武漢理工大學

毕业设计（论文）

心理测评系统的设计与实现

学院（系）：计算机科学与技术学院

专业班级: 软件工程 zy1501 班

学生姓名: 吕世豪

指导教师: 刘传文

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名：

年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于　1、保密 □，在　年解密后适用本授权书

2、不保密 □

（请在以上相应方框内打“√”）

作者签名：　　年　月　日

导师签名：　　年　月　日

摘要

心理评测是一种比较先进的测试方法，它是指通过一系列手段，将人的某些心理特征数量化，来衡量个体心理因素水平和个体心理差异差异的一种科学测量方法。按评测的内容、对象特点、表现形式、目的、时间、要求等分为若干种类。主要是各机关、企业、组织等用来选拔人才、安置岗位以及对一个人进行诊断、评价、辅助咨询的一种手段，心理测评包含能力测试、人格测试和兴趣测试等。

本论文实现了一个 B/S 架构的心理评测系统，基于 Python 的 flask 框架，React 框架和 MySQL 数据库，实现了用户的注册登录、账号激活、找回密码、管理员管理试卷、管理员管理用户、计算分数、分析答案得出结论的功能。

使用系统可以进行心理评测并得出结论，每份试卷针对性的让使用者了解自己对应指标的分数，更加了解自己对应方面的优点和缺点，同时也使管理员能够根据成绩进行筛选用户，了解大众的心理趋势。

关键词: 心理评测，结果分析，档案管理，目标筛选

Abstract

Psychological assessment is a relatively advanced testing method. It refers to a scientific measurement method to measure the level of individual psychological factors and individual psychological differences by quantifying some psychological characteristics of human beings through a series of means. According to the content, object characteristics, manifestation, purpose, time and requirements of the evaluation, it can be divided into several categories. It is mainly a means used by various organs, enterprises and organizations to select talents, place posts, diagnose, evaluate and assist a person in counseling. Psychological assessment includes ability test, personality test and interest test. Generally speaking, psychological assessment has great significance in education, society and work. This system completes the basic function of psychometric assessment.

This paper implements a psychological evaluation system based on B/S architecture. Based on Python flask framework, React framework and MySQL database, the functions of user registration, account activation, password retrieval, administrator management test paper, administrator management user, calculating scores, analyzing answers and drawing conclusions are realized.

The system can be used to carry out psychological evaluation and draw conclusions. Each test paper aims to let users know their corresponding index scores, better understand the advantages and disadvantages of their corresponding aspects. At the same time, administrators can screen users according to their scores and understand the psychological trends of the public.

Key Words: Psychological assessment, Result analysis, File management, Target selection

目录

第 1 章　需求分析

心理评测是一种比较先进的测试方法，它是指通过一系列手段，将人的某些心理特征数量化，来衡量个体心理因素水平和个体心理差异差异的一种科学测量方法。按评测的内容、对象特点、表现形式、目的、时间、要求等分为若干种类。主要是各机关、企业、组织等用来选拔人才、安置岗位以及对一个人进行诊断、评价、辅助咨询的一种手段，心理测评包含能力测试、人格测试和兴趣测试等。

1.1　心理测评的意义

1.1.1　心理评测在社会中的意义

1. 描述：心理评测可以从个体的智力、能力倾向、创造力、人格、心理健康等各方面对个体进行全面的描述，说明个体的心理特性和行为。

2. 诊断：心理评测可以对同一个人的不同心理特征间的额差异进行比较，从而确定其相对优势和不足，发现行为变化的原因，为决策提供信息。

3. 预测：心理测评可以确定个体间的差异，并由此来预测不同的个体在将来的活动中可能出现的差别，或推测个体在某个领域未来成功的可能性。

4. 评价：可以评价个体在学习或者能力上的差异，人格的特点以及相对长处和弱点，评价儿童达到的发展阶段等。

5. 选拔：心理评测可以客观、全面、科学、定量化地选拔人才。

6. 安置：了解个体的能力、人格、心理健康等心理特征，从而为因材施教或人尽其才提供依据。

7. 咨询：心理测评可以为学校的升学就业咨询提供参考，帮助学生了解自己的能力倾向和人格特征，确定最有可能成功的专业或者职业。

1.1.2　心理测评在理论研究中的意义

1. 搜集资料：心理测评是搜集有关心理学资料的一个简单易行而又可靠的方法。

2. 建立和检验假设：从心里测评的资料中可以发现问题，建立理论假设，并且可通过测量结果进一步来检验这些假设，进而推动心理科学的发展，例如智力结构理论的提出和发展，智力测验就起了重要作用。

1.1.3 心理测评在教育领域中的意义

1. 心理测评是科学教学和学习的良好工具：心理评测可以帮助教师和学生更准确、客观、迅速、全面的了解学生或自己的心理特点，教师便可因材施教，学生也可以选择适合自己心理特点的学习方式。心理测评可以评价教师的教学成功，又可评价学生的学习效果，从而为教育提供反馈信息，一方面可使教师总结自己的经验和教训，改进教学，提高教学质量，另一方面可使学生省思和优化学习方式，提高学习效率。

2. 心理测评是教育研究和改革的重要手段：迅速大量的搜集学生的信息，从中可以归纳出学生心理发展的普遍规律和特点，从而神话心理学理论研究，进而据此制定出教育目标、内容和方法。此外，任何一种新的教育理论、新的教材、新的教学方法和教育实验的价值，在没有通过测量检验其教育效果之前，都无法科学的评定其价值，而心理测评的结果可以用来评价教育效果，提供客观的精确的依据，进而衡量他们的价值。

1.2 国内外心理评测现状分析

心理测评的成千上万的受测者往往来自全国各地，很难进行集中的测试，因此现状是例用计算机互联网的方式进行测评是有利条件。

另外，针对儿童的测评借助计算机的方式实现文字、语音、图画、录像等多种方式的结合。

对受测者需要进行多套经典量表才能全面的评估个体的心理，存在诸多的局限。量表多意味着题量大，答题时间长，受测者经常会出现不良情绪，影响作答效率，宽窄网的研究人员曾对三万人的施测进行研究，发现测评进行 20 分钟的时候，15% 的受测者会出现疲劳、烦躁等不良情绪，进行 30 分钟的时候，24% 受测者会出现不良情绪，造成测评的效率低。多套量表的设定会出现维度重复甚至是不需要评测的维度，造成评测资源的重复和浪费，而且还会影响后期数据的录入和清理，浪费大量的时间和人工成本。

国内目前的心理咨询师培养要求和国际上的要求距离甚远，这有待国家某些制度的完善。心理学研究的领域非常的广泛，APA（美国心理协会）就有 53 个学科分支，每个分支都有其广阔的发展前景。从偏于基础性的实验心理学、认知心理学、生理心理学，到偏于应用性的教育心理学、社会心理学、军事心理学、管理心理学等。

1.3 心理评测系统的目的

市面上已经有很多知名的评测系统，当然也不乏心理评测系统。一个基本的评测系统要能进行用户的注册登录，做题，得到自己的结果，同时管理人员可以进行试卷的管理和上传，用户的权限设置，用户的批量上传，以及对成绩的筛选。好的心理评测系统

可以做到后面遇到的题目和之前选择的答案相关，以及可以通过音频和图片来进行问题和答案的阐述。

所以我们需要做的应该包含用户模块，用户可以注册登陆，可以修改自己的信息，可以查看自己做的试卷的成绩，作为管理员可以上传试卷，管理试卷，查看试卷的综合成绩，根据试卷成绩进行筛选用户。

最终我们的系统要达到做题的时候清晰的理解题目和答案的意思，要做到这个我们可以用图片辅以说明，同时在使用者答题疲倦的时候可以做到暂停保存答案，下次登陆的时候可以继续做，每个人可以根据我们系统的分析结果了解当前自己的心理状态，明确自己的目标，改进自己的不足，系统管理者可以方便的操作试卷和根据成绩筛选出目标用户。

第 2 章　系统设计

此心理评测系统使用 B/S 架构，前端使用 React 框架搭建用户界面，后端使用 Python 的 Flask Web 框架搭建 Restful API。

2.1　项目架构

好的项目目录有清晰的条理，可以通过目录知道各个文件夹的作用，易于拓展。

2.1.1　前端项目架构

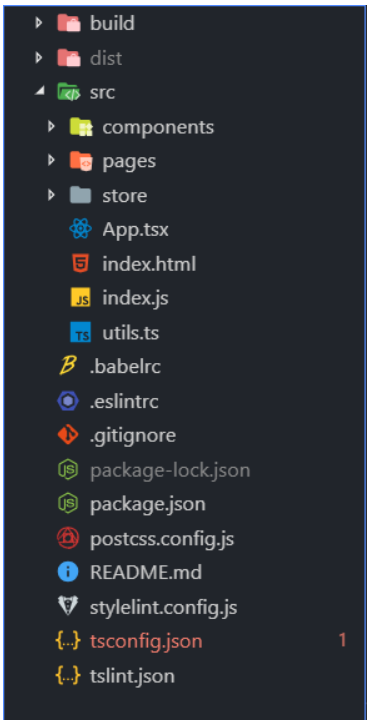

图 2.1: 前端项目目录

build 文件夹下存放着 webpack 的编译打包配置文件，分为两个环境，development 开发环境和 production 生产环境。

dist 文件夹是存放着 webpack 打包后的文件，用作于 webpack-dev-server 的开发环境文件和上线后的生产环境文件。

src 文件夹是项目的根目录，里面的 components 文件夹存放着公用的组件，pages 存放着系统的各个页面文件，store 存放着项目所需要用到的 redux 状态仓库，App.tsx 是项目的主页面，index.html 是 webpack 打包注入 js 依赖的 html 文件，index.js 是项目的主入口，utils.js 是项目所需要的共用工具。

.babelrc 是 babel-loader 的配置文件，配置编译 js 的目标版本。

.eslintrc 是 eslint 代码检查的配置文件。

.gitignore 是 git 仓库 push 的时候忽略的文件。

package.json 是前端项目的架构文件，记载该前端项目所需要的依赖和依赖版本和该前端项目的基本信息。

postcss.config.js 是 postcss 的配置文件。

README.md 是描述该项目的 markdown 文件。

stylelint.config.js 是 stylelint css 代码检查的配置文件。

tsconfig.json 是 Typescript 的编译配置文件。

tslint.json 是 tslint 的配置文件。

2.1.2 后端项目架构

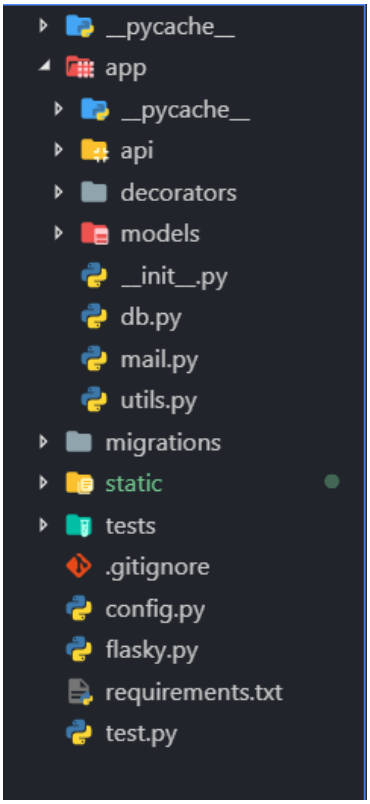

图 2.2: 后端项目目录

___pycache___ 是 IDE 自动生成的 sourcemap 索引文件夹。

___init___.py 文件是该 Python 包的默认包导出文件。

app 文件夹是项目的根目录，api 文件夹存放着 Resuful API 的文件，decorators 文件夹存放着项目中所用到的装饰器，models 文件夹存放着该项目所用到的所有对应数据库中的数据模型，db.py 是该项目连接数据库获得数据库实例的文件，mail.py 是该项目所使用到的邮箱模块，utils.py 是该项目的工具库。

migrations 是 Flask-Migrate 根据 Sqlalchemy ORM 数据模型定义进行数据库备份的文件夹。

static 是该后端项目的静态文件存放目录。

tests 是后端项目的测试文件存放目录。

.gitignore 是 git 仓库索要忽略文件的声明文件。

config.py 是该项目的配置文件。

flasky.py 是该项目的主入口文件，启动文件。

requirements.txt 记录着该项目的依赖和依赖版本。

2.2 项目重要依赖

2.2.1 React

React 是一个声明式，组件化的前端框架。其使用 Virtual DOM 技术把应用状态和 DOM 分离开来，搭配 Diff 算法来最小颗粒的更新 DOM，而这一切开发者都不需要关心，开发者只需要专注注意力在开发业务代码上。React 拥有极丰富的生态，本项目中还使用了 React-redux 来进行 redux 状态仓库的管理。

```
class App extends Component {
  public componentDidMount() {
        ...
  }

  public render() {
        return (
          <div>
            this is App
              <a onClick={this.onClick}>
          </div>
        )
  }

  private onClick = (e: ClickEvent) => {
    e.preventDefault();
    ...
  }
}
```

React 编写组件

所有 React 组件都要继承 React.Component，拥有 React 组件的生命周期，每个 React 组件需要实现一个继承父类的 render 方法，来控制组件如何进行渲染。

2.2.2 Flask

Flask 是一个使用 Python 编写的轻量级 Web 应用框架。其 WSGI 工具箱采用 Werkzeug ,模板引擎则使用 Jinja2 。Flask 使用 BSD 授权。Flask 也被称为“microframework”，因为它使用简单的核心，用 extension 增加其他功能。Flask 没有默认使用的数据库、窗体验证工具。本项目中使用了 Flask-Migrate 来进行数据库的备份，Flask-Mail 来进行邮件的发送，Flask-SqlAlchemy 来进行更加简单的操作 SqlAlchemy。

```
@app.route("/index")
def Index():
        return "<h1>Hello, Index</h1>"
```

Flask 定义视图路由

传给 app.route 的是一个路径字符串，该方法返回一个装饰器，装饰器装饰路由函数，每个路由函数返回一个页面或者数据。

2.2.3 Typescript 和 Javascript

项目使用的前端语言并非常规的 Javascript 而是它的超集——Typescript，其作为超集在绝对兼容 Javascript 代码的前提下，加入了许多静态语言所拥有的特性，比如接口、泛型、类型判断等等。Javascript 作为脚本语言其灵活性和方便性为大家所称道，但是也由于其灵活性，前端代码的质量低下导致项目难以维护和扩展，Typescript 就是解决 Javascript 严谨性而诞生的一门语言。Java 作为世界上用途最广泛，使用人数最多的语言就是因为其严谨性得到了很多企业的偏爱，Typescript 看起来 80% 都和 Java 很相似，可以理解为在 Javascript 的基础上加了一层代码检查和提示。

2.2.4 前端构建工具——Webpack

浏览器的版本众多，浏览器与浏览器之间的内核有许多差异，这导致同一份代码可能在不同浏览器下的表现不同，通过人工的方式编写兼容代码会有很高的人力成本和时间成本。同时在以前的开发过程中，前端依赖后端导致前端开发与后端严重耦合，降低了开发效率。前端工程化时代的到来，让前端拥有自己的项目构建能力，同时还解决了

不同浏览器之间的兼容问题，诞生了许多前端构件工具，Gul、Grunt、Webpack 等等。其中 Webpack 是当下活跃度最高，最火热的前端构建工具。

Webpack 最主要的两个功能是编译和打包。本项目所使用的 React 是使用 JSX 语法，不是浏览器默认可支持的语法，所以要将我们的前端代码编译成浏览器可识别的代码。Webpack 最重要的两个概念就是 loader 和 plugin，loader 负责各种类型文件（.css，.js，.ts，.png.....）的编译工作，可以将他们转化成目标浏览器的可识别文件，plugin 则负责除了编译以外的所有工作，包括打包，体积优化，分离依赖等等。

```
{
  test: /\.jsx?$/,
  exclude: /node_modules/,
  use: ["babel-loader"]
},
{
  test: /\.tsx?$/,
  exclude: /node_modules/,
  use: [{
    loader: "ts-loader",
    options: {
      transpileOnly: true,
      getCustomTransformers: () => ({
          before: [
            require("ts-import-plugin")({
              libraryName: "antd",
              libraryDirectory: "es",
              style: "css"})
          ]
      }),
      compilerOptions: { module: "es2015" }
    }
  }]
}
```

Webpack 编译 jsx 和 tsx 文件成 es5 的配置

Webpack 还提供了前端可实时预览的热更新服务器————webpack-dev-server，该插件可以实时的检测项目文件的变化并重新编译和打包让浏览器的页面刷新。

Webpack 打包体积优化也是一个深奥的学问。

Webpack 在原始配置的情况下，因为是单页面应用，所以打包出来以后的 bundle 文件非常之大，高达 4MB，这对于首屏加载是非常不利的，过长的白屏会流失很多的用户量，所以 Webpack 的优化也是一门学问。

常用的优化有以下几点：

1. 配置 optimization 中的 splitChunks 选项，会根据依赖图对于多次依赖的包单独打包然后进行引用。
2. 配置 externals 选项对于第三库在 html 文件中进行 CDN 引用，打包时候忽略该库。
3. 使用 uglifyjsPlugin 对打包文件进行压缩。
4. 使用懒加载进行加载组件和页面，react-loadable 可以通过一个高阶组件包裹我们的异步组件使用 import 函数进行异步的加载组件，当加载成功的时候显示组件否则就加载 Loading 组件。
5. 配置 Gzip。

进行上面的优化以后，打包以后的文件按照路由和组件分成了十几个文件，最大的体积不超过 400KB，减少了首屏加载的时间。

2.2.5 SqlAlchemy

SqlAlchemy 是 Python 编程语言下的一款开源数据 ORM 框架。提供了 SQL 工具包及对象关系映射（ORM）工具，使用 MIT 许可证发行。SQLAlchemy “采用简单的 Python 语言，为高效和高性能的数据库访问设计，实现了完整的企业级持久模型”。SQLAlchemy 的理念是，SQL 数据库的量级和性能重要于对象集合；而对象集合的抽象又重要于表和行。因此，SQLAlchemy 采用了类似于 Java 里 Hibernate 的数据映射模型，而不是其他 ORM 框架采用的 Active Record 模型。不过，Elixir 和 declarative 等可选插件可以让用户使用声明语法。

```
class User(Model):
  __tablename__ = "users"
  id = Column(Integer, primary_key=True)
  name = Column(String(32), nullable=False)
  password = Column(String(32), nullable=False)
```

使用 SqlAlchemy 创建一个用户模型

在上面我们建立了一个用户模型，对应数据库中的 users 表，该模型拥有 id、name、password 等字段，其中 id 是主键。

2.2.6 Pandas

Pandas 是一个数据科学分析第三方库，可以方便的建立矩阵以及对矩阵中的数据进行各种数学运算和变换，在解析试卷和成绩分析的时候有很多使用。同时可以对矩阵中的数据进行清理和类型的转换。

2.3 数据库设计

通过 SqlAlchemy 我们可以方便的在数据库中创建一个表并生成记录，同时也可以方便的进行数据库操作，定义数据之间的关系，搭配 Flask-Migrate 还可以方便的进行数据库备份、恢复和回滚。

这个系统的主要有三个表，用户表 (users)，试卷表 (papers)，成绩表 (grades)，数据库实体之间的关系，箭头都代表一对多关系，粗体字体为必填字段。

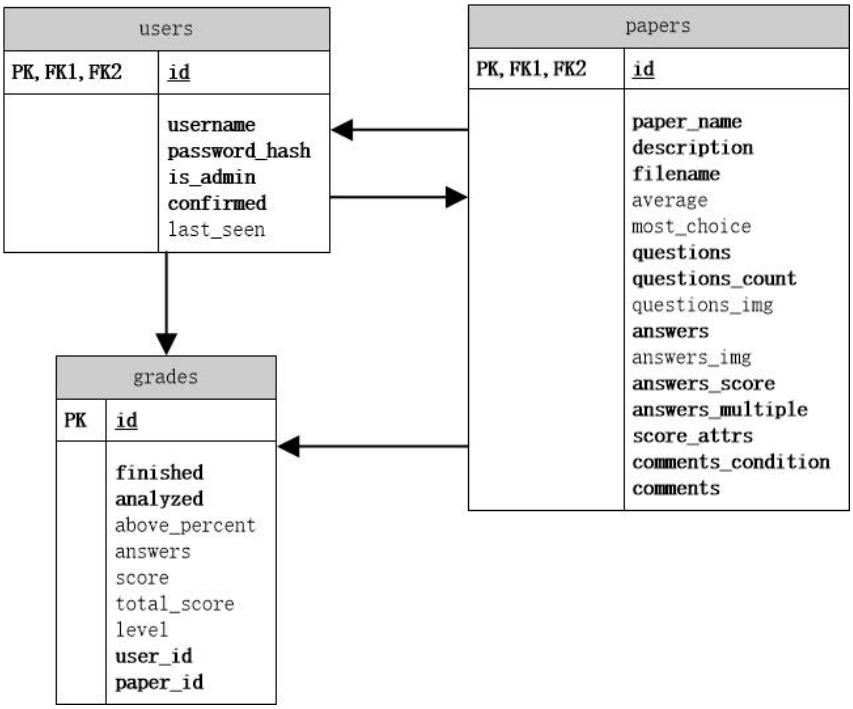

图 2.3: 数据库关系

2.3.1 一对多关系

如图中，用户表和成绩表，试卷表和成绩表之间都为一对多关系。SqlAlchemy 中可以方便的通过 relationship 和 foreignKey 进行设置一对多关系。

```
class User(Model):
...
grades = relationship("Grade", backref="user", lazy="dynamic",
cascade="all, delete-orphan", passive_deletes=True)

class Paper(Model):
...
grades = relationship("Grade", backref="paper", lazy="dynamic",
cascade="all, delete-orphan", passive_deletes=True)

class Grade(Model):
...
user_id = Column(Integer, ForeignKey("users.id",
ondelete="CASCADE"))
paper_id = Column(Integer, ForeignKey("papers.id",
ondelete="CASCADE"))
```

建立用户和成绩，试卷与成绩之间的一对多关系

relationship 的这一端代表着 1 的这一端，backref 传入的是在 Grade 对象下如何访问 User 或者 Paper 对象，lazy 代表如何访问这个对象，一对多之间的关系通过外键的形式进行设置，比如 users.id 代表着是 users 这张表中的 id 作为外键。同时外键的字段设置需要和做外键的字段类型一模一样，否则会报错。

1. select，默认选项，直接查找出所有对象，例如 user.grades 直接通过列表列出所有该用户下的 grade
2. dynamic 返回的是一个查询集，例如 users.grades.filter_by(...) 返回的是一个查询集可以进行筛选操作。
3. joine 类似于 select，但是 seletc 在查询的过程中需要查询两个表，而 joined 是建立一个新的连结表进行查询，在数额大的情况下使用 joined 能节省不少性能。

lazy 选项

SqlAlchemy 作为最强大的 ORM 工具当然不只是定义关系，还有就是定义实体不存在的时候如何处理与其相关的数据，加入一个用户被删除了，那么于其关联的成绩也应该要全部删除，而不是作为冗余数据存在数据库中，一套试卷被删除了同理。那么 cascade 就是干这个事情的。

1. save-update，默认选项，添加一条数据的时候会把其他和它相关的数据都添加到数据库中。
2. merge，默认选项，合并一个对象的时候会将使用了 relationship 相关联的对象也进行 merge 操作。
3. expunge，移除操作的时候，会将相关联的对象也进行移除，这个操作只是从 session 中删除，而不是从真正的数据库中删除。
4. all，包含上面所有的选项。
5. delete-orphan，删除相关的孤儿数据。

cascade 选项

对应的要在多的那一端设置 ondelete 属性。

至此一对多的关系就设置好了，使用 Flask-Migrate 升级数据库的时候可以自动检测相关外键的设置然后更新数据库。

2.3.2 多对多关系

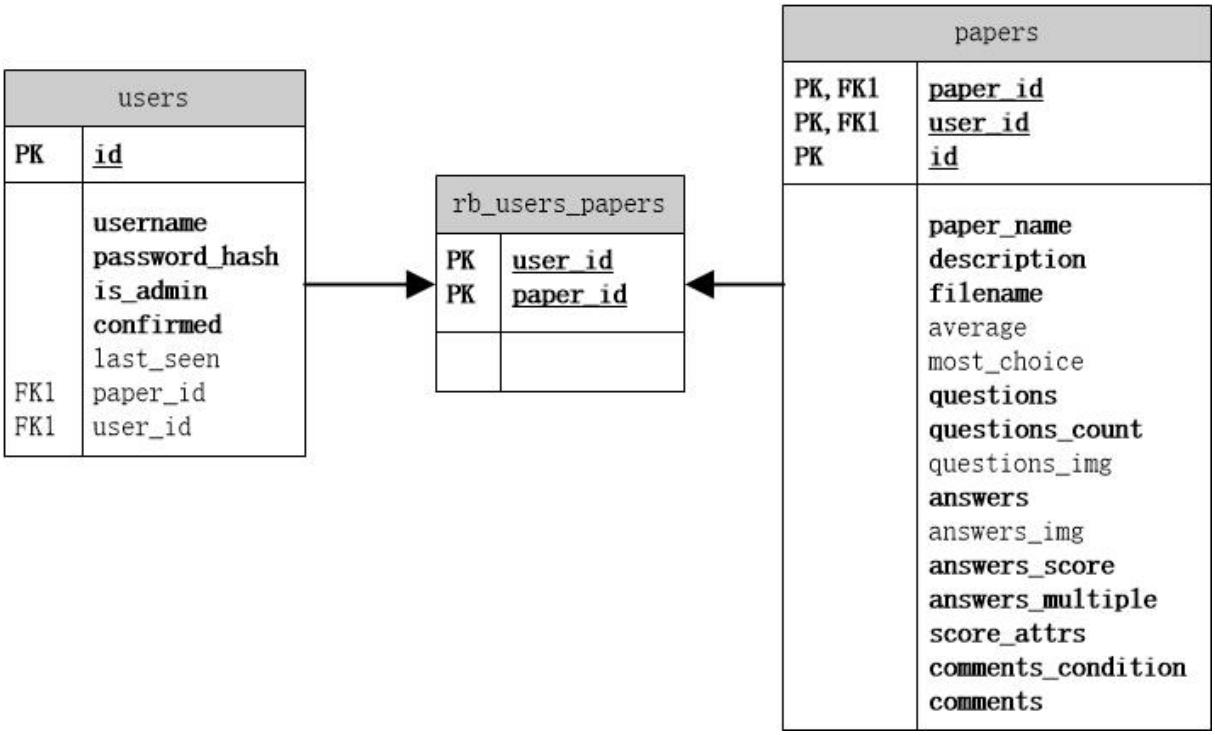

图 2.4: 多对多关系

一个用户可以做多张试卷，一张试卷也可以有很多用户来做。这就是一个多对多关系。多对多关系中，需要一个额外的连接表来做查询。

```
class User(Model):
...
papers = relationship("Paper", secondary=rb_users_papers,
lazy="dynamic")

class Paper(Model):
...
users = relationship("User", secondary=rb_users_papers,
lazy="dynamic")

// 连接表
rb_users_papers = Table(
"rb_users_papers",
Column("user_id", Integer, ForeignKey("users.id"),
primary_key=True),
Column("paper_id", Integer, ForeignKey("papers.id"),
primary_key=True)
)
```

多对多关系代码

上面的代码说明了 users 表和 papers 表之间是通过一个 rb_users_papers 进行连接查询的，同时连接表中的每个字段都需要与原字段的属性相同。在原表中的 relationship 中需要设置 secondary 属性来进行使用连接表的查询。使用连接表进行连接的两个表，任何一条记录被删除以后相关的连接表记录也会被删除，当然也可以设置与其关联的记录也被删除，但在此处的情况不适合这样做，试想一下，如果一个用户记录被删除了，那么该用户做过所有的试卷记录也会被删除，别人就无法做了，这样是非常的不合适的情况。

数据库的设计是一个完善的软件系统必备也是最重要的一个阶段，一个好的数据库设计能让一个系统的性能良好、易于维护、易于拓展。有些人可能会觉得直接手写 sql 更加灵活和方便，但是如果作为一个多人维护开发的系统来说，每个人都有自己写 sql 的习惯，会让项目代码水平层次不齐难于维护，这时候就要使用 SqlAlchemy 这种工具了，通过一种 OOP 的方式来进行数据库的管理。

第 3 章　系统实现

如果说一个项目的架构和数据库设计是一个项目的骨架，那么业务代码就是在其骨架上面生长的骨肉。一个好的骨架能够让血肉长的更加健康。

3.1　RESTful API

REST 全称是 Representational State Transfer，中文意思是表述（编者注：通常译为表征）性状态转移。它首次出现在 2000 年 Roy Fielding 的博士论文中，Roy Fielding 是 HTTP 规范的主要编写者之一。他在论文中提到：” 我这篇文章的写作目的，就是想在符合架构原理的前提下，理解和评估以网络为基础的应用软件的架构设计，得到一个功能强、性能好、适宜通信的架构。REST 指的是一组架构约束条件和原则。” 如果一个架构符合 REST 的约束条件和原则，我们就称它为 RESTful 架构。

REST 本身并没有创造新的技术、组件或服务，而隐藏在 RESTful 背后的理念就是使用 Web 的现有特征和能力，更好地使用现有 Web 标准中的一些准则和约束。虽然 REST 本身受 Web 技术的影响很深，但是理论上 REST 架构风格并不是绑定在 HTTP 上，只不过目前 HTTP 是唯一与 REST 相关的实例。所以我们这里描述的 REST 也是通过 HTTP 实现的 REST。

RESTful API 有以下的特征：

1. 每一个 uri 代表一种资源。
2. 客户端和服务器之间，传递这种资源的某种表现层。
3. 客户端通过四个 HTTP 动词 (GET、POST、DELETE、PUT)，对服务器端资源进行操作，实现” 表现层状态转化”(增删改查)。
4. URL 中通常不出现动词，只有名词。
5. 使用 JSON 不使用 XML。

3.1.1　在 Flask 中编写 RESTful API

Flask 作为一个轻量级的后端框架，不使用扩展的情况下，就可以使用路由功能提供 RESTful 接口。

```
@app.route("/index", method=["GET", "POST", "PUT", "DELETE"])
def Index():
if request.Method == "GET":
...
```

```
elif request.Method == "POST":
...
elif request.Method == "PUT":
...
elif request.Method == "DELETE":
...
```

Flask 编写 RESTful API 接口

3.1.2 前端使用 RESTful 接口

axios 是一个封装了 xhr http 服务的一个前端 http 请求工具。

```
axios({
method: "get" / "post" / "put" / "delete",
url: "",
data: ""
})
```

3.2 使用 Redux 存储网络请求状态

Redux 是 JavaScript 状态容器，提供可预测化的状态管理。可以让你构建一致化的应用，运行于不同的环境（客户端、服务器、原生应用），并且易于测试。其使用 Immutable 数据结合 React 能让页面相应数据进行最小颗粒度的更新 DOM 结点。

因为 Ajax 都是异步的一个请求过程，需要在回调函数内对数据进行处理，这种就会出现当依赖的 Ajax 请求过多的情况下，不断的编写和嵌套回调函数的时候代码就会变得非常难以阅读和维护，所以使用 Promise 封装一种通用的 Ajax API 同时用 Redux 进行对数据的保存是非常有必要的。

当进行一个 Ajax 请求的时候，我们默认出发一个 Redux Action 将 Ajax 的数据储存在 Redux 中然后使用 React-Redux 的 Connect 函数将我们的组件根据 Redux 中的状态变化进行更新。

```
const registHttpAction: (actionName: string) => void =
(actionName: string) => {
reducerActions[actionName] = createAction(
actionName.toUpperCase(),
(payload: IHttpResponse) => payload);
```

```
actionsMap[actionName] = async (params: IHttpRequest):
Promise<void> => {
try {
const response: AxiosResponse = await axios(params);
const { headers } = response;
if (headers.redirect && headers.redirect === "login") {
if (!m) {
m = message.error("验证失败，需要登陆，
三秒后自动跳转 Authenticate fail ,
require login , auto-redirect after 3s");
}
setTimeout(() => {
location.href = "/login?redirect=" + location.pathname;
if (m) {
m();
}
}, 3000);
} else if (headers.redirect && headers.redirect === "admin") {
message.error("没有权限 Forbidden", 3);
}
store.dispatch(reducerActions[actionName](response.data));
return Promise.resolve();
} catch (err) {
message.error("服务器错误 Server Internal Error");
return Promise.reject();
}
};
```

Redux Http Action 注册函数

注意这里我们一定要在 PromiseLike Function 里面返回一个 Promise 对象来表示该 Promise 的状态，否则在使用 Await 关键字的时候，不会进行阻塞。

这样做的好处就是，通过封装了 axios 进行 http 请求的时候可以有一个公共的地方存储返回的数据，同时还可以在发生错误的时候或者在权限不够的时候读取 headers 直接取消渲染并给出提示让用户进行相应的操作。

3.3 账号系统

一个系统想要有人使用，给每个人都提供服务，都需要一个完善的账号系统，每个人都有自己的账号，进行登陆，使用系统，更改设置等等。

3.3.1 密码加密

出于安全考虑，用户的密码不通过明文保存，除非你对自己的运维系统有着极高的自信，当然这样也不建议这样做。毕竟人外有人，天外有天，如果没有被盗取那么说明你的系统价值还不是那么的高。

werkzeug.security 是一个 Python 的密码包，其中的 generate_password_hash 的方法可以创建 MD5 加密的字符串，check_password_hash 可以用于验证密码是否正确。

```
@property
def password(self):
raise TypeError("cannot␣read␣a␣user's␣password")

@password.setter
def password(self, password):
self.password_hash = generate_password_hash(password)

def verify_password(self, password):
if not password:
return False
return check_password_hash(self.password_hash, password)
```

用户密码的创建和验证

3.3.2 token 的创建和验证

token 用于用户的登陆状态的保存，其通过 cookie 的形式保存在 request 头中，后端通过对 response 的 set-cookie 字段对网页的该域下进行 cookie 的设置。同时 token 可以设置过期时间。

```
def generate_confirm_token(self):
s = Serializer(current_app.config["SECRETE_KEY"],
expires_in=60 * 5, salt=current_app.config["CONFIRM_TOKEN"])
token = s.dumps({"user": self.username})
```

```
return token.decode()
```

得到加密 token 的代码

初始化对象的时候向 Serializer 内部传入解密字符串，token 的有效时长，以及加密的盐。此 token 的时长是五分钟，将用户的用户名做了加密通过 HTTP 传给前端，这里要注意，因为 token 生成的是二进制字符串，我们先要把它解密成普通字符串以后才能设置 cookie，前端下次请求的时候会带上这个数据，后端再进行解密即可获得用户的状态。

```
response.set_cookie("token", token, max_age=max_age,
httponly=True)
```

response 设置 header 中的 cookie

```
s = Serializer(current_app.config["SECRETE_KEY"],
salt=current_app.config["CONFIRM_TOKEN"])
try:
data = s.loads(token)
if username != data["user"]:
return -2, "Bad signature"
return 0, "Confirm success"
except SignatureExpired:
return -1, "Signature expired"
except BadSignature:
return -2, "Bad signature"
```

验证 token 的代码

同样的，在解密的过程中需要获得 Serializer 的实例，当 loads 一个 token 的时候，如果验证失败或者 token 超时，代码会抛出异常，SignatureExpired 和 BadSignature，我们需要对对应的状态进行处理给前端响应。

3.3.3 注册与邮箱验证

现在许多软件注册都需要用手机号，并且一个手机号只能注册一次，但在此发送短信需要使用第三方收费服务，所以我使用替代的邮箱来进行验证。既然需要邮箱，我们就需要写一个页面来进行验证结果的展示，同时也用到了 token。以一个注册用户名为 cindy 的用户来看，我们把 username: "cindy" 通过 Serializer 进行加密生成一个 token，同时我们的注册页面的 pathname 是/confirm，那么我们给该用户注册成功发送邮箱中的链接就是 http(s)://origin/confirm?token={token}，当访问这个页面的时候我们把 token 传给后端，后端进行解密然后验证 token，把数据库中相应用户的邮箱激活。当然如果在验证时候 token 过期了还可以重新发送验证邮箱。

3.3.4 登陆与找回密码

登陆的时候会首先查看是否邮箱已经激活，如果没有激活则发送新的验证邮箱去激活账号才能登陆。同样的如果此账号忘记了密码，则可以通过邮箱进行密码找回和重新设置密码。

3.4 上传文件与解析试卷

心理评测系统最重要的是试卷，作为管理员需要进行试卷的编写和上传，后端要对上传的文件进行分析然后记录到数据库中。

3.4.1 上传文件

上传文件的组件使用 Ant Design 提供的 Upload 组件，该组件可以在服务端在上传完成后返回一个数据结构，让前端记录该次上传的状态，比如返回一个 url 让前端可以进行该文件的下载。上传文件一般使用 POST 方法，因为 POST 传输数据没有体积的限制。

3.4.2 解析试卷

因为上传的是一个 xlsx 表格文件，所以需要对其进行读取和解析数据。Pandas 就是在此委以重任。

```
// 读取上传的文件建立矩阵
df = pd.DataFrame(pd.read_excel(file_path))
// 读取各列的数据并进行处理入库
for field in fields:
setattr(paper, field,
```

```
"@".join(df[field].dropna()
.astype("str")
.map(str.strip)
.values))
db.session.commit()
```

通过简单的几行代码，就可以把复杂的解析试卷和记录数据库的工作完成。

3.5 数据分析

管理员上传的试卷文件中有着每张试卷的得分指标以及每个问题每个答案的得分，在用户做完每套试卷后，数据库都存着用户回答每个问题的答案，所以我们要对这个每个答案的得分进行分析得出得分同时还要计算该得分项在所有用户中的占比以及每个问题的最多人选答案。

3.5.1 新开线程

这是一个复杂的过程，当你的试卷题目变多，分析将会是一个消耗时间和性能的过程。Ajax 是一个请求响应的过程，所以分析的过程不能阻塞 HTTP 的响应，否则会造成前端等待时间过长甚至造成超时，这不是一个好的体验，所以我们要先给页面一个响应后新开一个线程进行数据分析的过程，同时数据库中也需要对该成绩有一个状态的记录，是否在分析或者分析完毕了，通知前端什么时候可以进行查看分析结果。

```
@copy_request_context
def new_thread():
...
t = Thread(target=new_thread)
t.start()
```

新开一个线程

当新开一个线程的时候，如果在函数中使用了和 Flask 相关的工具，是访问不到这次请求的数据的，所以要使用 copy_request_context 这个装饰器来复制一个 request 请求的上下文。

3.5.2 计算试卷平均分

当我们计算出每个用户的得分以后，我们可以计算整个试卷的平均得分，使用 numpy 可以方便的进行批量的计算。

```
scores = np.array([list(map(int,
grade.score.split("@"))) for grade in grades])
paper.average = "@".join(map(str,
map(lambda x: round(x, 2), score_sum / finished_count)))
```

使用 numpy 建立一个 numpyList 类型的列表，因为其类型重写了 ___truediv___ 魔法函数，相当于 C++ 中的重载运算符，重写了除法的运算方式，可以方便得出每个指标的平均值。

3.5.3 计算每个问题最多回答的答案

想得到每个问题最多人回答的答案，我们需要对每个问题的答案进行计数找到最多的那个。通过自己循环编写当然可以实现，可 Python 作为一门受数据分析追捧的语言，当然不会那么麻烦。

```
choices = np
.array([grade.answers.split("@") for grade in grades]).T
most_choice = "@".join(map(lambda x: Counter(x)
.most_common(1)[0][0],
choices))
```

T 可以将一个矩阵转置。Counter 是 Python 内置的一个计数器类，可以直接获取每个元素出现的次数。

几行代码我们就可以完成复杂的计数运算。

第 4 章　业务功能

如果说一个项目的架构和数据库设计是一个项目的骨架，那么业务代码就是在其骨架上面生长的骨肉。一个好的骨架能够让血肉长的更加健康。

4.1 RESTful API

REST 全称是 Representational State Transfer，中文意思是表述（编者注：通常译为表征）性状态转移。它首次出现在 2000 年 Roy Fielding 的博士论文中，Roy Fielding 是 HTTP 规范的主要编写者之一。他在论文中提到：” 我这篇文章的写作目的，就是想在符合架构原理的前提下，理解和评估以网络为基础的应用软件的架构设计，得到一个功能强、性能好、适宜通信的架构。REST 指的是一组架构约束条件和原则。” 如果一个架构符合 REST 的约束条件和原则，我们就称它为 RESTful 架构。

REST 本身并没有创造新的技术、组件或服务，而隐藏在 RESTful 背后的理念就是使用 Web 的现有特征和能力，更好地使用现有 Web 标准中的一些准则和约束。虽然 REST 本身受 Web 技术的影响很深，但是理论上 REST 架构风格并不是绑定在 HTTP 上，只不过目前 HTTP 是唯一与 REST 相关的实例。所以我们这里描述的 REST 也是通过 HTTP 实现的 REST。

RESTful API 有以下的特征：

1. 每一个 uri 代表一种资源。
2. 客户端和服务器之间，传递这种资源的某种表现层。
3. 客户端通过四个 HTTP 动词 (GET、POST、DELETE、PUT)，对服务器端资源进行操作，实现” 表现层状态转化”(增删改查)。
4. URL 中通常不出现动词，只有名词。
5. 使用 JSON 不使用 XML。

4.1.1 在 Flask 中编写 RESTful API

Flask 作为一个轻量级的后端框架，不使用扩展的情况下，就可以使用路由功能提供 RESTful 接口。

```
@app.route("/index", method=["GET", "POST", "PUT", "DELETE"])
def Index():
    if request.Method == "GET":
        ...
```

```
elif request.Method == "POST":
  ...
elif request.Method == "PUT":
  ...
elif request.Method == "DELETE":
  ...
```

Flask 编写 RESTful API 接口

4.1.2 前端使用 RESTful 接口

axios 是一个封装了 xhr http 服务的一个前端 http 请求工具。

```
axios({
  method: "get" / "post" / "put" / "delete",
  url: "",
  data: ""
})
```

4.2 使用 Redux 存储网络请求状态

Redux 是 JavaScript 状态容器，提供可预测化的状态管理。可以让你构建一致化的应用，运行于不同的环境（客户端、服务器、原生应用），并且易于测试。其使用 Immutable 数据结合 React 能让页面相应数据进行最小颗粒度的更新 DOM 结点。

因为 Ajax 都是异步的一个请求过程，需要在回调函数内对数据进行处理，这种就会出现当依赖的 Ajax 请求过多的情况下，不断的编写和嵌套回调函数的时候代码就会变得非常难以阅读和维护，所以使用 Promise 封装一种通用的 Ajax API 同时用 Redux 进行对数据的保存是非常有必要的。

当进行一个 Ajax 请求的时候，我们默认出发一个 Redux Action 将 Ajax 的数据储存在 Redux 中然后使用 React-Redux 的 Connect 函数将我们的组件根据 Redux 中的状态变化进行更新。

```
const registHttpAction: (actionName: string) => void =
(actionName: string) => {
  reducerActions[actionName] = createAction(
  actionName.toUpperCase(),
  (payload: IHttpResponse) => payload);
```

```
  actionsMap[actionName] = async (params: IHttpRequest):
  Promise<void> => {
  try {
    const response: AxiosResponse = await axios(params);
    const { headers } = response;
    if (headers.redirect && headers.redirect === "login") {
      if (!m) {
        m = message.error("验证失败，需要登陆，
␣␣␣␣␣␣␣␣三秒后自动跳转␣Authenticate␣fail,
␣␣␣␣␣␣␣␣require␣login,␣auto-redirect␣after␣3s");
      }
      setTimeout(() => {
        location.href = "/login?redirect=" + location.pathname;
        if (m) {
          m();
        }
      }, 3000);
    } else if (headers.redirect && headers.redirect === "admin") {
      message.error("没有权限␣Forbidden", 3);
    }
    store.dispatch(reducerActions[actionName](response.data));
    return Promise.resolve();
  } catch (err) {
    message.error("服务器错误␣Server␣Internal␣Error");
    return Promise.reject();
  }
};
```

Redux Http Action 注册函数

注意这里我们一定要在 PromiseLike Function 里面返回一个 Promise 对象来表示该 Promise 的状态，否则在使用 Await 关键字的时候，不会进行阻塞。

这样做的好处就是，通过封装了 axios 进行 http 请求的时候可以有一个公共的地方存储返回的数据，同时还可以在发生错误的时候或者在权限不够的时候读取 headers 直接取消渲染并给出提示让用户进行相应的操作。

4.3 账号系统

一个系统想要有人使用，给每个人都提供服务，都需要一个完善的账号系统，每个人都有自己的账号，进行登陆，使用系统，更改设置等等。

4.3.1 密码加密

出于安全考虑，用户的密码不通过明文保存，除非你对自己的运维系统有着极高的自信，当然这样也不建议这样做。毕竟人外有人，天外有天，如果没有被盗取那么说明你的系统价值还不是那么的高。

werkzeug.security 是一个 Python 的密码包，其中的 generate_password_hash 的方法可以创建 MD5 加密的字符串，check_password_hash 可以用于验证密码是否正确。

```
@property
def password(self):
  raise TypeError("cannot␣read␣a␣user's␣password")

@password.setter
def password(self, password):
  self.password_hash = generate_password_hash(password)

def verify_password(self, password):
  if not password:
    return False
  return check_password_hash(self.password_hash, password)
```

用户密码的创建和验证

4.3.2 token 的创建和验证

token 用于用户的登陆状态的保存，其通过 cookie 的形式保存在 request 头中，后端通过对 response 的 set-cookie 字段对网页的该域下进行 cookie 的设置。同时 token 可以设置过期时间。

```
def generate_confirm_token(self):
s = Serializer(current_app.config["SECRETE_KEY"],
expires_in=60 * 5, salt=current_app.config["CONFIRM_TOKEN"])
token = s.dumps({"user": self.username})
```

```
return token.decode()
```

得到加密 token 的代码

初始化对象的时候向 Serializer 内部传入解密字符串，token 的有效时长，以及加密的盐。此 token 的时长是五分钟，将用户的用户名做了加密通过 HTTP 传给前端，这里要注意，因为 token 生成的是二进制字符串，我们先要把它解密成普通字符串以后才能设置 cookie，前端下次请求的时候会带上这个数据，后端再进行解密即可获得用户的状态。

```
response.set_cookie("token", token, max_age=max_age,
httponly=True)
```

response 设置 header 中的 cookie

```
s = Serializer(current_app.config["SECRETE_KEY"],
salt=current_app.config["CONFIRM_TOKEN"])
try:
data = s.loads(token)
if username != data["user"]:
return -2, "Bad signature"
return 0, "Confirm success"
except SignatureExpired:
return -1, "Signature expired"
except BadSignature:
return -2, "Bad signature"
```

验证 token 的代码

同样的，在解密的过程中需要获得 Serializer 的实例，当 loads 一个 token 的时候，如果验证失败或者 token 超时，代码会抛出异常，SignatureExpired 和 BadSignature，我们需要对对应的状态进行处理给前端响应。

4.3.3 注册与邮箱验证

现在许多软件注册都需要用手机号，并且一个手机号只能注册一次，但在此发送短信需要使用第三方收费服务，所以我使用替代的邮箱来进行验证。既然需要邮箱，我们就需要写一个页面来进行验证结果的展示，同时也用到了 token。以一个注册用户名为 cindy 的用户来看，我们把 username: "cindy" 通过 Serializer 进行加密生成一个 token，同时我们的注册页面的 pathname 是/confirm，那么我们给该用户注册成功发送邮箱中的链接就是 http(s)://origin/confirm?token={token}，当访问这个页面的时候我们把 token 传给后端，后端进行解密然后验证 token，把数据库中相应用户的邮箱激活。当然如果在验证时候 token 过期了还可以重新发送验证邮箱。

4.3.4 登陆与找回密码

登陆的时候会首先查看是否邮箱已经激活，如果没有激活则发送新的验证邮箱去激活账号才能登陆。同样的如果此账号忘记了密码，则可以通过邮箱进行密码找回和重新设置密码。

4.4 上传文件与解析试卷

心理评测系统最重要的是试卷，作为管理员需要进行试卷的编写和上传，后端要对上传的文件进行分析然后记录到数据库中。

4.4.1 上传文件

上传文件的组件使用 Ant Design 提供的 Upload 组件，该组件可以在服务端在上传完成后返回一个数据结构，让前端记录该次上传的状态，比如返回一个 url 让前端可以进行该文件的下载。上传文件一般使用 POST 方法，因为 POST 传输数据没有体积的限制。

4.4.2 解析试卷

因为上传的是一个 xlsx 表格文件，所以需要对其进行读取和解析数据。Pandas 就是在此委以重任。

```
// 读取上传的文件建立矩阵
df = pd.DataFrame(pd.read_excel(file_path))
// 读取各列的数据并进行处理入库
for field in fields:
  setattr(paper, field,
```

```
    "@".join(df[field].dropna()
    .astype("str")
    .map(str.strip)
    .values))
db.session.commit()
```

通过简单的几行代码，就可以把复杂的解析试卷和记录数据库的工作完成。

4.5 数据分析

管理员上传的试卷文件中有着每张试卷的得分指标以及每个问题每个答案的得分，在用户做完每套试卷后，数据库都存着用户回答每个问题的答案，所以我们要对这个每个答案的得分进行分析得出得分同时还要计算该得分项在所有用户中的占比以及每个问题的最多人选答案。

4.5.1 新开线程

这是一个复杂的过程，当你的试卷题目变多，分析将会是一个消耗时间和性能的过程。Ajax 是一个请求响应的过程，所以分析的过程不能阻塞 HTTP 的响应，否则会造成前端等待时间过长甚至造成超时，这不是一个好的体验，所以我们要先给页面一个响应后新开一个线程进行数据分析的过程，同时数据库中也需要对该成绩有一个状态的记录，是否在分析或者分析完毕了，通知前端什么时候可以进行查看分析结果。

```
@copy_request_context
def new_thread():
        ...
t = Thread(target=new_thread)
t.start()
```

新开一个线程

当新开一个线程的时候，如果在函数中使用了和 Flask 相关的工具，是访问不到这次请求的数据的，所以要使用 copy_request_context 这个装饰器来复制一个 request 请求的上下文。

4.5.2 计算试卷平均分

当我们计算出每个用户的得分以后，我们可以计算整个试卷的平均得分，使用 numpy 可以方便的进行批量的计算。

```
scores = np.array([list(map(int,
grade.score.split("@"))) for grade in grades])
paper.average = "@".join(map(str,
map(lambda x: round(x, 2), score_sum / finished_count)))
```

使用 numpy 建立一个 numpyList 类型的列表，因为其类型重写了 ___truediv___ 魔法函数，相当于 C++ 中的重载运算符，重写了除法的运算方式，可以方便得出每个指标的平均值。

4.5.3 计算每个问题最多回答的答案

想得到每个问题最多人回答的答案，我们需要对每个问题的答案进行计数找到最多的那个。通过自己循环编写当然可以实现，可 Python 作为一门受数据分析追捧的语言，当然不会那么麻烦。

```
choices = np
.array([grade.answers.split("@") for grade in grades]).T
most_choice = "@".join(map(lambda x: Counter(x)
.most_common(1)[0][0],
choices))
```

T 可以将一个矩阵转置。Counter 是 Python 内置的一个计数器类，可以直接获取每个元素出现的次数。

几行代码我们就可以完成复杂的计数运算。

[1] 未来科技. jQuery 实战从入门到精通 [J]. 水利水电出版社, 2017.

[2] 叶维忠. Python 编程从入门到精通 [J]. 人民邮电出版社, 2017.

[3] 杜文洁. 高等学校毕业设计（论文）指导教程——电子信息类专业 [J]. 水利水电出版社, 2015.

[4] 刘一奇. React 与 Redux 开发实例精解 [J], 2016.

[5] 程墨. 深入浅出 React 和 Redux[J]. 机械工业出版社, 2017.

[6] 程墨. 深入浅出 RxJS[J]. 机械工业出版社, 2018.

[7] 张静. 慧眼识人——心理学评测在人才管理中的应用 [J]. 辽宁科技, 2016.

[8] 吴浩麟. 深入浅出 webpack[J]. 电子工业出版社, 2018.

[9] Crockford. JavaScript：The Good Parts[J]. Southeast University Press, 2008.

[10] BANKS A, PORCELLO E. Python 编程从入门到精通 [J]. China Electric Power Press, 2017.

[11] JAWORSKI M, ZIADE T. Expert Python Programming[J]. PacktPublishing, 2016.

[12] RAMALHO L. Fluent Python[J]. Posts and Telecommunications Press, 2017.

[13] GORELICK M, OZSVALD I. High Performance Python[J]. Posts and Telecommunications Press, 2017.

[14] Miguel Greenberg. Flask Web Development[J]. Posts and Telecommunications Press, 2018.

[15] Gustave Le Bon. The crowd: a study of the popular mind[J]. Guangxi Normal University Press, 2011.

致谢

四年的大学学习生活在即将划上一个句号，而于我的人生来说却仅仅只是一个逗号，我将面对新的征程的开始。本研究及论文是在我的导师刘传文的亲切关怀和耐心的指导下完成的。伟人、名人固然为我所崇拜，可是我更迫切地想要把我的敬意献给给一位平凡的人，我的导师刘传文老师。也许我不是您最出色的学生，但您却是我所最尊敬的老师。您是如此的治学严谨，学识渊博，视野广阔，思想深刻，您用心为我营造一种良好的学术氛围，让我的论文更加的严谨。至此论文付梓之际，我的心情无法保持平静，从开始选择课题到论文的顺利答辩，有无数可敬的师长、朋友给了我很多的帮助，在这里请您接受我诚挚的谢意！最后，再次对那些在论文完成过程中，关心、帮助我的同学和朋友们表示衷心地感谢！