上角同步题库按钮并立即关闭网络，再开启网络并点击同步题库按钮。

（3）在云图界面不断点击各种练习模式，快速切换题目，多次点击不同的选项，连续点击底部的工具栏。

（4）不断快速切换底部云学、云图、我 tab。

经过云图测试用例的多次测试操作，云图系统都能正常、稳定地应对，即使面对非正常的极限情况，也有一定的抵御能力，云图系统测试成功。

## 5.2 健壮性测试

健壮性测试的内容包括：

（1）用非常规数据进行暴力测试，观察云图系统行为；

（2）关闭数据连接，观察云图系统行为；

（3）无逻辑极限操作，观察云图系统是否异常；

对于基于 iOS 的"云类识别"移动学堂而言，比较容易出问题的地方就是云图数据库的更新与替换，因此，健壮性测试主要就针对网络请求是否更新云图数据库过程的一个暴力测试。在点击同步题库按钮后，立即关闭网络，极限操作多次发现本系统均能弹框提示请求失败。

经多次验证，本系统在代码层面的健壮性处理比较到位，对数组越界等常见易 crash 问题也一一作了防御。

# 6 总结与展望

基于 iOS 的"云类识别"移动学堂的顺利完成，为高校气象专业学生以及广大云类爱好者提供了一个便捷的学习与研究的移动平台。用户只需一部 iOS 平台的设备即可随时随地进行“云类识别”的学习与考试。本文从技术实现和云图功能模块等多个方面具体阐述了云图系统从需求分析到具体实现的过程。由于个人开发经验有限和时间不足等多方面的原因，本系统还存在着一些值得深度优化的地方，比如云图模拟考试模块的随机抽取算法，尽管已经做到了根据各个章节的题目数按比例随机抽取，但是后续的版本可以考虑根据每个章节的重要程度进行更精细化的考题抽取。在后续的云图系统版本迭代中这些优化点将一一得到解决与完善。

在本次云图系统开发过程中，我也学到了很多开发技巧和技能，从刚开始对 iOS 开发零基础到能够独立完成一些基本的需求，面对各种各样的疑难问题，我从一开始的沮丧无助到后来的沉着应对，这其中除了自己不断地给自己加油打气、也多亏了开发经验丰富的老师同学的帮助与指导。整个云图系统的开发过程中我收获颇丰、受益匪浅。

由于云图系统直接对接了高校师生的学习需求，并且利用了 iOS 移动平台的巨大优势，加之云类识别学科不断热化，本系统的发展前景不容小觑，随着这些因素的持续影响，本系统定会得到更加广阔的用户市场。

# 参考文献：


[1] David Mark．精通 iOS 开发[M]．周庆成，译．北京：人民邮电出版社，2015：9－1．

[2]唐巧．iOS 开发进阶[M]. 北京：电子工业出版社，2014：12－1．

[3] Carlo Chung．Objective-C 编程之道：iOS 设计模式解析[M]．张龙，译．北京：人民邮电出版社，2011．

[4] Erica Sadun．iOS Auto Layout 开发秘籍[M]．孟立标，译．北京：清华大学出版社，2015：1－1．

[5]沙梓社，吴航．iOS 应用逆向工程[M]．北京：机械工业出版社，2015．

[6] Paul Pocatilu．Developing an M-Learning Application for iOS [J]．Informatica Economică，2013，63-68．

[7]Waqar Malik，Mark Dalrymple．Objective-C 基础教程[M]．周庆成，译．2 版．北京：人民邮电出版社，2013．

[8] 古贺直树．好设计不简单[M]．张君艳，译．2 版．北京：人民邮电出版社，2014．

[9]龚全福．基于 iOS 的新浪微博 iPhone 客户端的设计与实现[D]．成都：成都电子科技大学，2011．

[10]张丽娜．基于 iOS 的智能交通信息发布系统的设计与实现[D]．山东：山东大学，2012．

[11]刘鹏．基于 iOS 的个人健康管理系统客户端的开发[D]．辽宁：大连理工大学，2012：62－67．

[12] Chung C，Bucanek J．Pro Objective-C Design Patterns for iOS[M]．New York: Apress，2011．

[13]Simbeye D S，Zhao J，Yang S．Design and deployment of wireless sensor networks for aquaculture monitoring and control based on virtual instruments[J]．Computers and Electronics in Agriculture，2014，102：31-42．

[14] Weizhi Meng，Wang Hao Lee，S.R. Murali，Krishnan．JuiceCaster: Towards Automatic Juice Filming Attacks on Smartphones[J]．Journal of Network and Computer Applications, 2016，04－015．

# 致谢

在此毕业设计完成之际，我要对所有关心和照顾过我的人表示由衷的感谢！

首先，需要感谢我的毕业设计指导老师——王保卫，在他的悉心指导与热切关怀下我才能最终顺利完成本论文。在完成毕业设计的过程中，王老师合理安排我的毕业设计进度，从需求的分析到功能的实现，王老师持续地给予着我最耐心的指导与意见。在我遇到技术难题和开发障碍时，王老师引导着我一步一步地分析难题的关键所在，用身体力行的事实教育了我授之以鱼不如授之以渔的深刻道理。感谢王老师的敬业精神与耐心教导，我会用最优秀的成绩来回报老师的关爱。

我还要感谢我的同学们，感谢同学们在大学四年里给予我的莫大的关怀与真挚的友谊，在我感到迷茫甚至怀疑人生的时候，是同学们的真诚与善良，让我重新燃起了对工作、生活的激情。感谢大家一路相伴，真诚的希望大家未来前程似锦。

最后，感谢计算机与软件学院的领导和老师，是他们在这四年里为了我默默付出，我才能打好学科基础，才能顺利完成我的本科毕业设计，再次感谢。