



03 项目实施文档

《互联网数据库开发》课程团队作业

小组成员：姜宇，唐明昊，王禹衡，徐海濛

学号：2210705、2113927、2213040、2212180



1 前期讨论与分工安排:

本次作业为团队合作的作业，我们组建了群聊，用来进行任务的分配和交流。

如下所示为我们进行分工讨论的一些记录：

我们这周配好环境，下周开始正式的工作吧，看了看项目可以分成下面四个部分的工作：

1. 数据库设计和后端相关接口实现，后端开发
2. 前端页面优化，前端开发
3. 添加功能，对接前后端，全栈开发
4. 数据收集后期报告相关撰写以及视频录制

也不一定按照这个做，大家有意见随时说我们可以合理调度一下🐱

我现在在做3功能添加，也就是现在仓库里那个的大模型接入

我已经改了两个地方，我在后端接口处添加了说明，很多地方需要用get传参调用接口，你一看应该就能懂，我写好注释了

第一，视频无法显示的问题，你按照我的方法修改，@一把野菜（挖香菜版 小王需要更新数据库中的url为后端资源的url，不要直接在SQL上改，访问8000端口哭的后台改

第二，评论无法显示的问题，我同样也是改了前端的调用，@Tony 你学习这两个是怎么调用的改好其他的f12报错以及非法的成员不显示问题。

我添加了学生数据表和相关接口()，只需要传入id就可以返回组员信息

更新了视频，这下应该都可以看到了

正在尝试添加一点SQL












语句

防止因为外键依赖

导致没法正常插表

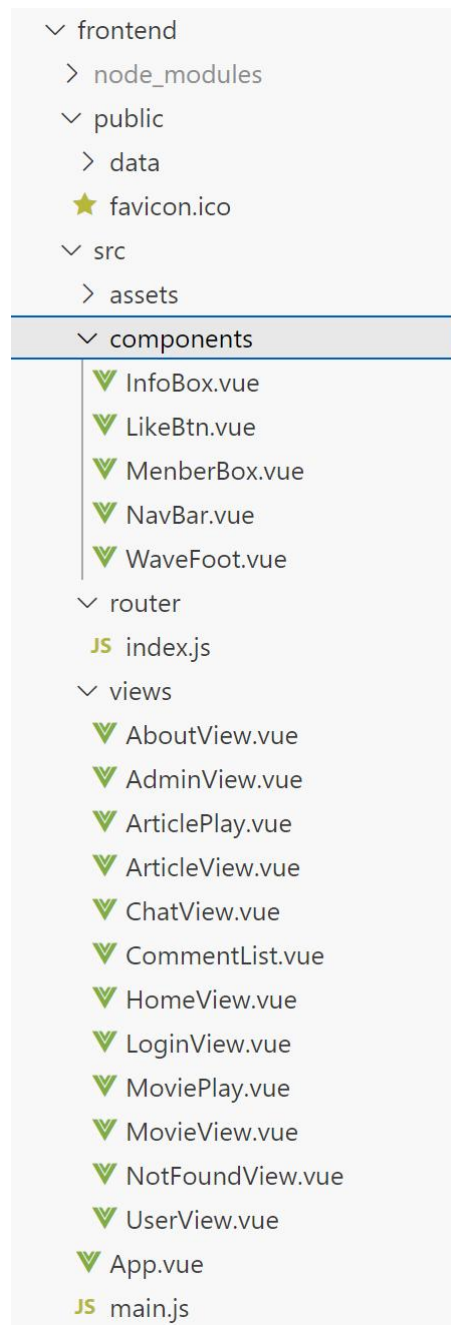
2 项目完成过程记录:

为了便于项目代码的同步过程, 我们选择了 `github` 来进行代码的托管服务:

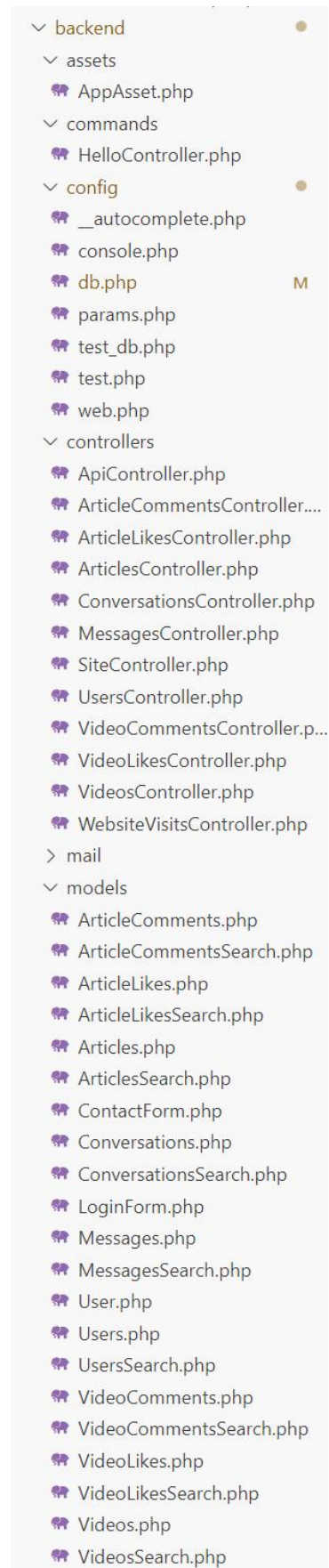
	<div><div>update README</div><div> mh-tang committed last week</div></div>
	<div><div>update all(backend, data, frontend)</div><div> YoreJ committed last week</div></div>
	<div><div>update all(backend, data, frontend)</div><div> YoreJ committed last week</div></div>
-O-	Commits on Dec 15, 2024
	<div><div>frontend</div><div> cyborvirtue committed last week</div></div>
-O-	Commits on Dec 14, 2024
	<div><div>更新SQL文件</div><div> varyhoo committed last week</div></div>
-O-	Commits on Dec 13, 2024
	<div><div>update backend</div><div> YoreJ committed last week</div></div>
	<div><div>update backend</div><div> YoreJ committed last week</div></div>
-O-	Commits on Dec 10, 2024
	<div><div>数据库实现02</div><div> varyhoo committed 2 weeks ago</div></div>
	<div><div>部分完成数据库设计</div><div> varyhoo committed 2 weeks ago</div></div>
-O-	Commits on Dec 9, 2024
	<div><div>update login post/get request</div><div> YoreJ committed 2 weeks ago</div></div>
-O-	Commits on Dec 6, 2024
	<div><div>Initial commit</div><div> YoreJ committed 2 weeks ago</div></div>

3 项目目录展示:

3.1 前端项目目录展示:



3.2 后端项目目录展示:



4 前端代码实现:

在前端代码实现中,我们主要使用了 `vue` 进行页面的编写。我们为每个功能的实现都编写了一个页面,例如: `ArticleView.vue` 用来查看所有文章的标题, `ArticlePlay.vue` 用于查看文章的正文。下面是部分代码截图:

实现对于点赞量的获取:

```
getLikeNum() {
  const id = this.$route.params.id;
  axios
    .get('http://localhost:8080/api/likenumarticle?articleId=' + id)
    .then((response) => {
      this.likeNum = response.data.likeCount;
    })
    .catch((error) => {
      console.error('请求数据失败', error);
    });
},

likeOr() {
  const id = this.$route.params.id;
  const userid = sessionStorage.getItem('UserID');
  axios
    .get('http://localhost:8080/api/getlikearticle?userId=' + userid + '&articleId=' + id)
    .then((response) => {
      this.like = response.data.liked;
    })
    .catch((error) => {
      console.error('请求数据失败', error);
    });
},
```

前端代码是用户进行交互的主要部分,用户可以通过前端的按钮点击,评论的输入和发送,来实现相关功能的交互,所以,我们需要前端代码实现的就是,能够在接收到用户的请求之后,通过异步的处理逻辑,支持对应的请求,并为用户提供相应的服务。

我们以获取点赞量为例子,讲解前端代码的实现逻辑:

`getLikeNum` 方法通过发送 HTTP GET 请求,从后端 API 获取指定文章的点赞数量,并将其存储在组件的 `likeNum` 属性中。

(1) 首先,我们从当前的路由参数中提取出对应的文章 `id`,而后使用 `axios` 库发送一个 GET 请求,将获取到的文章 `id` 和对应的查询 API 都传送给后端。

(2) `.then((response) =>):` 在后端 API 的请求成功之后,会从响应的对象 `response` 中提取出对应的 `data.likeCount`,即点赞量;

(3) 而后将提取到的点赞数量赋值给组件的 `likenum` 属性,以便在模板中展示或进行其他逻辑处理。

(4) 如果请求失败，那么会调用 `catch(error)` 的情况，向用户展示错误的提示，使用弹窗通知等。

5 后端代码实现：

4.1 数据库实现：

由于我们需要制作的网站的性质是一个有关人工智能的宣传的网站。因此，我们需要展示一些基础的人工智能项目介绍，为了能够保存每个用户的记录，我们在数据库中，创建了例如：成员表，新闻表，视频表等多个基础的表。同时考虑到增加网页功能的丰富性，我们还创建了例如评论表，点赞表等多种数据统计的表格。由于我们的网站中还实现了有关大模型的聊天对话功能，我们添加了对话表和历史记录保存表等多种表格来保存用户的历史聊天记录。(数据表的SQL代码实现详见项目设计文档)

4.2 后端 API 的实现：

(1) 获取前端页面信息修改后端数据库的所有 API 的展示：

从之前的分析，我们不难看出，我们需要根据前端的交互功能来实现对于后端数据库中数据的实时更新。因此，我们需要在后端添加一些 API 来统计和修改数据库信息的变化，如下所示为我们在本次实验中所使用到的所有的功能 API。


```

'db' => $db,
'urlManager' => [
  'enablePrettyUrl' => true,
  'showScriptName' => false,
  'rules' => [
    'api/addwebviews' => 'api/addwebviews', // finish 添加网站访问量
    'api/getwebviews' => 'api/getwebviews', // finish 获取网站访问量
    'api/login' => 'api/login', // finish 登录
    'api/signup' => 'api/signup', // finish 注册

    'api/getuser' => 'api/getuser', // finish 获取用户信息
    'api/chat' => 'api/chatbot', // finish 聊天机器人

    'api/getarticle' => 'api/getarticle', // finish 获取文章
    'api/addarticle' => 'api/addarticle', // finish 添加文章
    'api/deletearticle' => 'api/deletearticle', // finish 删除文章
    'api/viewarticle' => 'api/viewarticle', // finish 增加文章访问量
    'api/likearticle' => 'api/likearticle', // finish 喜欢文章
    'api/likenumarticle' => 'api/likenumarticle', // finish 获取文章喜欢数
    'api/getlikearticle' => 'api/getlikearticle', // finish 获取是否喜欢文章
    'api/commentarticle' => 'api/commentarticle', // finish 评论文章
    'api/showcommentarticle' => 'api/showcommentarticle', //finish 显示文章评论
    'api/deletecommentarticle' => 'api/deletecommentarticle', //finish 删除文章评论
    'api/getarticlepagecount' => 'api/getarticlepagecount', //finish 获取文章页数

    'api/getvideo' => 'api/getvideo', // finish 获取视频
    'api/addvideo' => 'api/addvideo', // finish 添加视频
    'api/deletevideo' => 'api/deletevideo', // finish 删除视频
    'api/viewvideo' => 'api/viewvideo', // finish 增加视频访问量
    'api/likevideo' => 'api/likevideo', // finish 喜欢视频
    'api/likenumvideo' => 'api/likenumvideo', // finish 获取视频喜欢数
    'api/getlikevideo' => 'api/getlikevideo', // finish 获取是否喜欢视频
    'api/commentvideo' => 'api/commentvideo', // finish 评论视频
    'api/showcommentvideo' => 'api/showcommentvideo', //finish 显示视频评论
    'api/deletecommentvideo' => 'api/deletecommentvideo', //finish 删除视频评论
    'api/getvideopagecount' => 'api/getvideopagecount', //finish 获取视频页数
  ],
],

```

接下来，我们选取几个典型的功能 API 来进行介绍：

(2) 网页访问量的更新和获取功能 API:

I. 网页访问量的获取:

总体来说，前端通过 Vue.js 提供的 axios 发送 HTTP 请求到后端 API。首先，该方法通过 axios.get 发送一个 GET 请求到后端接口 8080/api/getwebviews，请求当前的网站访问数据量。

如果请求成功时，后端返回的数据包含访问量，此时前端将该值赋给 views 数据属性，从而显示页面的访问量数据。

```
/**
 * Increments the visit count.
 *
 * @return boolean whether the increment was successful
 */
public function incrementVisitCount()
{
    $websiteVisit = WebsiteVisits::find()->one();
    if ($websiteVisit === null) {
        $websiteVisit = new WebsiteVisits();
        $websiteVisit->VisitCount = 1;
    } else {
        $websiteVisit->VisitCount += 1;
    }
    return $websiteVisit->save();
}

/**
 * Gets the current visit count.
 *
 * @return int the current visit count
 */
public function getVisitCount()
{
    $websiteVisit = WebsiteVisits::find()->one();
    return $websiteVisit ? $websiteVisit->VisitCount : 0;
}
```

II. 网页访问量的更新:

网页访问量的更新也是相同的流程，我们在这里主要介绍函数的功能:

首先，使用 `$websiteVisit = WebsiteVisits::find()->one();` 语句查询数据库，该语句会查找数据库中是否已有 `WebsiteVisits` 表的记录。判断是否存在访问记录。如果存在访问记录，那么就将当前网页的访问量加 1，否则就将创建一条新的记录，记录网页的访问量，最后调用 `save()` 函数，来将修改保存到

本地的数据库中。

(3) 获取新闻资源功能的 API:

获取新闻信息功能对应的功能 API 为:

```
'api/getarticle' => 'api/getarticle'
```

```
public function attributeLabels()
{
    return [
        'CommentID' => 'Comment ID',
        'UserID' => 'User ID',
        'ArticleID' => 'Article ID',
        'Content' => 'Content',
        'CommentedAt' => 'Commented At',
    ];
}

/**
 * Gets query for [[Article]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getArticle()
{
    return $this->hasOne(Articles::class, ['ArticleID' => 'ArticleID']);
}

/**
 * Gets query for [[User]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getUser()
{
    return $this->hasOne(Users::class, ['UserID' => 'UserID']);
}
```

在前端发起获取文章的请求后，axios 会向后端发送对应的请求，而后 Yii2 后端在收到 getArticle 的请求之后。会调用 getArticle()来将对应的数据并将处理好的数据传递给前端，进行展示。