数据库期末大作业 实验报告

小组成员及分工：

叶盛源（组长） 17341190 负责： 产品设计、数据库设计及操作、后端编写、实验报告编写

严宏昌 17341180 负责：产品设计、数据库设计及操作、界面设计、实验报告编写

郑昊怀 17341210 负责：数据库设计、前端开发、界面设计、PPT制作、运维

赵俊威 17341207 负责：数据库设计、前端开发、界面设计、PPT制作、运维

第一部分：应用需求介绍

开发平台选择：

因为考虑到开发能力的问题，传统的安卓和IOS程序开发往往需要学习很多新的框架或者需要用特定的大型开发软件开发，学习成本比较高，还有兼容性的问题，安卓端和IOS端并不能很好的相通，因此我们选择了开发微信小程序。

微信小程序的优点在于开发语言的基础是基于html和css的，我们已经有一定的基础，同时也有丰富的组件可以直接使用，不需要全部都自己编写。并且微信小程序很轻便，打开微信就可以使用，十分便捷，可以更好的吸引他人体验和使用。

开发背景：

我们本次开发的是一个有温度的、智能的大学生 心理健康小程序——LIGHT。

我们之所以要开发这款软件，是因为近段时间里，我们在网上各种热搜榜时时能看见因为心里问题而得重病，或者甚至结束了生命的悲惨事件发生，比如：前几周南方周末的 《“不寒而栗”的爱情:*北大自杀*女生的聊天记录》 就令我深有感触很难想象北京大学的高材生，竟然会有这样的心理疾病，却没有取寻求进一步的帮助。

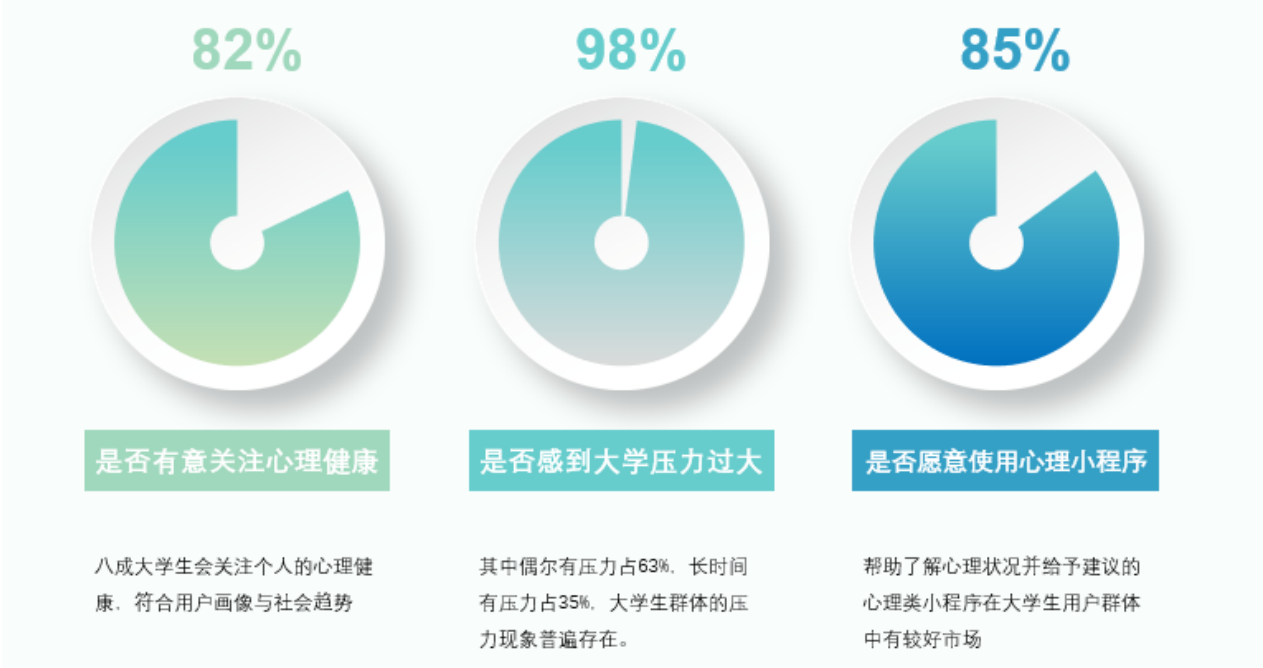
我们觉得，很多时候都是因为这些大学生缺少对自己心理健康的认识，缺少能学习和了解客服心里疾病的方法，缺少和其他有相同经历大学生交流的机会，使得很多大学生往往会忽略小的心理疾病问题，最后到了影响生活、学业，甚至危及生命的地步。

如果我们能提供这样一个学习交流的平台，或许很多人就可以很早的意识到自己的心理疾病和问题，比如事件种的女主，如果能早早意识到自己的状况的危险，就可以更早的寻求他人的帮助，也许就不用付出生命的代价。

这款产品的实用人群是： 只要是重视个人心理健康，对心理学知识感兴趣的大学生 都可以使用我们的产品，而产品的主要用户是处于短暂的 情绪低落时期、因为心理问题而影响到自己学业和生活的 大学生。对于有比较严重的心理问题的同学，我们也能给 他们建议、鼓励，指引他们获得进一步的帮助。

简单问卷调查：

在开发之前我们进行了一个小规模的调查， 回收有效问卷 208 份。我们把用户定位到了大学生群体，因为该群体对自我心理健康有更高的关注，且大学生遭受心理困扰的情况十分普遍，着重体现于学业压力、人 际关系和对未来的规划。可以看到我们简单的调查结果如下：



第二部分 应用系统的设计

1 功能设计

我们先对系统的功能进行了设计。因为是一款心理问题的软件，我们经过小组讨论和分析，设计软件的主要功能有以下几点：

- **智能问答机器人：**允许人和机器人对话。建立简单的问答对应数据表，根据相似度匹配标准的问

题，找到对应的答案并返回结果到界面上。这个功能让人找到聊天解压的对象，目前还是简单的相似度匹配，未来可以加入自然语言的模型提高准确度。

- **解压舒压板块：**这个板块包括一些小游戏和音乐等组件，这个尚未开发完全
- **专业心理测评表：**这个板块是从网上下载专业的心理测评的问答和结果数据，让用户可以进行自我的心理测评并会给与一个测评的报告，测评的信息都从专业网站上获取。这个板块尚未开发完全。
- **心理知识干货和心理课堂：**允许用户自己编写文章发布，或者发布一些心理课堂的视频。当然小程序这边也会提供一些专业的文章，和教学视频给用户去查看和学习。用户可以对文章进行点赞，收藏和评论等。
- **互助交流社区：**社区板块是让用户可以自由发布帖子进行交流学习，用户可以自己发帖，也可以看到热度最高的帖子或者最新发布的帖子，可以进行浏览，回复，评论，收藏，点赞等功能。
- **个人中心：**这个中心允许用户查看自己的个人信息。查看心理测评结果，发送的文章，帖子，和收藏的内容等，用户还可以进行每日签到获取积分。后期还准备继续开发允许用户自定义个人信息，积分商城等诸多功能。

2 界面设计和前端开发

划分了功能后就可以进行界面的设计，因为微信小程序的限制，需要有bar栏，因此对bar栏继续宁划分成首页，测评，干货，社区，我的板块。因为要使用一些高级组件colorUI，所以图标，页面设计都是从colorUI中选取一些图片或者布局套用，模仿一些程序的界面，比如苹果应用商店APP Store的版块设计等。考虑到是心理APP，所以整体颜色偏浅色，使用了浅蓝绿色作为主色调。

前端开发方面，我们使用了微信原生的官方编辑器：**微信web开发者工具**和微信的原生开发语言来进行开发。我们前端的代码都放在github上， 地址为： https://github.com/DNB2019/MiniProgram，需要的可以查看。

本次实验因为是数据库的期末大作业，重点不在于前端，而且代码比较繁琐重复度很高，因此没有展示在实验报告中，具体的实现也就不深入分析了。

2 后端开发

后端使用了python3的轻量级web应用框架**flask**。

前后端交互方式

后端主要是负责处理一些逻辑，提供算法服务和数据库交互等。Flask主要是和前端编写的js用json格式进行信息交互的，flask提供了函数给我们从json格式中解析出数据，或者把数据打包成json格式返回前端：

```
@app.route('/OpenID', methods=['GET', 'POST'])
def getOpenID():
    '''
    根据前端传来的code 从微信服务器获取openid和会话id并返回
    '''
    code = str(request.get_json()['code'])
    # JSCODE = base64.b64decode(code) # 传过来的code是base64的，需要解码
    req_params = {
        'appid': APPID,
```

```
        'secret': SECRET,
        'js_code': code,
        'grant_type': 'authorization_code'
    }
    wx_login_api = 'https://api.weixin.qq.com/sns/jscode2session'
    response_data = requests.get(wx_login_api, params=req_params)  # 向API发起
GET请求
    res = response_data.json()  # 将获取的session_key 和 openid装填称json
    # print(res) # 输出结果
    return Response(json.dumps(res), content_type='application/json')
```

比如上面这个部分是在用户登录小程序的时候我们取微信获取用户具体信息的部分。我们通过提供一个URL `/OpenID` 给前端，前端请求这个URL的时候，Flask就会新建一个线程，并调用这个函数处理请求。`request.get_json()` 从请求的json文件中解析中字段 `code` 对应的数据，然后进行一系列的逻辑处理，向微信的网站请求用户信息，然后用 `json.dumps` 打包成json文件，放在 `Response` 中返回结果。

后端连接数据库：

后端连接数据库使用一个很出名的数据库操作框架flask-alchemy。这个框架是将每个数据表当作一个类，每个记录的属性就是这个类的属性，之后对数据库的操作就相当于对python中的类对象实例进行操作了，这个框架内部还是用pymysql驱动的，但他可以避免直接在python中敲sql语句，比较便捷。我们通过简单的配置语句和数据就能建立连接：

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql+pymysql://root:12345@127.0.0.1:3306/light?charset=utf8mb4'
# app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://{}:{}@{}:{}/light?
charset=utf8mb4'.format(MYSQL_USERNAME,MYSQL_PASSWORD,MYSQL_IP,MYSQL_PORT)
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
DB_light = SQLAlchemy(app=app)
```

我们可以自己编写python类的文件来建立数据库，但因为我们数据库使用sql语句建立的，所以我们不需要自己用python代码建立数据库，但我们需要将远程的已有数据表打包成python文件，我们可以使用这个框架的一个插件，并输入指令：

```
sqlacodegen mysql://root:{}@{}:3306/light > models.py
```

就会打包生成一个文件model，这个文件包含了数据表的所有关系，**这里就不展示代码，完整代码在附录中附上**，我们就可以直接操作python对象来操作数据库了。举个例子，假如我们有了用户的数据需要插入到数据表中：

```
u = models.User(openid=user_data['openid'],
                nickName=user_data['nickName'],
avatarUrl=user_data['avatarUrl'],
                gender=user_data['gender'],
                province=user_data['province'],
                city=user_data['city'])
db.session.add(u)
db.session.commit()
db.session.close()
```

创建一个User的实体，然后给它的属性赋值，最后提交数据，和数据库SQL语句的操作流程差不多，其实它的驱动pymysql本质上也是在运行sql语句。

有了前后端交互，又有了数据库操作的框架，我们就可以处理前台发来的请求，返回他们需要的数据，或者编写复杂的算法处理前台的逻辑并返回结果了。

3 数据库开发和设计（重点）

本次数据库使用了Mysql。下面是数据库的设计部分，因为数据库表比较多，有一些也没有完全开发，所以我们选取几个有代表性板块来逐个描述。**完整的SQL代码我们放在了实验报告的最后一部分附录中**。

个人中心：

首先需要建立一个user表，保存所有登录用户的信息，默认保存微信官方提供的用户微信的个人信息。我们需要的关键字段如下：

- openid：微信官方提供的唯一确定某一个用户的id号，作为主键。

其他的还包括，昵称，背景图片，性别等等各种个人的信息数据，为了方便后台处理，我们都进行了非空的设置，**数字默认是0，字符串都默认是空串，不会出现默认是NULL的情况**。使用下列SQL创建：

```
CREATE TABLE `USER` (
`openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`nickName` varchar(100) CHARACTER SET utf8mb4 NULL DEFAULT '',
`avatarUrl` varchar(300) CHARACTER SET utf8 NULL DEFAULT '',
`gender` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`province` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`city` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`grade` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Register_Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,
`points` int unsigned NULL DEFAULT 0,
`use_days` int unsigned NULL DEFAULT 0,
PRIMARY KEY (`openid`)
);
```

然后我们还需要记录用户的历史记录，收藏记录等，这里就展示用户的收藏记录的数据表。要记录用户收藏的内容，我们需要以下关键字段：

- openid：外键，从用户的id引用
- Tag：因为用户可以收藏多种不同类型的，比如文章，音乐等，所以用Tag区分
- ID：收藏的篇章或视频等在对应Tag类型中的ID号

我们可以通过以下SQL语句创建：

```
-- 要主键自增
CREATE TABLE `COLLECT` (
-- `ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Tag` varchar(100) CHARACTER SET utf8 not NULL DEFAULT '',
`ID` int unsigned NOT NULL,
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY(`openid`, `Tag`, `ID`)
);
ALTER TABLE `COLLECT` ADD CONSTRAINT `fk_COLLECT_USER_1` FOREIGN KEY
(`openid`) REFERENCES `USER` (`openid`) ON DELETE CASCADE ON UPDATE CASCADE;
```

用户打开收藏记录栏，就会用用户id筛选出收藏的内容，并返回给前台展示。

智能问答机器人：

这个版块需要定义一张数据表，用来装所有人工编写的标准问题和标准回答对，进行相似度匹配并返回结果。我们主要需要ID，关键词，问题，回答字段，使用下列SQL语句建表：

```
CREATE TABLE `ROBOTQA` (
`ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`Key_Word` varchar(300) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Question` text NULL,
`Answer1` text NULL,
`If_RE` int NOT NULL DEFAULT 0
)
AUTO_INCREMENT = 1;
```

对于ID我们使用主键自增的模式，因为这样我们在后台插入数据到数据库的时候就不需要每次都自己生成一个随机的不同的id插入了，节省了工作量，而且自增也可以保证不会重复插入主键。

心理知识干货：

这个板块我们首先需要开一张表保存每一篇干货文章的数据，主键和外键字段在下面列出：

- Article_ID:唯一标识一篇文章，设置为主键，并且自增。
- Issuer_ID：外键，从用户的表中user_ID引用，要求数据库中的文章中必须是小程序的用户或者管理员发布的

其他的字段还需要：发布时间，标题正文作者昵称，点赞收藏转发数和背景图片链接等。使用的SQL语句建表如下：

```
-- 要主键自增
CREATE TABLE `ARTICLE` (
`Article_ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`Title` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Author` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Issuer_ID` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Issue_time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,
`Context` text NULL,
`Watches` int unsigned NULL DEFAULT 0,
`Likes` int unsigned NULL DEFAULT 0,
`Transmit` int unsigned NULL DEFAULT 0,
`Back_Image` varchar(300) CHARACTER SET utf8 NULL DEFAULT '',
`Tag` varchar(100) CHARACTER SET utf8 NULL DEFAULT ''
)
AUTO_INCREMENT = 1;
```

在我们点击进入文章之后，用户是可以给文章点赞收藏的，因此我们需要一张点赞表和一张收藏表。点赞表和收藏表类似，就一起说明：

- open_id: 外键，引用user表
- Tag：标记这个点赞属于那一个类型的，因为用户可以给文章，音乐，视频等点赞，所以用tag来区分属于那一个类别
- ID：标记这个点赞的目标是某个Tag中的ID号，比如属于文章类的5号文章。

上述三者共同构成主键，他们可以唯一确定一个记录，在mysql中多属性作为主键，如果查询第一个属性，是会加速查询速度，因为我们通过用户ID查询的情况比较多，因此把用户ID放在了第一个位置。使用下面的SQL语句建表：

```
CREATE TABLE `FAVOR` (
-- `ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Tag` varchar(100) CHARACTER SET utf8 not NULL DEFAULT '',
`ID` int unsigned NOT NULL,
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY(`openid`, `Tag`, `ID`)
);
ALTER TABLE `FAVOR` ADD CONSTRAINT `fk_FAVOR_USER_1` FOREIGN KEY (`openid`)
REFERENCES `USER` (`openid`) ON DELETE CASCADE ON UPDATE CASCADE;
```

进入文章后还有一个功能是用户可以对文章进行评论，因此我们还需要一个表记录用户评论的内容，主要字段如下：

- ID：主键，唯一索引一个评论

- Article_ID：评论的文章ID
- Commoner_ID：外键，从User中用户id引用

其他还需要评论内容，时间，被点赞数等。用下列的SQL语句创建：

```
-- 要主键自增
CREATE TABLE `COMMON_ARTICLE` (
`ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`Article_ID` int unsigned NOT NULL,
`Commoner_ID` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Content` text NULL,
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
`Likes` int unsigned NULL DEFAULT 0
)
AUTO_INCREMENT = 1;
```

社区版块

社区部分和干货部分几乎相同，因为只是帖子和文章的区别。首先我们需要有一张数据表储存所有的用户发出的帖子，主要的字段如下：

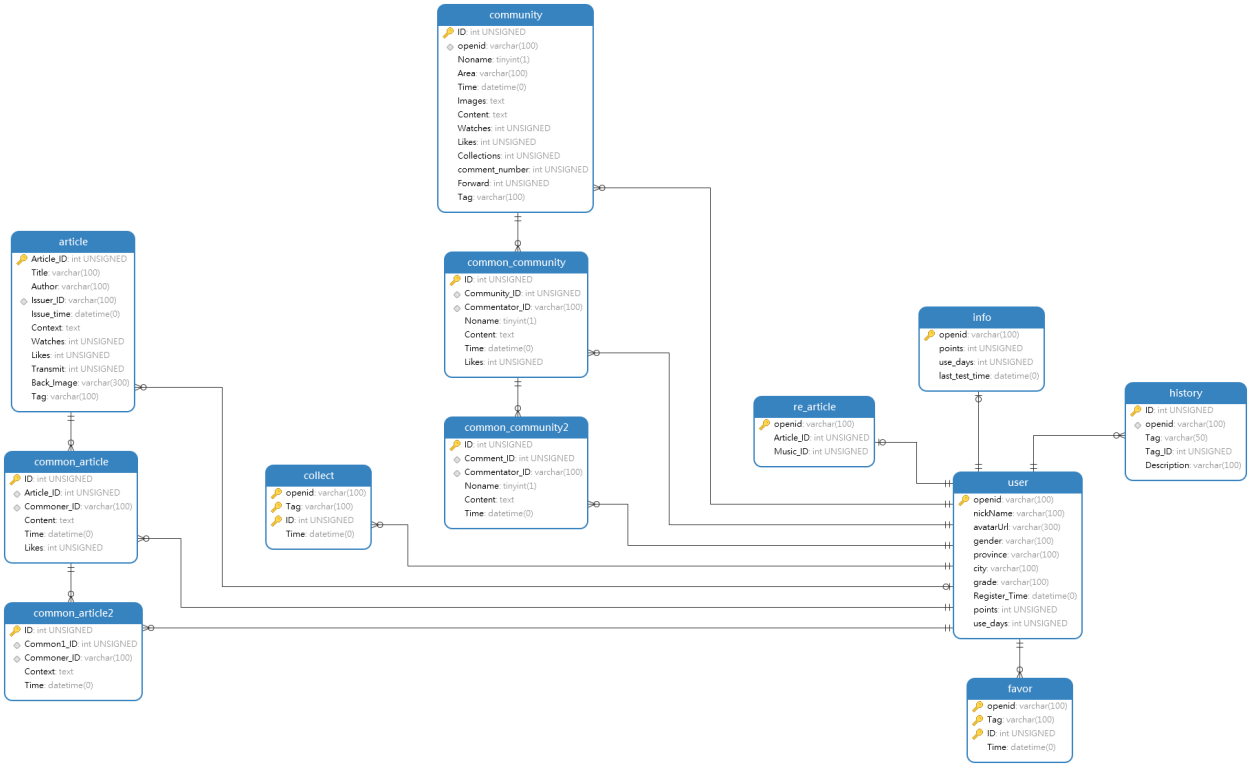
- ID：主键，唯一索引一个帖子。
- openid: 外键，从用户的表中user_ID引用，要求发布的帖子必须是用户发送的
- Time：时间，我们可用时间对发布的帖子进行排序在前台展示。

其他字段还包括：是否允许匿名，发布时间，发布板块，点赞评论收藏转发数等等。用下列SQL语句创建：

```
CREATE TABLE `COMMUNITY` (
`ID` int unsigned NOT NULL PRIMARY Key AUTO_INCREMENT,
`openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Noname` tinyint(1) DEFAULT 0,
`Area` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
`Images` text,
`Content` text NULL,
`Watches` int unsigned NULL DEFAULT 0,
`Likes` int unsigned NULL DEFAULT 0,
`Collections` int unsigned NULL DEFAULT 0,
`comment_number` int unsigned NULL DEFAULT 0,
`Forward` int unsigned NULL DEFAULT 0,
`Tag` varchar(100) CHARACTER SET utf8 NULL DEFAULT ''
)
AUTO_INCREMENT = 1;
ALTER TABLE `COMMUNITY` ADD CONSTRAINT `fk_COMMUNITY_USER_1` FOREIGN
Key(`openid`) REFERENCES `USER`(`openid`) ON DELETE CASCADE ON UPDATE CASCADE;
```

同时社区部分也可以评论，评论的表和文章评论表几乎相同，就不加赘述。

最后我们整合所有的数据表关系，绘制出了下面的ER图模型：



第三部分：系统实现界面展示

我们分各个版块录制了不同的视频来展示我们系统的功能，并在制作了一个**PPT**来帮助查看。可以点击让PPT播放，切换不同的页面PPT上的视频就会自动播放并展示功能。为防止PPT失效，附带了录制的视频在压缩包内。

下面对各个板块的页面截图展示，这只是部分功能，更具体的可以查看视频。

首页:


首页
晚上好！一点点
艰难的生活永无止境，但因此生长也无止境。
召唤小L
今日活动
每天活动一下，保持健康心理！
签到
解压小游戏
冥想音乐
不定期小挑战
每日一测
每天测评一下，了解真实的自己！
DASS-21
vConsole
万事皆有裂缝，那是光照进来的地方
-LIGHT开发团队-
首页
测评
干货
社区
我的

个人中心：


我的
一点点
陪伴你的第0天
30
我的心理档案
最近测试时间 2019-11-09
• 本周分数
中度抑郁
24 / 42
24 / 42
中度焦虑
12 / 42
轻度压力
结果根据每周一测（xxx测试）生成
查看完整档案
我的功能
我的收藏
历史浏览
测试记录
我的动态
vConsole
首页
测评
干货
社区
我的

社区：


11:10
LIGHT
搜索
交流圈
倾诉港湾
正能量供应站
学业压力
恋爱情
一点点
12-06 22:54
色彩美学是电影最重要的部分之一，除了烘托氛围，还能够传递出角色的内心情感。其中蓝色调的电影往往带着一股忧郁悲伤的感觉
0 0
匿名
12-06 22:53
vConsole
是我，是你，还是他
0 0
匿名
12-06 22:53
首页
测评
干货
社区
我的

干货和心理课堂：


干货知识
搜索
文章干货
你真的需要对象吗？也许你...
“你们吵架，我就涕泗横流...
连信推出社会心理服务中心...
怎样判断你的孩子是否有安...
更多
心理课堂
vConsole
首页
测评
干货
社区
我的

心理测试：


11:05
心理测评
搜索
我的
心理
档案
点击查看
为你推荐
test5
健康
情感
人际
能力
性格
发展
国际标准抑郁测试
vConsole
首页
测评
干货
社区
我的

第四部分：附录

附上建立数据表的所有SQL语句：

```
-- 不需要主键自增
CREATE TABLE `USER` (
`openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`nickName` varchar(100) CHARACTER SET utf8mb4 NULL DEFAULT '',
`avatarUrl` varchar(300) CHARACTER SET utf8 NULL DEFAULT '',
`gender` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`province` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`city` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
```

```
`grade` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Register_Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,
`points` int unsigned NULL DEFAULT 0,
`use_days` int unsigned NULL DEFAULT 0,
PRIMARY KEY (`openid`)
);
-- AUTO_INCREMENT = 1;

/* 自动更新天数 */
drop EVENT IF EXISTS update_event;
CREATE EVENT update_event
    ON SCHEDULE EVERY 1 DAY STARTS CURDATE()
    ON COMPLETION PRESERVE ENABLE
    DO update USER set use_days=use_days+1;


CREATE TABLE `INFO` (
`openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`points` int unsigned NULL DEFAULT 0,
`use_days` int unsigned NULL DEFAULT 0,
`last_test_time` datetime NULL DEFAULT NULL,
PRIMARY KEY(`openid`)
);
ALTER TABLE `INFO` ADD CONSTRAINT `fk_INFO_USER_1` FOREIGN KEY(`openid`)
REFERENCES `USER`(`openid`) ON DELETE CASCADE ON UPDATE CASCADE;



-- 要主键自增
CREATE TABLE `ARTICLE` (
`Article_ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`Title` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Author` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Issuer_ID` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Issue_time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,
`Context` text NULL,
`Watches` int unsigned NULL DEFAULT 0,
`Likes` int unsigned NULL DEFAULT 0,
`Transmit` int unsigned NULL DEFAULT 0,
`Back_Image` varchar(300) CHARACTER SET utf8 NULL DEFAULT '',
`Tag` varchar(100) CHARACTER SET utf8 NULL DEFAULT ''
)
AUTO_INCREMENT = 1;


-- 要主键自增
CREATE TABLE `COMMON_ARTICLE` (
```

```
`ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`Article_ID` int unsigned NOT NULL,
`Commoner_ID` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Content` text NULL,
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
`Likes` int unsigned NULL DEFAULT 0
)
AUTO_INCREMENT = 1;


-- 要主键自增
CREATE TABLE `COMMON_ARTICLE2` (
`ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`Common1_ID` int unsigned NOT NULL,
`Commoner_ID` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Context` text NULL,
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
AUTO_INCREMENT = 1;


CREATE TABLE `FAVOR` (
-- `ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Tag` varchar(100) CHARACTER SET utf8 not NULL DEFAULT '',
`ID` int unsigned NOT NULL,
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY(`openid`, `Tag`, `ID`)
);
ALTER TABLE `FAVOR` ADD CONSTRAINT `fk_FAVOR_USER_1` FOREIGN KEY (`openid`)
REFERENCES `USER` (`openid`) ON DELETE CASCADE ON UPDATE CASCADE;


-- 要主键自增
CREATE TABLE `COLLECT` (
-- `ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Tag` varchar(100) CHARACTER SET utf8 not NULL DEFAULT '',
`ID` int unsigned NOT NULL,
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY(`openid`, `Tag`, `ID`)
);
ALTER TABLE `COLLECT` ADD CONSTRAINT `fk_COLLECT_USER_1` FOREIGN KEY
(`openid`) REFERENCES `USER` (`openid`) ON DELETE CASCADE ON UPDATE CASCADE;


-- 不需要主键自增
CREATE TABLE `RE_ARTICLE` (
    `openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
```

```
    `Article_ID` int unsigned,
    `Music_ID` int unsigned,
    PRIMARY KEY (`openid`)
);


CREATE TABLE `ROBOTQA` (
`ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`Key_Word` varchar(300) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Question` text NULL,
`Answer1` text NULL,
`If_RE` int NOT NULL DEFAULT 0
)
AUTO_INCREMENT = 1;


CREATE TABLE `COMMUNITY` (
`ID` int unsigned NOT NULL PRIMARY Key AUTO_INCREMENT,
`openid` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Noname` tinyint(1) DEFAULT 0,
`Area` varchar(100) CHARACTER SET utf8 NULL DEFAULT '',
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
`Images` text,
`Content` text NULL,
`Watches` int unsigned NULL DEFAULT 0,
`Likes` int unsigned NULL DEFAULT 0,
`Collections` int unsigned NULL DEFAULT 0,
`comment_number` int unsigned NULL DEFAULT 0,
`Forward` int unsigned NULL DEFAULT 0,
`Tag` varchar(100) CHARACTER SET utf8 NULL DEFAULT ''
)
AUTO_INCREMENT = 1;
ALTER TABLE `COMMUNITY` ADD CONSTRAINT `fk_COMMUNITY_USER_1` FOREIGN
Key(`openid`) REFERENCES `USER`(`openid`) ON DELETE CASCADE ON UPDATE CASCADE;



CREATE TABLE `COMMON_COMMUNITY` (
`ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`Community_ID` int unsigned NOT NULL,
`Commentator_ID` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Noname` tinyint(1) DEFAULT 0,
`Content` text NULL,
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
`Likes` int unsigned NULL DEFAULT 0
)
AUTO_INCREMENT = 1;
```

```
ALTER TABLE `COMMON_COMMUNITY` ADD CONSTRAINT `fk_COMMON_COMMUNITY_USER_1`
FOREIGN KEY(`Commentator_ID`) REFERENCES `USER`(`openid`) ON DELETE CASCADE ON
UPDATE CASCADE;
ALTER TABLE `COMMON_COMMUNITY` ADD CONSTRAINT
`fk_COMMON_COMMUNITY_COMMUNITY_1` FOREIGN KEY(`Community_ID`) REFERENCES
`COMMUNITY`(`ID`) ON DELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE `COMMON_COMMUNITY2` (
`ID` int unsigned NOT NULL PRIMARY KEY AUTO_INCREMENT,
`Comment_ID` int unsigned NOT NULL,
`Commentator_ID` varchar(100) CHARACTER SET utf8 NOT NULL DEFAULT '',
`Noname` tinyint(1) DEFAULT 0,
`Content` text NULL,
`Time` datetime NULL ON UPDATE CURRENT_TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
AUTO_INCREMENT = 1;

ALTER TABLE `COMMON_COMMUNITY2` ADD CONSTRAINT `fk_COMMON_COMMUNITY2_USER_1`
FOREIGN KEY(`Commentator_ID`) REFERENCES `USER`(`openid`) ON DELETE CASCADE ON
UPDATE CASCADE;

ALTER TABLE `COMMON_COMMUNITY2` ADD CONSTRAINT
`fk_COMMON_COMMUNITY2_COMMUNITY_1` FOREIGN KEY(`Comment_ID`) REFERENCES
`COMMON_COMMUNITY`(`ID`) ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `ARTICLE` ADD CONSTRAINT `fk_ARTICLE_USER_1` FOREIGN KEY
(`Issuer_ID`) REFERENCES `USER` (`openid`) ON DELETE CASCADE ON UPDATE
CASCADE;


ALTER TABLE `HISTORY` ADD CONSTRAINT `fk_HISTORY_USER_1` FOREIGN KEY
(`openid`) REFERENCES `USER` (`openid`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `RE_ARTICLE` ADD CONSTRAINT `fk_RE_ARTICLE_USER_1` FOREIGN KEY
(`openid`) REFERENCES `USER` (`openid`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `COMMON_ARTICLE` ADD CONSTRAINT `fk_COMMON_ARTICLE_USER_1` FOREIGN
KEY (`Commoner_ID`) REFERENCES `USER` (`openid`) ON DELETE CASCADE ON UPDATE
CASCADE;

ALTER TABLE `COMMON_ARTICLE` ADD CONSTRAINT `fk_COMMON_ARTICLE_ARTICLE_1`
FOREIGN KEY (`Article_ID`) REFERENCES `ARTICLE` (`Article_ID`) ON DELETE
CASCADE ON UPDATE CASCADE;

ALTER TABLE `COMMON_ARTICLE2` ADD CONSTRAINT
`fk_COMMON_ARTICLE2_COMMON_ARTICLE_1` FOREIGN KEY (`Common1_ID`) REFERENCES
`COMMON_ARTICLE` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE `COMMON_ARTICLE2` ADD CONSTRAINT `fk_COMMON_ARTICLE2_USER_1`
FOREIGN KEY (`Commoner_ID`) REFERENCES `USER` (`openid`) ON DELETE CASCADE ON
UPDATE CASCADE;
```

model.py文件内容

```
# coding: utf-8
from sqlalchemy import Column, DateTime, ForeignKey, Text, text
from sqlalchemy.dialects.mysql import INTEGER, TINYINT, VARCHAR
from sqlalchemy.orm import relationship
from sqlalchemy.ext.declarative import declarative_base

from app import DB_light


class Collect(DB_light.Model):
    __tablename__ = 'collect'

    openid = Column(VARCHAR(100), primary_key=True, nullable=False,
server_default=text("''"))
    Tag = Column(VARCHAR(100), primary_key=True, nullable=False,
server_default=text("''"))
    ID = Column(INTEGER(10), primary_key=True, nullable=False)
    Time = Column(DateTime, server_default=text("CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP"))


class Favor(DB_light.Model):
    __tablename__ = 'favor'

    openid = Column(VARCHAR(100), primary_key=True, nullable=False,
server_default=text("''"))
    Tag = Column(VARCHAR(100), primary_key=True, nullable=False,
server_default=text("''"))
    ID = Column(INTEGER(10), primary_key=True, nullable=False)
    Time = Column(DateTime, server_default=text("CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP"))


class Greeting(DB_light.Model):
    __tablename__ = 'greetings'

    ID = Column(INTEGER(10), primary_key=True)
    Context = Column(Text)


class Music(DB_light.Model):
```

```
    __tablename__ = 'music'

    Music_ID = Column(INTEGER(10), primary_key=True)
    Name = Column(VARCHAR(100), server_default=text("''"))
    Singer = Column(VARCHAR(50), server_default=text("''"))
    Type = Column(VARCHAR(20), server_default=text("''"))


class Robotqa(DB_light.Model):
    __tablename__ = 'robotqa'

    ID = Column(INTEGER(10), primary_key=True)
    Key_Word = Column(VARCHAR(300), nullable=False, server_default=text("''"))
    Question = Column(Text)
    Answer1 = Column(Text)
    If_RE = Column(INTEGER(11), nullable=False, server_default=text("'0'"))


class User(DB_light.Model):
    __tablename__ = 'user'

    openid = Column(VARCHAR(100), primary_key=True, server_default=text("''"))
    nickName = Column(VARCHAR(100), server_default=text("''"))
    avatarUrl = Column(VARCHAR(300), server_default=text("''"))
    gender = Column(VARCHAR(100), server_default=text("''"))
    province = Column(VARCHAR(100), server_default=text("''"))
    city = Column(VARCHAR(100), server_default=text("''"))
    grade = Column(VARCHAR(100), server_default=text("''"))
    Register_Time = Column(DateTime, server_default=text("CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP"))
    points = Column(INTEGER(10), server_default=text("'0'"))
    use_days = Column(INTEGER(10), server_default=text("'0'"))


class ReArticle(User):
    __tablename__ = 're_article'

    openid = Column(ForeignKey('user.openid', ondelete='CASCADE',
onupdate='CASCADE'), primary_key=True, server_default=text("''"))
    Article_ID = Column(INTEGER(10))
    Music_ID = Column(INTEGER(10))


class Article(DB_light.Model):
    __tablename__ = 'article'

    Article_ID = Column(INTEGER(10), primary_key=True)
    Title = Column(VARCHAR(100), server_default=text("''"))
    Author = Column(VARCHAR(100), server_default=text("''"))
```

```
    Issuer_ID = Column(ForeignKey('user.openid', ondelete='CASCADE',
onupdate='CASCADE'), index=True, server_default=text("''"))
    Issue_time = Column(DateTime, server_default=text("CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP"))
    Context = Column(Text)
    Watches = Column(INTEGER(10), server_default=text("'0'"))
    Likes = Column(INTEGER(10), server_default=text("'0'"))
    Collections = Column(INTEGER(10), server_default=text("'0'"))
    Forward = Column(INTEGER(10), server_default=text("'0'"))
    Back_Image = Column(VARCHAR(300), server_default=text("''"))
    Tag = Column(VARCHAR(100), server_default=text("''"))

    user = relationship('User')


class Community(DB_light.Model):
    __tablename__ = 'community'

    ID = Column(INTEGER(10), primary_key=True)
    openid = Column(ForeignKey('user.openid', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True, server_default=text("''"))
    Noname = Column(TINYINT(1), server_default=text("'0'"))
    Area = Column(VARCHAR(100), server_default=text("''"))
    Time = Column(DateTime, server_default=text("CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP"))
    Images = Column(Text)
    Content = Column(Text)
    Watches = Column(INTEGER(10), server_default=text("'0'"))
    Likes = Column(INTEGER(10), server_default=text("'0'"))
    Collections = Column(INTEGER(10), server_default=text("'0'"))
    comment_number = Column(INTEGER(10), server_default=text("'0'"))
    Forward = Column(INTEGER(10), server_default=text("'0'"))
    Tag = Column(VARCHAR(100), server_default=text("''"))

    user = relationship('User')


class History(DB_light.Model):
    __tablename__ = 'history'

    ID = Column(INTEGER(10), primary_key=True)
    openid = Column(ForeignKey('user.openid', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True, server_default=text("''"))
    Tag = Column(VARCHAR(50), server_default=text("''"))
    Tag_ID = Column(INTEGER(10), nullable=False)
    Description = Column(VARCHAR(100), server_default=text("''"))

    user = relationship('User')
```

```
class CommentCommunity(DB_light.Model):
    __tablename__ = 'comment_community'

    ID = Column(INTEGER(10), primary_key=True)
    Community_ID = Column(ForeignKey('community.ID', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True)
    Commentator_ID = Column(ForeignKey('user.openid', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True, server_default=text("''"))
    Noname = Column(TINYINT(1), server_default=text("'0'"))
    Content = Column(Text)
    Time = Column(DateTime, server_default=text("CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP"))
    Likes = Column(INTEGER(10), server_default=text("'0'"))

    user = relationship('User')
    community = relationship('Community')


class CommonArticle(DB_light.Model):
    __tablename__ = 'common_article'

    ID = Column(INTEGER(10), primary_key=True)
    Article_ID = Column(ForeignKey('article.Article_ID', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True)
    Commoner_ID = Column(ForeignKey('user.openid', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True, server_default=text("''"))
    Content = Column(Text)
    Time = Column(DateTime, server_default=text("CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP"))
    Likes = Column(INTEGER(10), server_default=text("'0'"))

    article = relationship('Article')
    user = relationship('User')


class CommentCommunity2(DB_light.Model):
    __tablename__ = 'comment_community2'

    ID = Column(INTEGER(10), primary_key=True)
    Comment_ID = Column(ForeignKey('comment_community.ID', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True)
    Commentator_ID = Column(ForeignKey('user.openid', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True, server_default=text("''"))
    Noname = Column(TINYINT(1), server_default=text("'0'"))
    Content = Column(Text)
    Time = Column(DateTime, server_default=text("CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP"))
```

```
    comment_community = relationship('CommentCommunity')
    user = relationship('User')


class CommonArticle2(DB_light.Model):
    __tablename__ = 'common_article2'

    ID = Column(INTEGER(10), primary_key=True)
    Common1_ID = Column(ForeignKey('common_article.ID', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True)
    Commoner_ID = Column(ForeignKey('user.openid', ondelete='CASCADE',
onupdate='CASCADE'), nullable=False, index=True, server_default=text("''"))
    Context = Column(Text)
    Time = Column(DateTime, server_default=text("CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP"))
    Likes = Column(INTEGER(10), server_default=text("'0'"))

    common_article = relationship('CommonArticle')
    user = relationship('User')
```