**后端设计文档**

**一、数据库设计**

**（一）数据库选择**

选择MySQL关系型数据库。Client/server SQL数据库引擎力求实现数据的共享存储，强调可伸缩性、并发性、集中性和可控性。

**（二）数据表设计**

设计用户表如下，用于保存用户个人信息。

|  |
| --- |
| SQL CREATE DATABASE test\_db; use test\_db; CREATE TABLE Users (  sid varchar(20),  nickname varchar(255) ); |

**（三）具体实现**

代码实现中，使用SQLAlchemy库以面向对象的形式操作数据库，代码中提供了包括增添、查找用户及其昵称的方法。

|  |
| --- |
| Python # User class User(db.Model):  \_\_tablename\_\_ = 'Users'  # id = db.Column(db.Integer, primary\_key=True, autoincrement=True)  sid = db.Column(db.String(20), primary\_key=True, nullable=False)  nickname = db.Column(db.String(255), nullable=False) |

**二、模块设计**

**（一）总体设计**

**1.模块设计**

项目后端采用flask框架，较其他同类型框架更为灵活、轻便、安全且容易上手。

后端整体由六个模块组成。config：保存参数；saver：提供保存课表的方法；webvpn：提供登录、获取课程表等底层方法；schedule：提供获取课表的方法；user：提供次顶层登录和数据库方法；app：提供各种顶层方法。

**2.接口逻辑**

后端涉及到需要访问的接口如下。

|  |
| --- |
| Python # login base\_url = "https://webvpn.bit.edu.cn" login\_url = base\_url + \  "/https/77726476706e69737468656265737421fcf84695297e6a596a468ca88d1b203b/authserver/login?service=https%3A%2F%2Fwebvpn.bit.edu.cn%2Flogin%3Fcas\_login%3Dtrue" need\_captcha\_url = base\_url + \  "/https/77726476706e69737468656265737421fcf84695297e6a596a468ca88d1b203b/authserver/checkNeedCaptcha.htl" get\_captcha\_url = base\_url + \  "/https/77726476706e69737468656265737421fcf84695297e6a596a468ca88d1b203b/authserver/getCaptcha.htl"  # schedule schedule\_init\_url = base\_url + \  "/http/77726476706e69737468656265737421faef5b842238695c720999bcd6572a216b231105adc27d/jwapp/sys/funauthapp/api/getAppConfig/wdkbby-5959167891382285.do" schedule\_lang\_url = base\_url + \  "/http/77726476706e69737468656265737421faef5b842238695c720999bcd6572a216b231105adc27d/jwapp/i18n.do?appName=wdkbby&EMAP\_LANG=zh" schedule\_now\_term\_url = base\_url + \  "/http/77726476706e69737468656265737421faef5b842238695c720999bcd6572a216b231105adc27d/jwapp/sys/wdkbby/modules/jshkcb/dqxnxq.do" schedule\_all\_terms\_url = base\_url + \  "/http/77726476706e69737468656265737421faef5b842238695c720999bcd6572a216b231105adc27d/jwapp/sys/wdkbby/modules/jshkcb/xnxqcx.do" schedule\_url = base\_url + \  "/http/77726476706e69737468656265737421faef5b842238695c720999bcd6572a216b231105adc27d/jwapp/sys/wdkbby/modules/xskcb/cxxszhxqkb.do" schedule\_date\_url = base\_url + \  "/http/77726476706e69737468656265737421faef5b842238695c720999bcd6572a216b231105adc27d/jwapp/sys/wdkbby/wdkbByController/cxzkbrq.do" |

后端登录、获取课表的流程如下。

1. 登录部分，首先访问login\_url ，获取头中的Set-Cookie、页面中id分别为pwdEncryptSalt和execution的input标签的value（包裹在id为pwdFromId的form中）。几次密码错误后需要获取验证码captcha，获取验证码时使用访问登录页面获取的cookie。
2. 然后进行用户登录，login\_url 接口需要将用户密码进行加密，需要使用上一步获取到的pwdEncryptSalt字段值，并执行相应的js文件。然后使用用户名、加密后的密码、execution字段值、cookie、captcha作为接口参数，经过一系列重定向，重定向过程中需要一直携带cookie，最终成功登录。
3. 对于webvpn来说，之后使用同一个cookie就可以了，只不过访问每一个子模块或多或少都需要前序验证（经历一大堆重定向）。另外，由于webvpn相当于代理了内部的Cookie，所以内部的这些Cookie过期时间是不可知的，只能是用到相关服务的时候都把前序验证流程走一遍。
4. 对于非webvpn的直接访问来说，需要统一身份认证时会被重定向到登录页面，这时候只要携带着登录时拿到的Cookie就会被自动重定向回要访问的页面，并且下发可能需要的对应站点的Cookie。
5. 接下来可以获取用户课表。首先需要访问schedule\_init\_url 接口获取课表权限，然后访问schedule\_lang\_url 接口设置语言为中文，访问schedule\_date\_url 接口获取学期开始日期。接下来可以访问schedule\_url获取指定学期课表。

**（二）详细设计**

**1.config**

config文件中保存了数据库连接url、项目路径、保存文件路径、ics文件路径前缀等，为程序提供所需要的各种可变参数，便于修改。

|  |
| --- |
| Python # 数据库类型+数据库操作引擎://用户名:密码@主机名:端口/数据库名 db\_url = "mysql+pymysql://root:xxx@127.0.0.1:3306/test\_db" requests\_proxy = "" max\_upload\_size = 24 \* 1024 \* 1024 project\_path = "D:\\pycharm\\swProject\\" ics\_save\_path = project\_path + "save\\ics" json\_save\_path = project\_path + "save\\json" ics\_url\_prefix = "http://localhost:5000/download?type=ics&filename=" psw\_js\_path = project\_path + "EncryptPassword.js" |

**2.saver**

saver部分提供了将课表内容保存为本地ics和json文件的方法，分别由save\_ics和save\_json方法实现。其中，save\_json方法输入参数包括获取的原始json课表信息，在内部对其进行处理，筛选出需要保留的字段并更名，文件命名格式为sid + term + 'schedule.json'

|  |
| --- |
| Python # save schedule in icalendar type def save\_ics(filename, data)  # save schedule in json type def save\_json(sid, term, content) |

**3.webvpn**

webvpn部分提供登录、获取课程表等底层方法，具体包括初始化登录、登录、查询验证码、获取课表方法，还包括了重定向方法redirection，每个方法由输入参数获得需要访问实际接口所需要的参数。

其中，get\_schedule\_term方法在获取指定学期课表的时候，即按照接口逻辑部分所述，依次访问对应接口，最终得到用户课表。在获取到课表后， 调用saver.save\_json方法，简化课表数据，并将其以json格式保存在本地。

|  |
| --- |
| Python # redirection def redirection(url, head={}, data={}) # init\_login def init\_login() # captcha requirement check def need\_captcha(sid) # get captcha def get\_captcha(cookie) # login def login(username, password, execution, cookie, captcha="") # get schedule of specified term def get\_schedule\_term(sid, cookie, term='') |

json文件格式如下所示。

|  |
| --- |
| JSON {  "courses": [  {  "COURSENAME": "软件工程基础",  "COURSENO": "100081012",  "TEACHERNAME": "高玉金",  "CLASSNO": "07112003,07112004,07712201",  "ROOMNO": "综教A501",  "STARTTIME": "09:55",  "ENDTIME": "12:20",  "STARTWEEK": 9,  "ENDWEEK": 16,  "DETAIL": "9-16周 星期一 3-5节 综教A501,9-16周 星期四 6-7节 综教A501",  "WEEKINFO": "0000000011111111",  "WEEKDAY": 1,  "STARTSEQ": 3,  "ENDSEQ": 5  },  ...  ] } |

**4.schedule**

schedule部分提供获取课表的方法，具体包括获取课表ics文件url方法、获取周课表方法。

get\_ics\_url方法内部首先判断该用户请求的ics文件是否已经存在，若存在则直接返回对应url，否则调用webvpn.get\_schedule\_term方法，访问底层接口获取课表，并将其数据处理为icalendar格式，保存在本地，命名格式为sid + term + 'schedule.ics'。

get\_week\_schedule方法直接从本地json文件获取周课表，若对应文件不存在则提醒用户需要先获取课表，若存在则直接分析json文件，并按照前端要求的返回格式组织数据。

|  |
| --- |
| Python # save schedule and return url dict def get\_ics\_url(cookie, sid, term) # get schedule of specified week and return week schedule dict def get\_week\_schedule(sid, target\_week, term) |

**5.user**

user部分提供次顶层登录和数据库方法，具体包括初始化方法、验证方法，这两个方法封装了底层webvpn内部方法，分别为webvpn.init\_login、webvpn.login，共同实现了用户的登录。还包括操作数据库用户表的方法。

|  |
| --- |
| Python # initialization def init(sid) # verification def verify(username, password, execution, cookie, captcha="") # set nickname def set\_name(sid, nickname) # get nickname def get\_name(sid) |

**6.app**

app部分提供各种顶层方法，包括登录方法verify，接收参数为用户学号和密码，内部封装user部分的init和verify方法，实现访问登录界面、用户密码加密、进行登录的流程；设置和获取用户昵称方法set\_name和get\_name，对数据库进行增添与查找；获取用户课表方法get\_url，接收参数为用户学号、cookie、term，调用schedule.get\_ics\_url方法获取对应ics文件url，并返回给前端；获取周课表方法get\_week\_course，接收参数为用户学号、week、term，调用schedule.get\_week\_schedule方法获取周课表，并返回给前端。此外，还有一个download方法，接收参数为文件名和文件类型，用于直接下载对应的ics或json文件课表。

|  |
| --- |
| Python @app.route("/verify/", methods=['GET']) def verify()  @app.route("/setname/", methods=['GET']) def set\_name()  @app.route("/getname/", methods=['GET']) def get\_name()  @app.route("/geturl/", methods=['GET']) def get\_url()  @app.route("/weekcourse/", methods=['GET']) def get\_week\_course()  @app.route("/download/") def download(): |