知识星球

本项目是基于PHP的一个论坛管理系统与平台。

系 Shanghai Normal University 2021级 PHP程序设计课程的期末大作业。

# 项目简介

本项目是一个基于PHP的知识类论坛管理系统，用户可以在本网站发布文章，浏览文章，参与文章的编辑（包括别人的！），并且可以使用TAG系统方便地查找自己感兴趣的文章。

## 网站基本功能简介

## 主要功能模块

### 用户模块

本模块包含了管理用户的若干功能：

* 注册
* 登录
* 个人信息
  + 查看个人信息
  + 修改个人信息
* 修改密码（重置密码）
* 退出登录

### 文章模块

本模块包含了管理文章的若干功能，按照功能可以分为：

* 浏览文章
* 创建文章
* 修改文章

#### 浏览文章

浏览功能主要分为按照更新时间排序的文章列表，按照TAG分类的文章列表，以及搜索功能。

1. 对于使用更新时间排序的文章列表

* 用户可以选择合适的分页逻辑，查看文章的标题，作者，创建以及更新日期，TAG等。

1. 对于使用搜索功能获得的文章列表

* 在上述功能的基础上，用户可以输入关键词，查找包含关键词的文章。

1. 对于使用TAG分类的文章列表

* 在功能1的基础上，用户可以选择合适的TAG，查看包含该TAG的文章。
* 此处的TAG选项由系统自动生成，是按照热度（包含该TAG的文章数量）倒序给出的。
* 用户可以自由选择若干TAG，并可一键切换筛选TAG的逻辑：
  + 全包含 筛选出包含所有选择的TAG的文章
  + 任意包含 筛选出包含任意一个选择的TAG的文章

#### 创建文章

对于所有登录用户，均可以创建文章，创建文章时需要填写标题，内容，并在创建成功后自动成为文章的第一位作者。

#### 修改文章

对于所有管理员用户，均可以修改文章，修改文章主要包括：

* 对文章内容的修改 可修改标题，内容。并在修改成功后自动成为文章的作者之一。
* 对文章TAG的修改 可修改文章的TAG，包括增加或者删除TAG以及为该文章新建一个TAG，对TAG的修改不会影响文章的作者。
* 对文章的删除 可删除文章，删除后文章将不再显示在文章列表中。

## 数据库设计

### 数据库实体

1. Users（用户）:
   * **user\_id: 主键，自增整数，唯一标识一个用户。**
   * username: 用户名，唯一且不包含逗号，用于用户登录。
   * password: 用户的密码，存储加密形式，保障账户安全。
   * email: 用户的电子邮件地址，用于联系和账户恢复。
   * is\_admin: 布尔值，标记是否为管理员，用于控制权限。
   * created\_at: 时间戳，记录用户账户的创建时间。
2. Articles（文章）:
   * **article\_id: 主键，自增整数，唯一标识一篇文章。**
   * title: 文章标题。
   * content: 文章内容，以长文本形式存储。
   * created\_at: 时间戳，文章的创建时间，自动创建。
   * updated\_at: 时间戳，文章的最后更新时间，自动更新。
3. Tags（标签）:
   * **tag\_id: 主键，自增整数，唯一标识一个标签。**
   * tag\_name: 标签名称，唯一且不包含逗号。
4. Article\_Tags（文章标签关系）:
   * **article\_id 和 tag\_id: 复合主键，标识文章和标签的关系。** 外键关系指向 Articles 和 Tags 表，支持级联删除。
5. Article\_Authors（文章作者关系）:
   * **article\_id 和 user\_id: 复合主键，标识文章和作者的关系。** 外键关系指向 Articles 和 Users 表，支持级联删除。

### 数据库关系

多对多关系:

* Articles 和 Tags: 通过 Article\_Tags 表实现多对多关系，一篇文章可以有多个标签，一个标签可以标注多篇文章。
* Articles 和 Users: 通过 Article\_Authors 表实现多对多关系，一篇文章可以有多个共同作者。

### 数据库层API

视图:

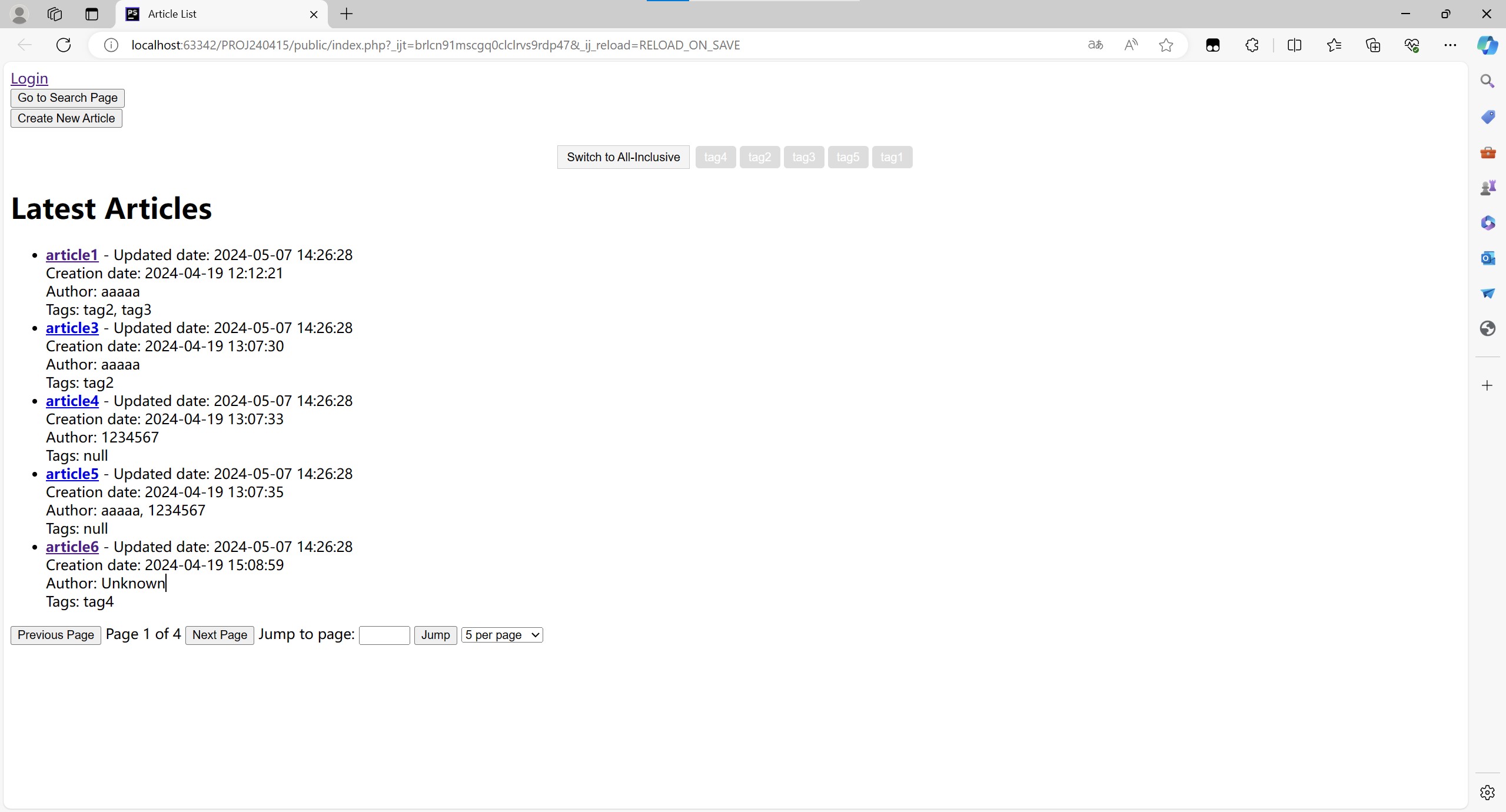
* article\_tags\_view: 聚合视图，显示每篇文章及其所有标签。
* article\_authors\_view: 聚合视图，显示每篇文章及其所有作者。
* article\_detail\_view: 综合视图，显示文章详细信息，包括标题、作者、创建时间、更新时间、标签和内容。
* article\_list\_view: 文章列表视图，显示文章的基本信息，支持排序和分页。
* article\_count\_view: 文章计数视图，显示总的文章数量。
* tag\_article\_count\_view：标签文章计数视图，显示每个标签下的文章数量。

存储过程:

* FetchArticlesByPage: 分页获取文章列表，支持按更新时间排序。
* FetchArticlesByTagsWithPaging: 根据标签（允许多个）过滤文章，并支持分页功能。允许部分标签匹配。
* FetchArticlesByAllTagsWithPaging: 根据标签（允许多个）过滤文章，并支持分页功能。要求完全匹配给定的标签。
* CreateArticle: 创建新文章，并将其与作者关联。
* UpdateArticle: 更新指定文章的内容和标题，并管理文章作者信息。当发生错误时主动返回错误信息。
* DeleteArticle: 根据文章ID删除文章。当发生错误时主动返回错误信息。
* UpdateArticleTags: 更新指定文章的标签，包括添加和删除标签（允许多个）。
* SearchArticlesWithPaging: 根据关键词搜索文章，并支持分页功能。

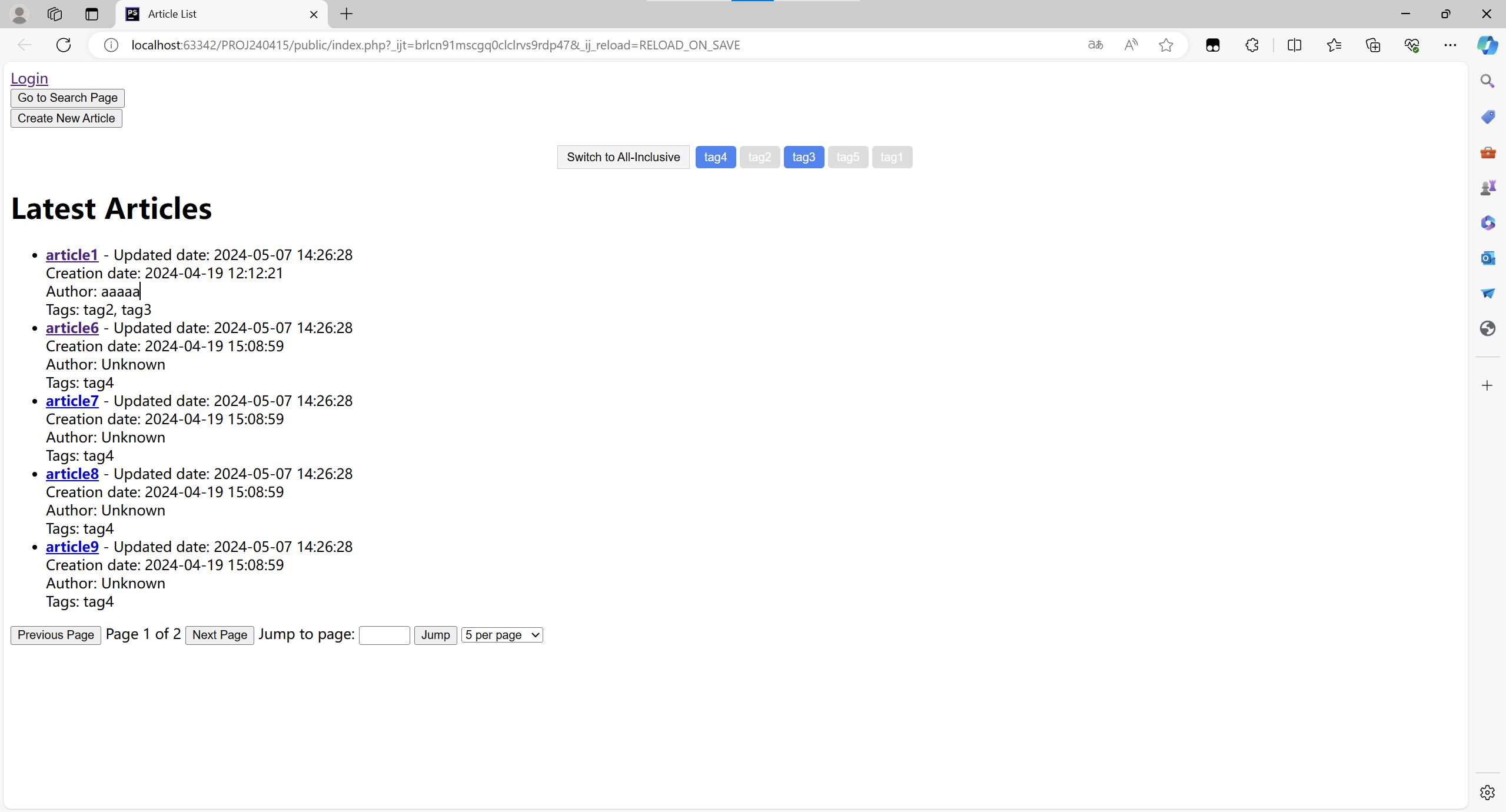
## 网站各功能截图

### 首页

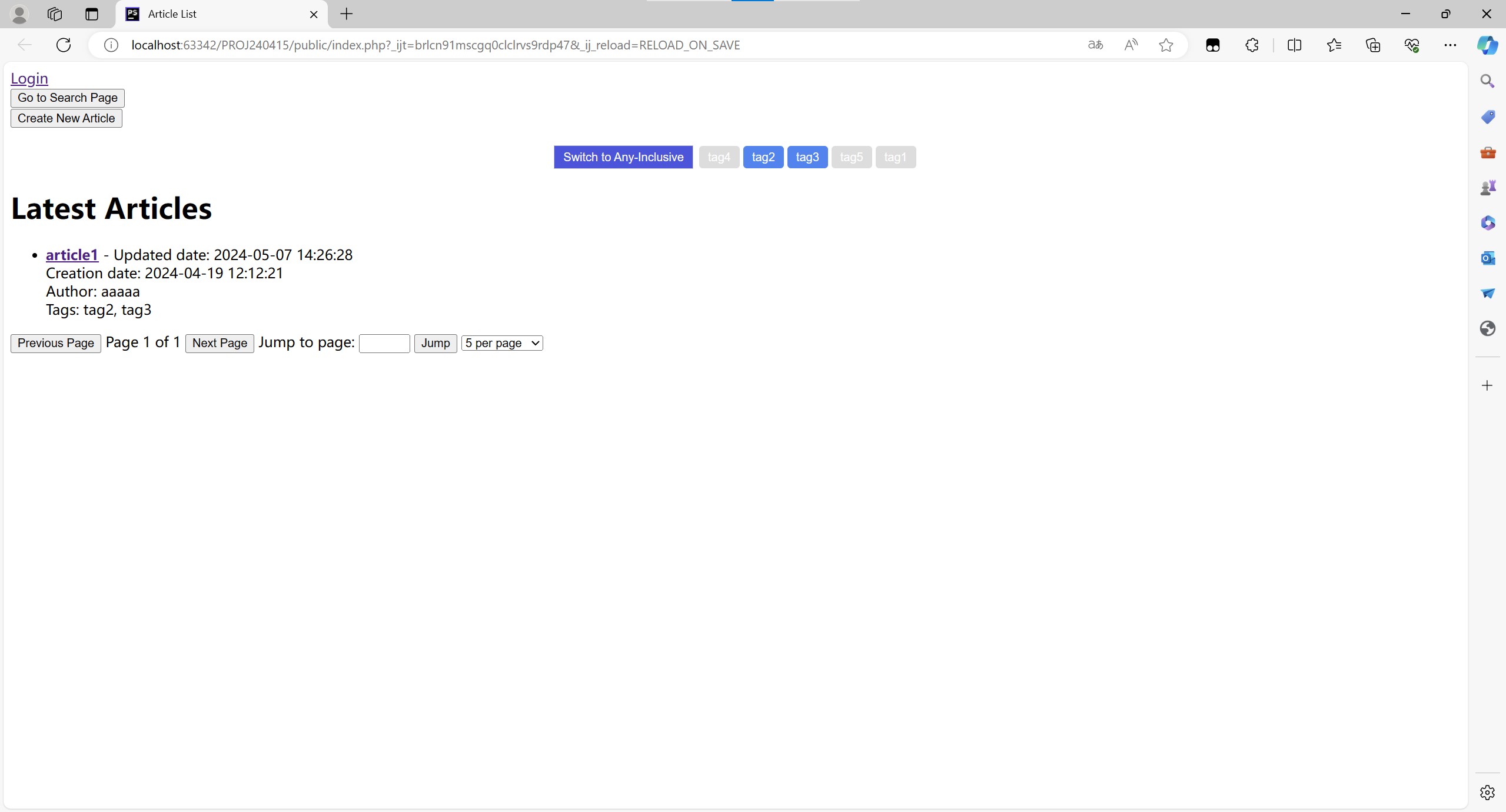


index

### 按照TAG浏览

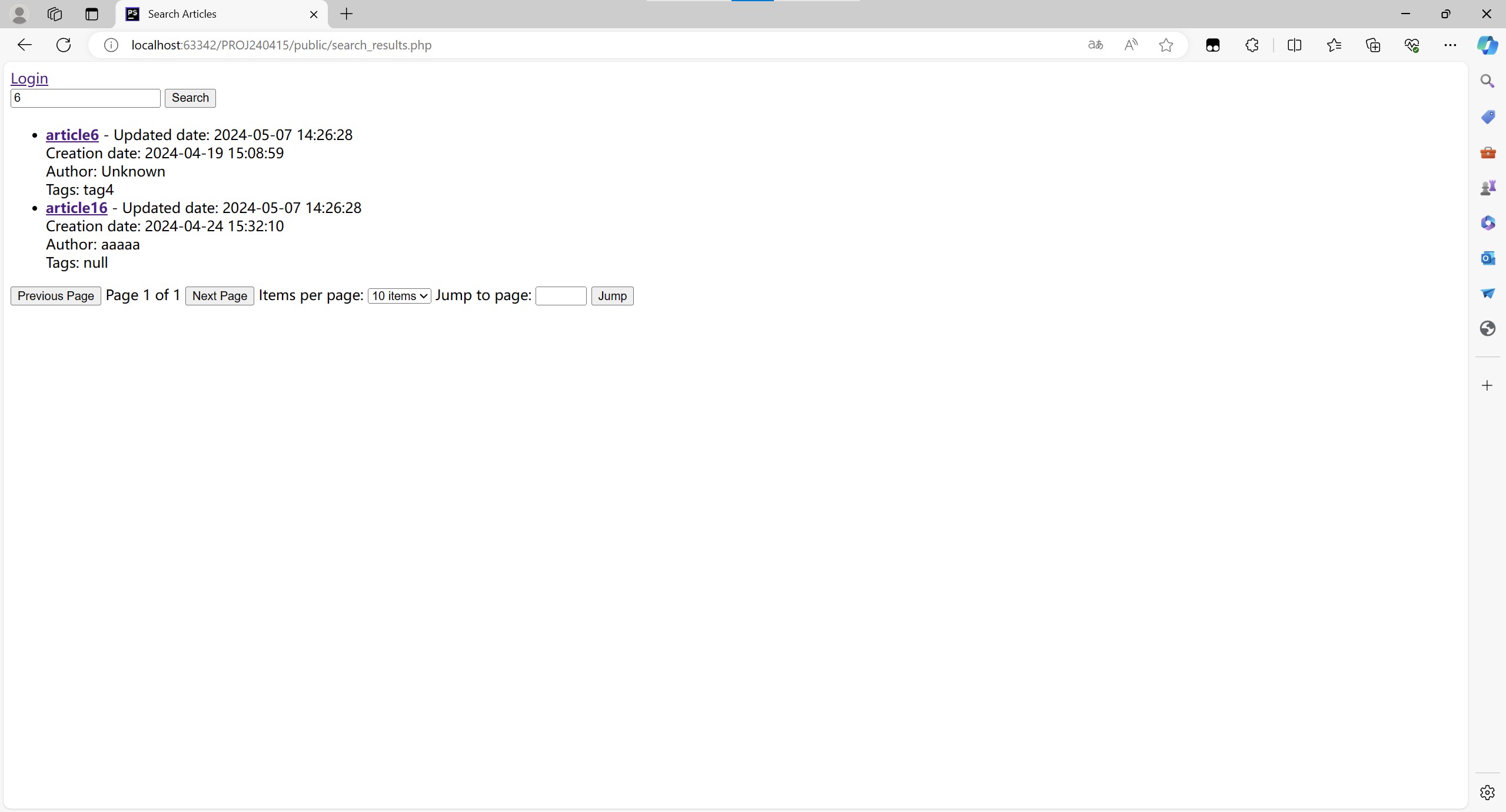


index\_tag1



index\_tag2

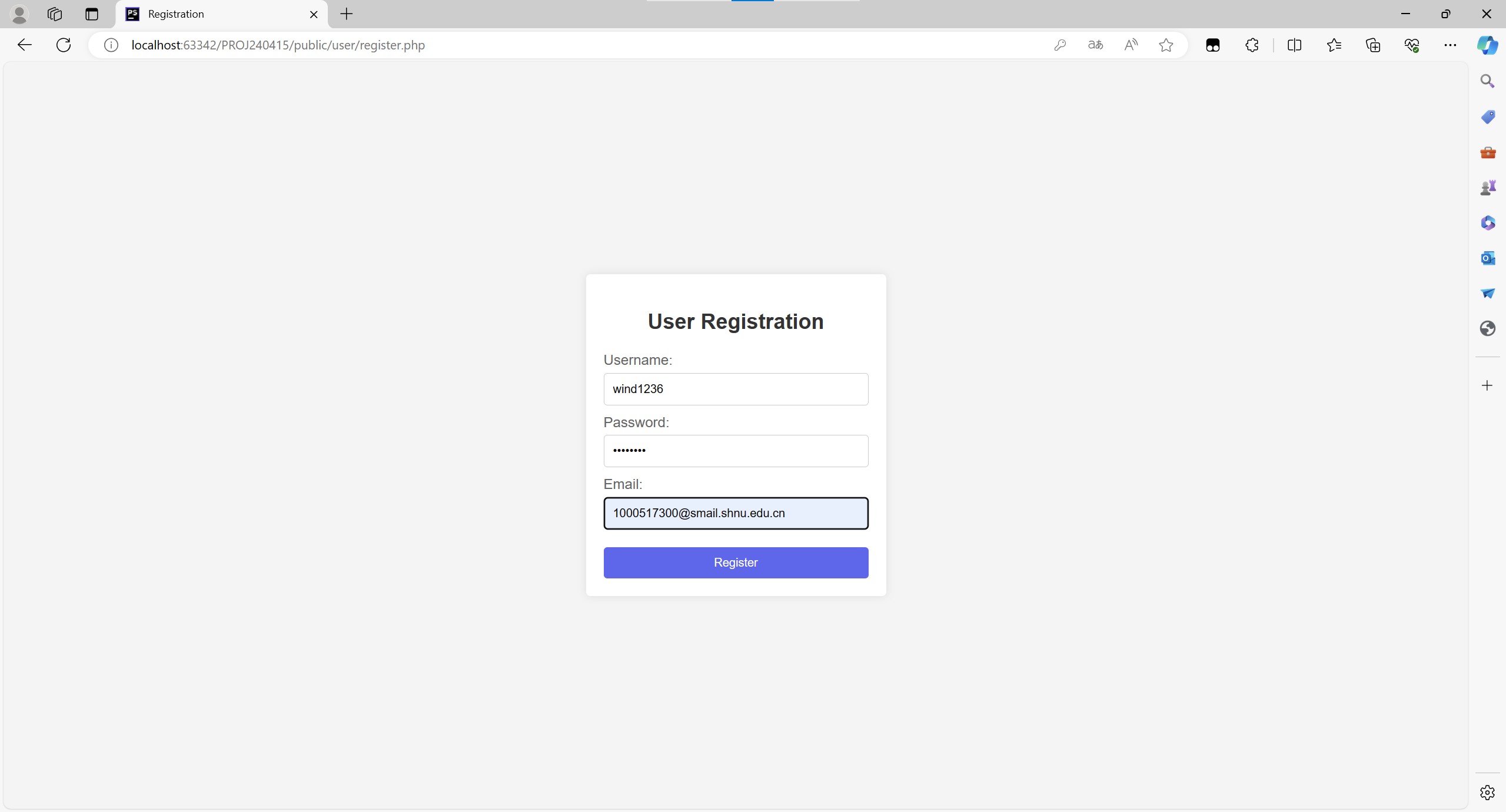
### 搜索页



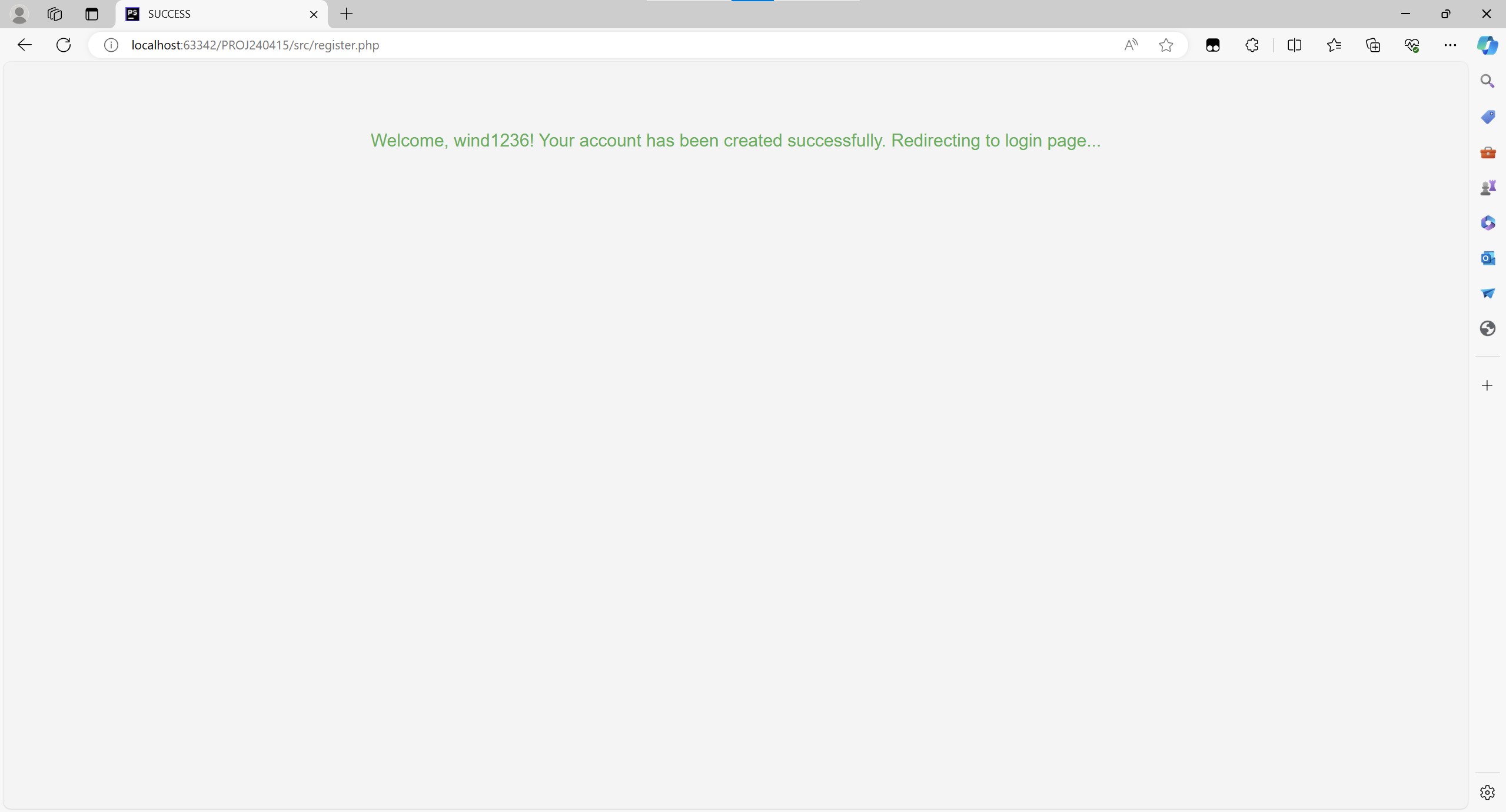
search

## 用户管理

### 注册

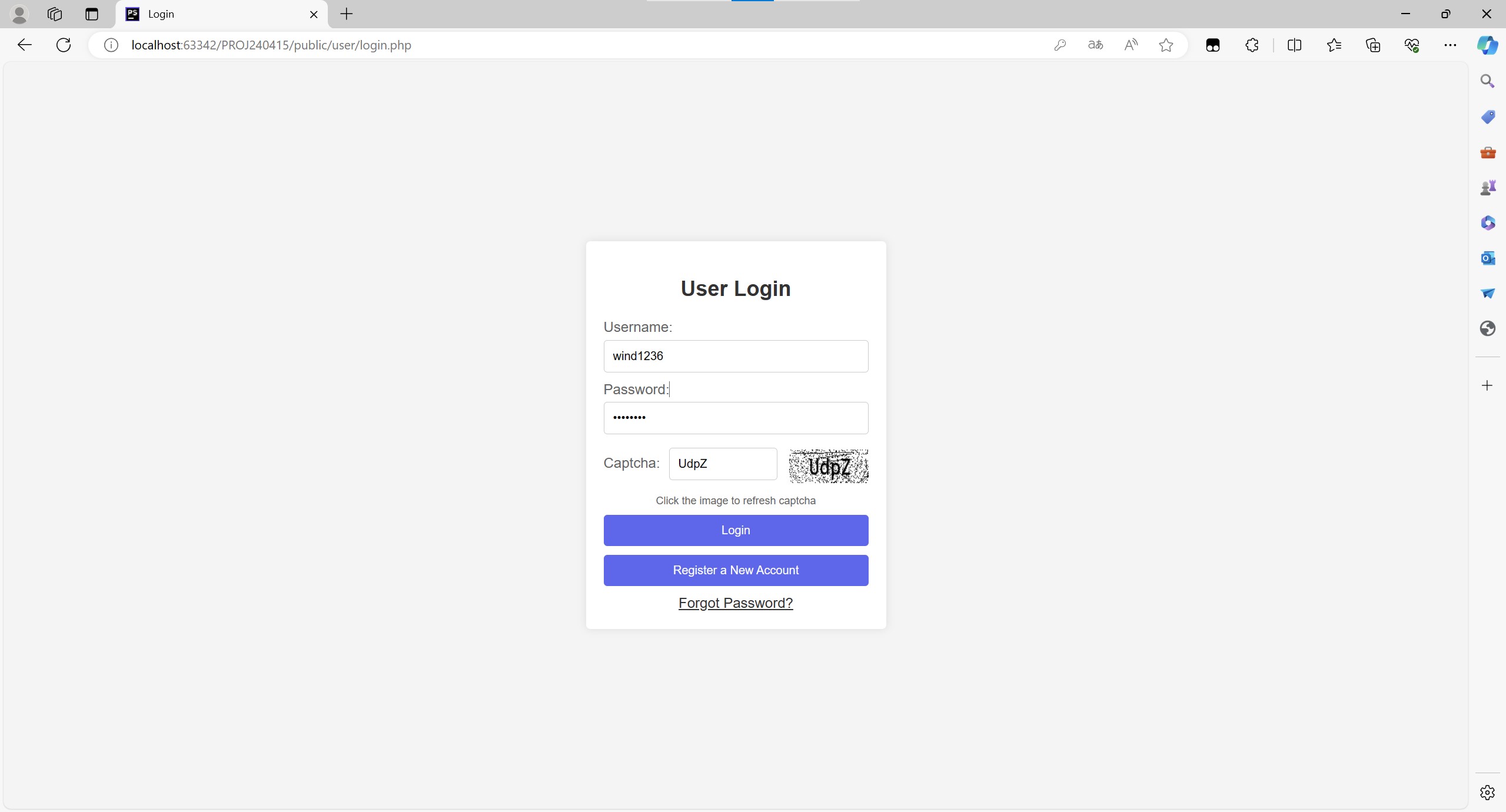


register

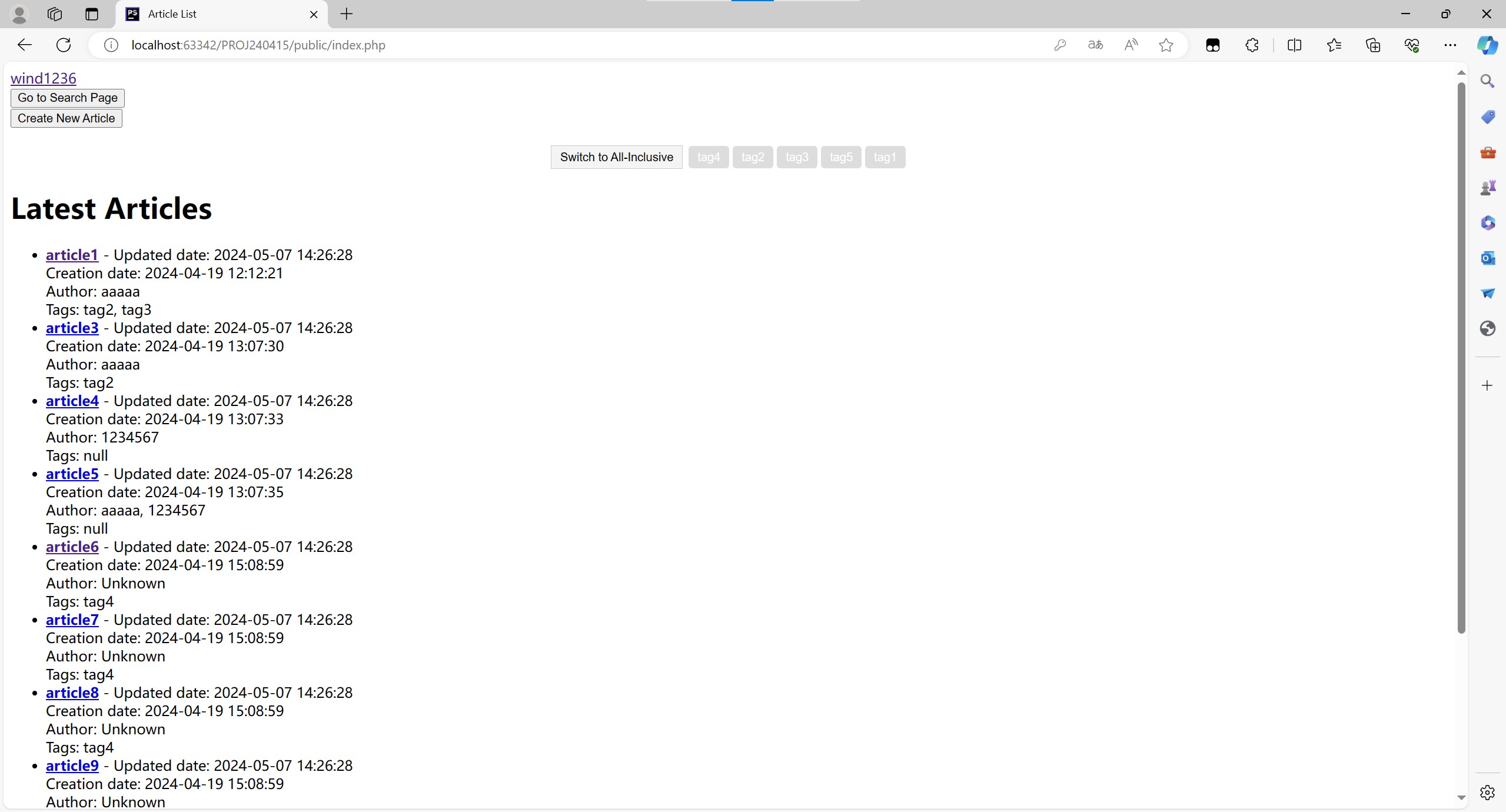


register\_success

### 登录



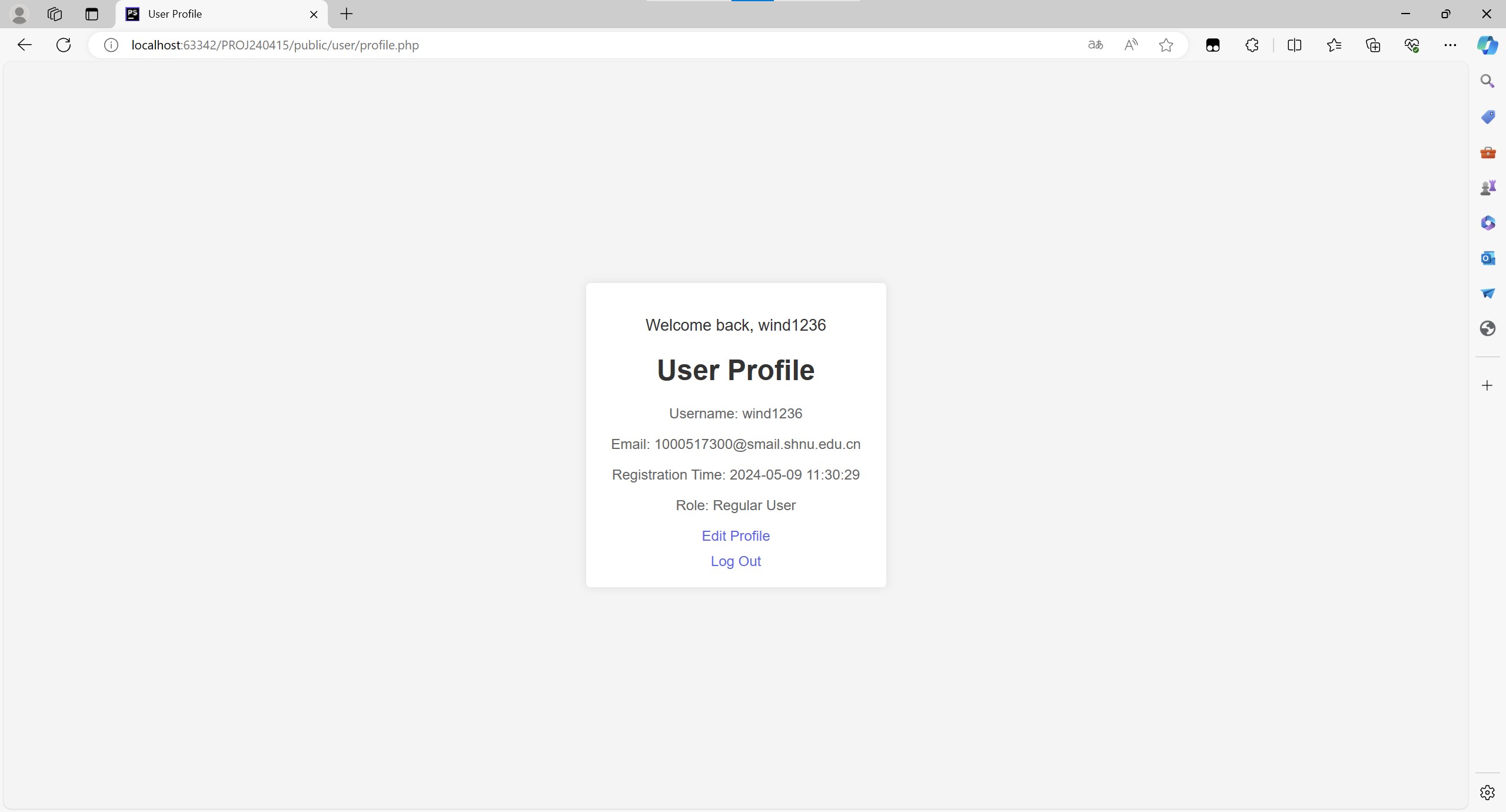
login



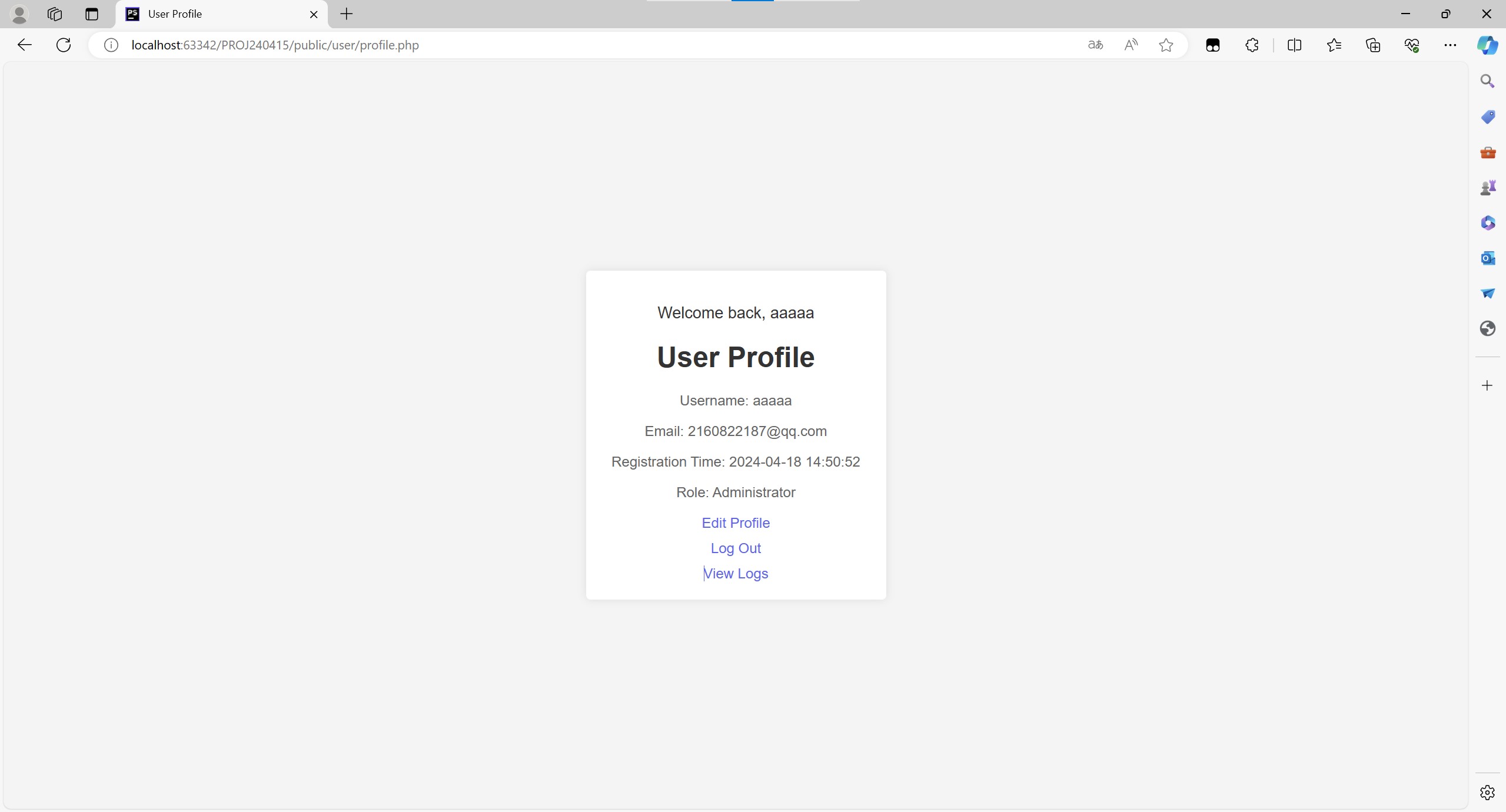
login\_success

### 个人信息

#### 查看个人信息

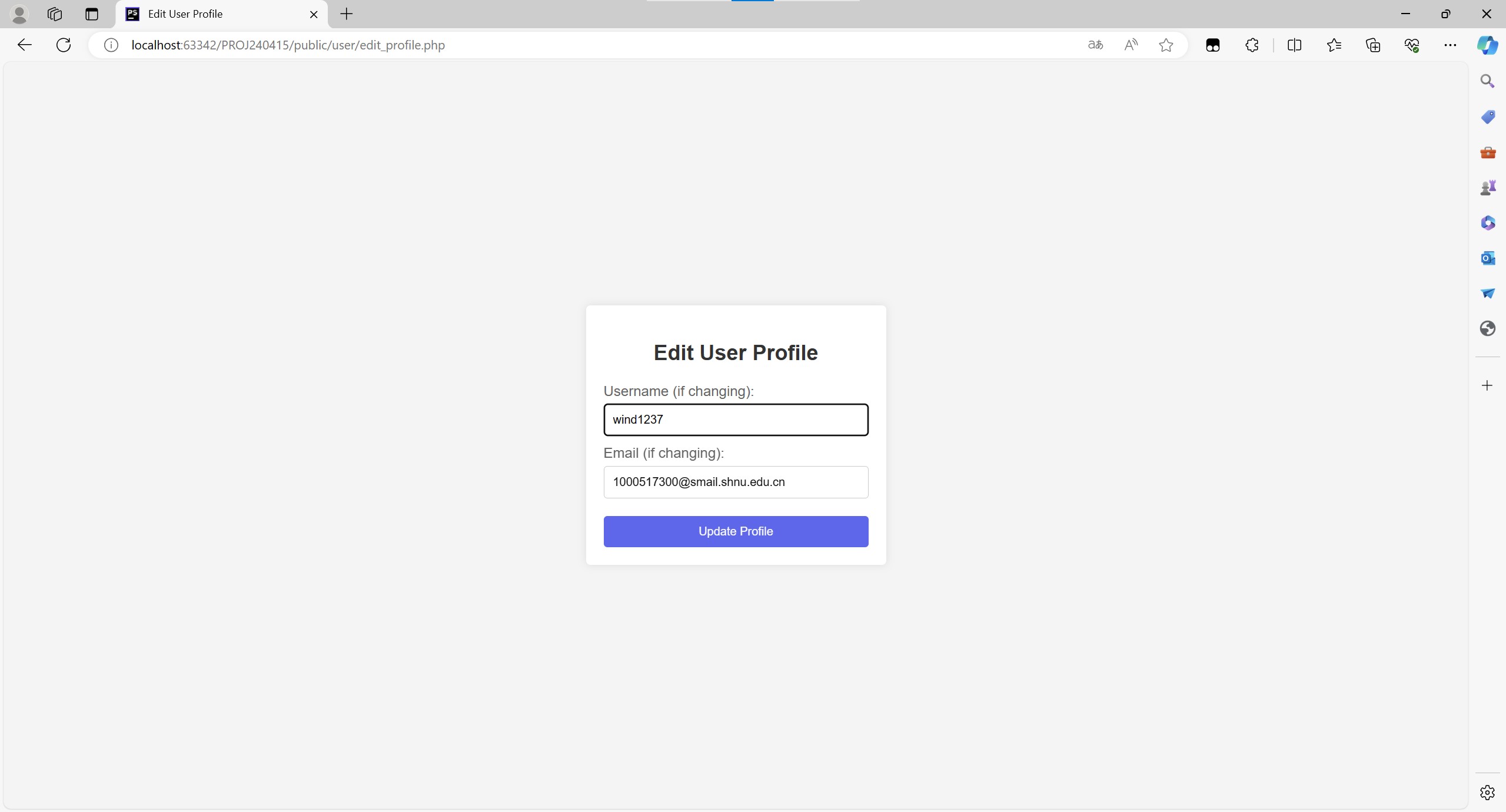


profile

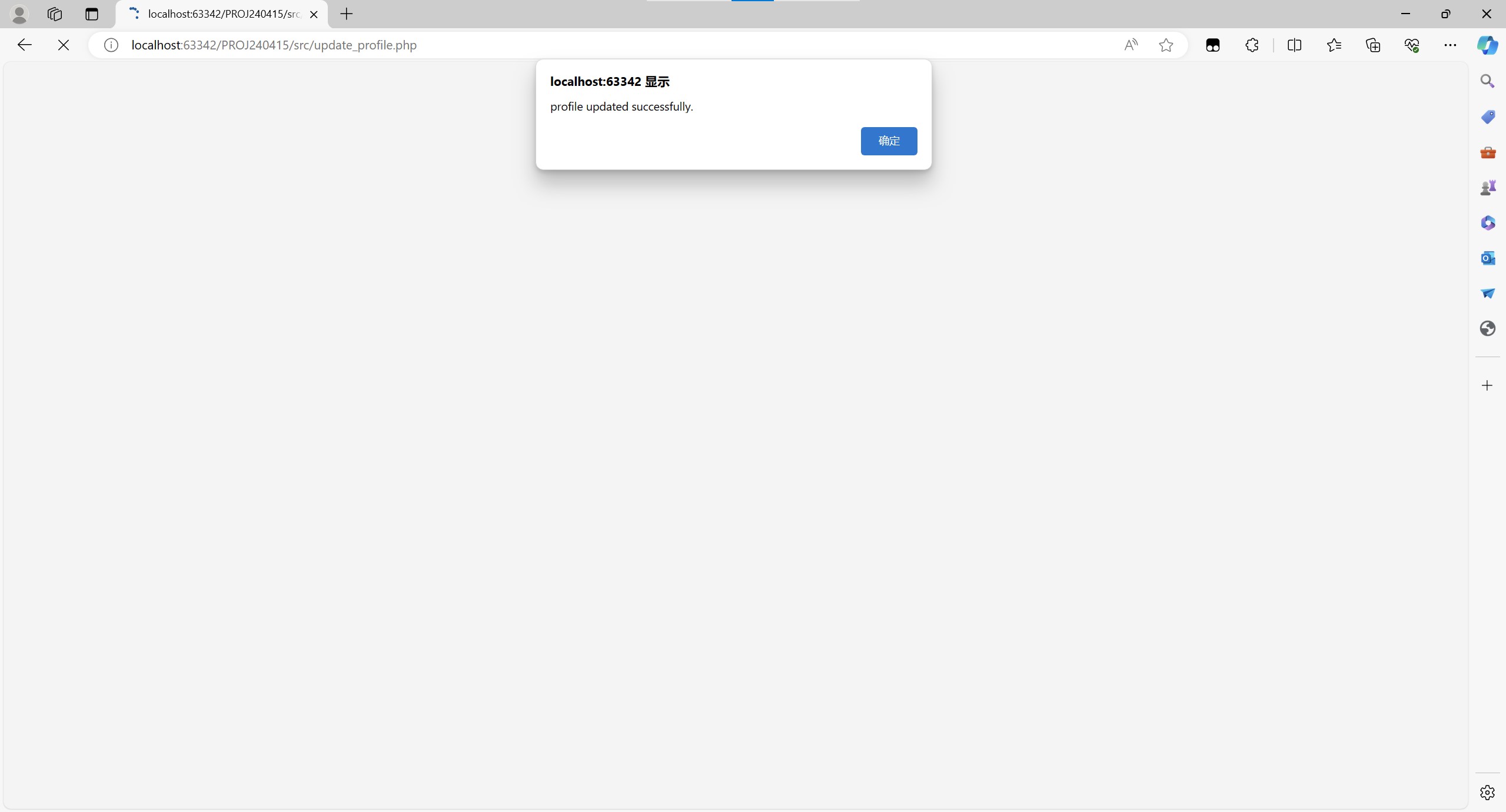


profile\_admin

#### 修改个人信息

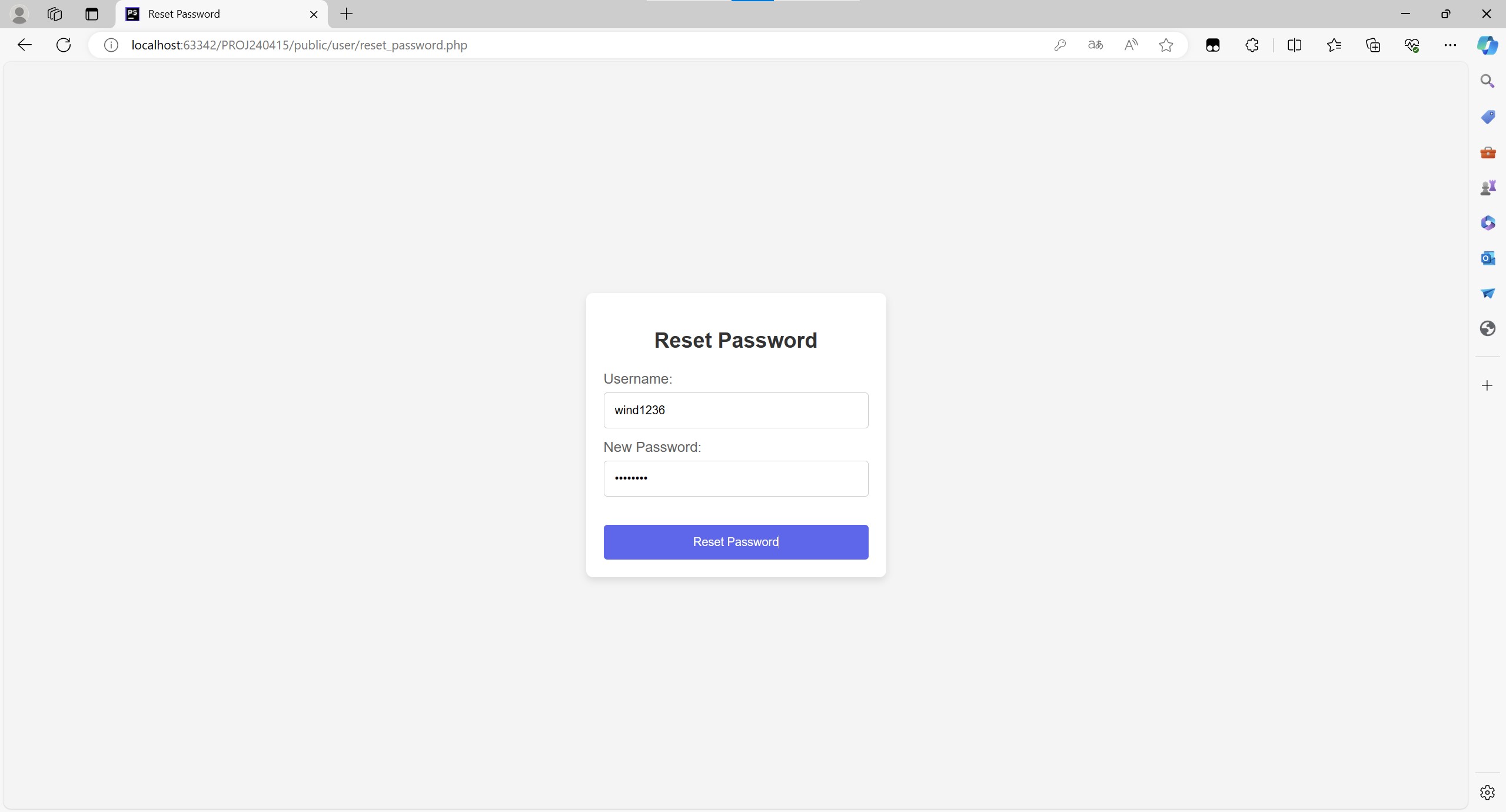


edit\_profile

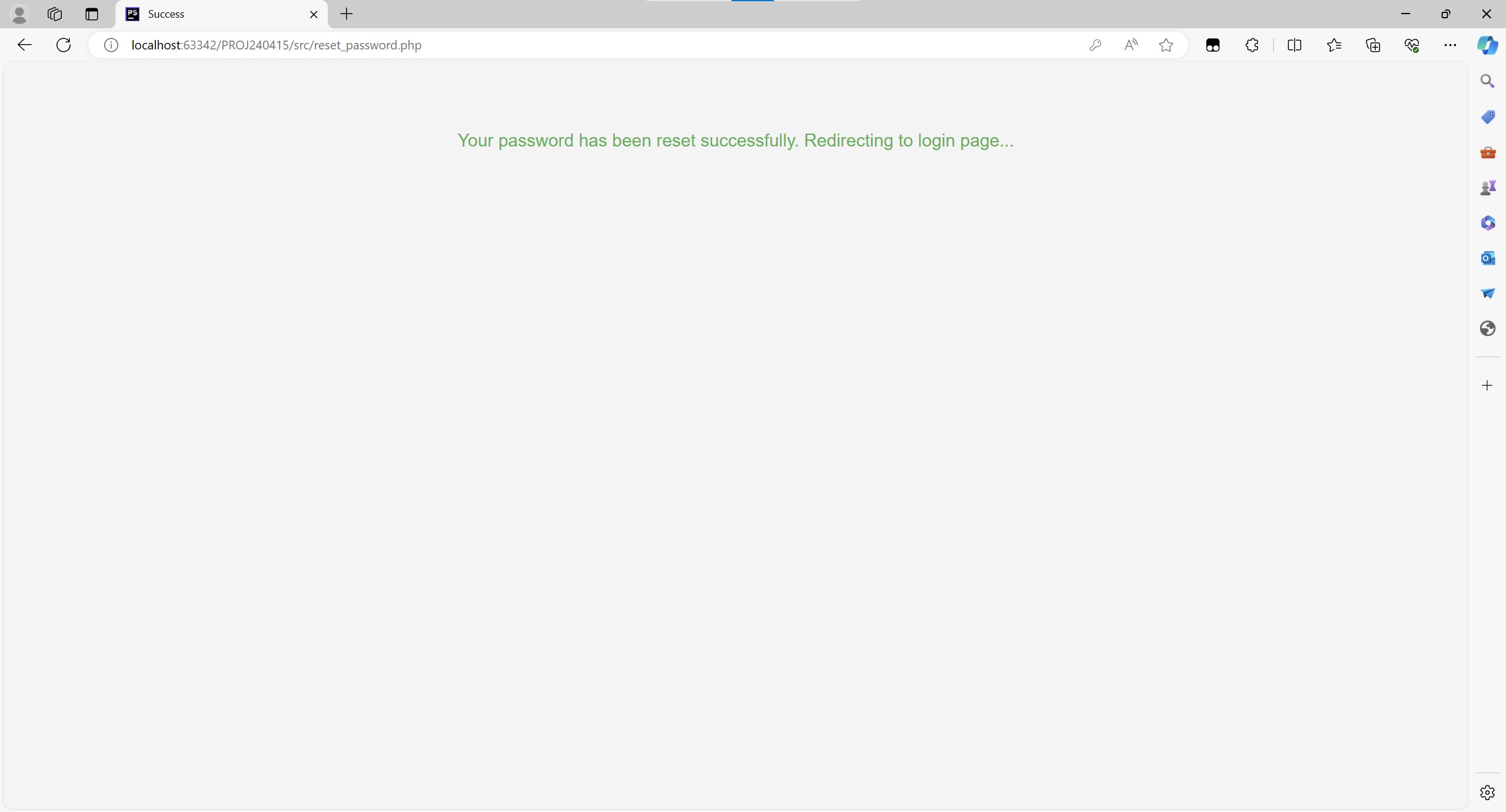


edit\_profile\_success

### 修改密码

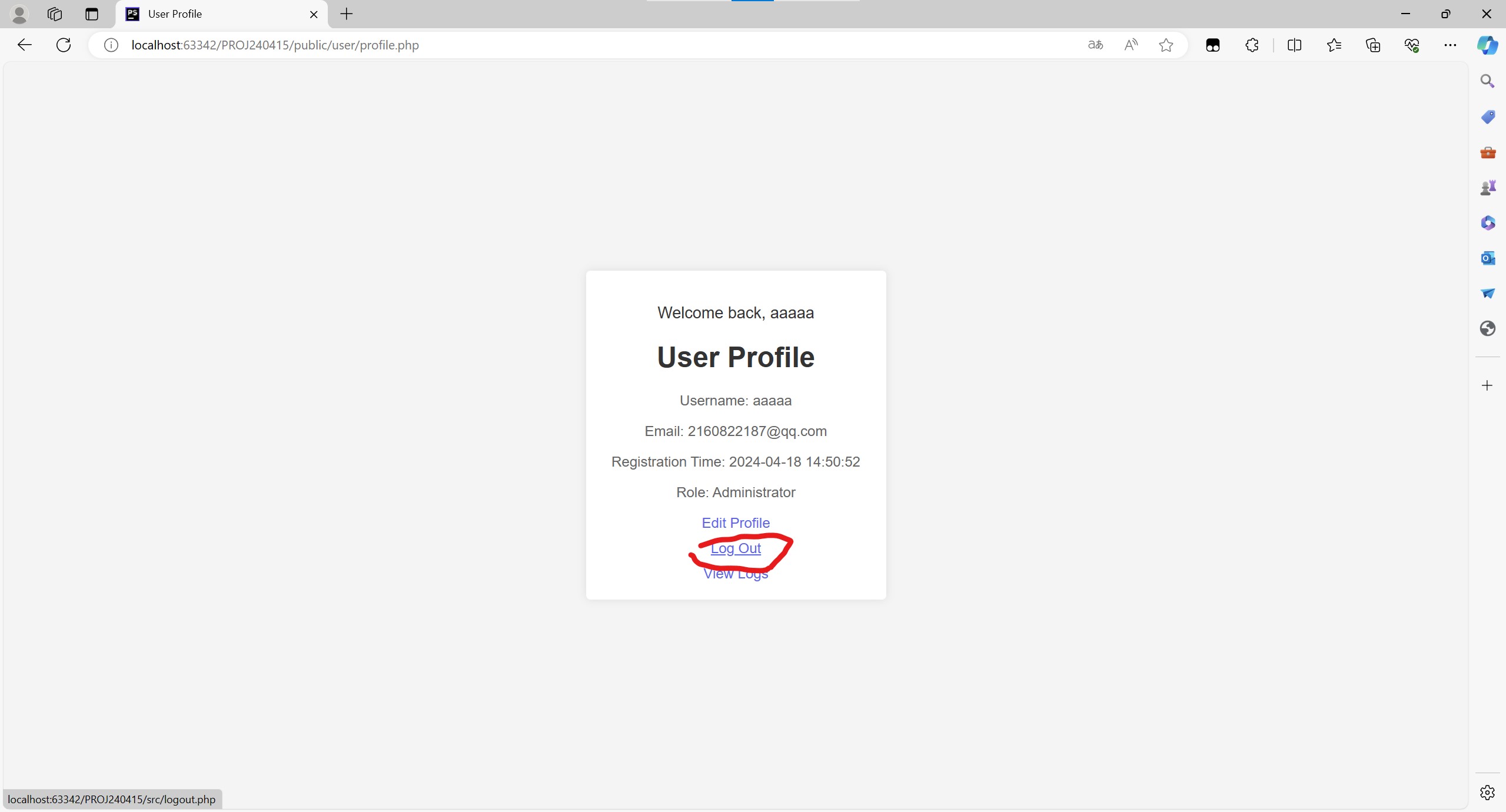


reset\_password



reset\_password\_success

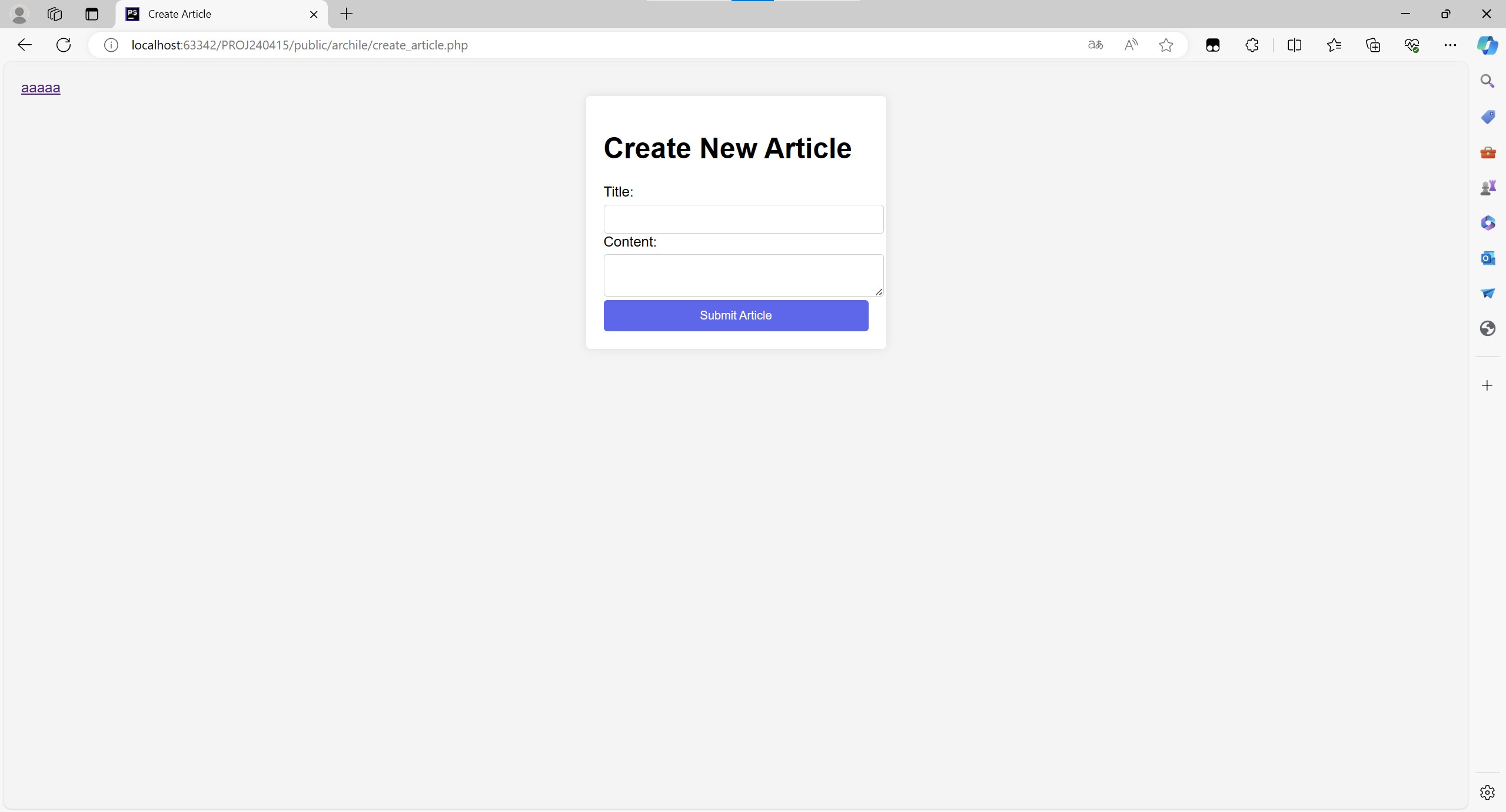
### 退出登录



logout

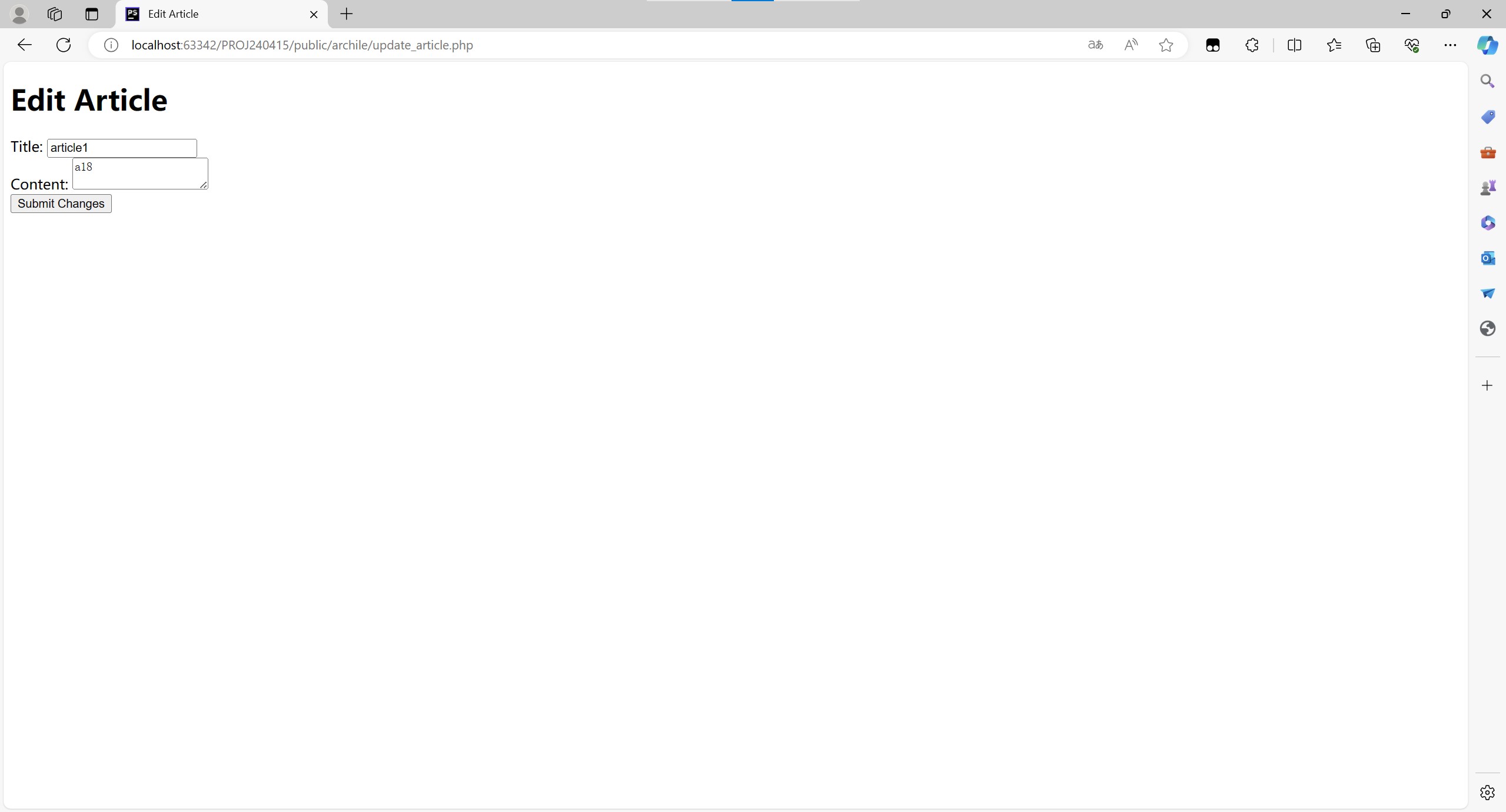
## 论坛管理

### 创建文章



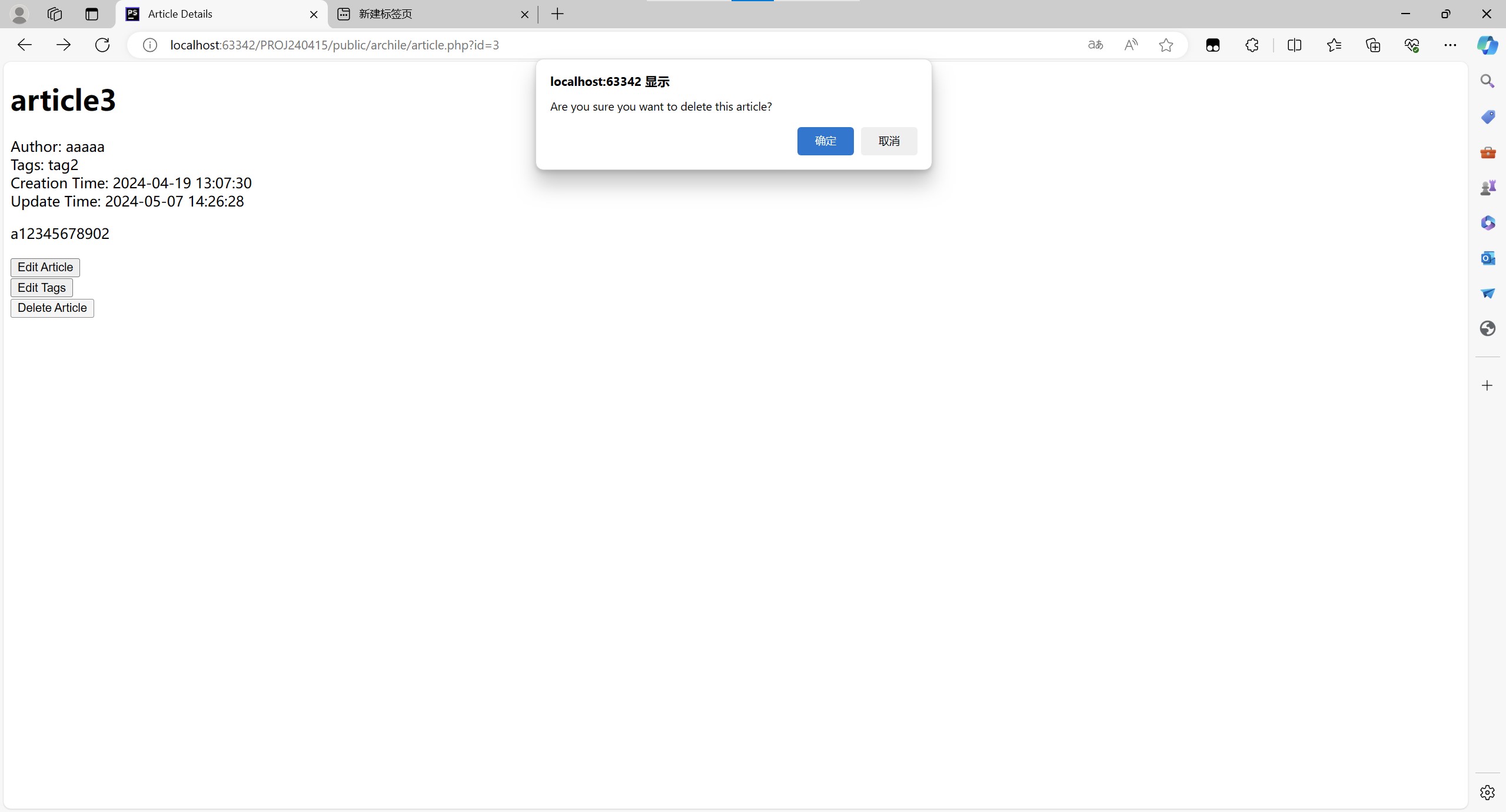
create\_article

### 编辑文章



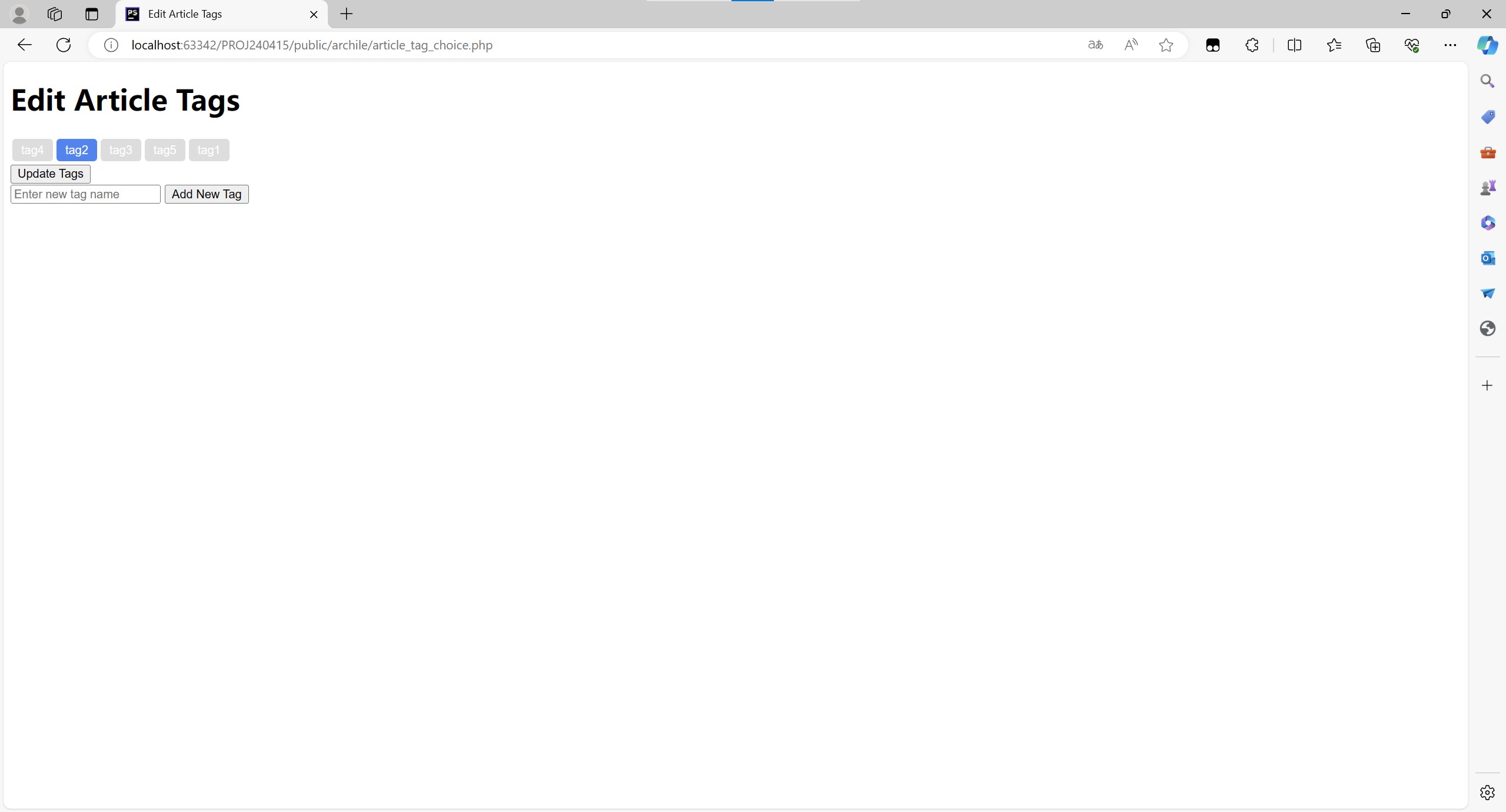
update\_article

### 删除文章



delete\_article

### 为文章编辑TAG



article\_tag\_choice

## 功能与文件对照表

* 根目录

| Function | Nom de fichier |
| --- | --- |
| 网站的主页，用于文章的浏览功能 | index.php |
| 关键字搜索功能 | search\_results.php |

* db

| Function | Nom de fichier |
| --- | --- |
| 数据库初始化 | make.sql |

* logs

| Function | Nom de fichier |
| --- | --- |
| 文章相关操作日志 | article\_logs.txt |
| 用户相关操作日志 | user.txt |
| 每日流量分析 | v1.php |
| 每日新增/活跃 | v2.php |
| 最热时段/页面/操作 | v3.php |
| 每日文章统计 | v4.php |
| 用户惯用分析 | v5.php |
| 日志分析导航页面 | view.php |

* css

| Function | Nom de fichier |
| --- | --- |
| 网站样式表 | style.css |
| 网站样式表 | style1.css |

* public
  + archile

| Function | Nom de fichier |
| --- | --- |
| 展示单个文章详情 | article.php |
| 更新文章TAG（包括增加新TAG） | article\_tag\_choice.php |
| 创建文章 | create\_article.php |
| 更新文章 | update\_article.php |

* + user

| Function | Nom de fichier |
| --- | --- |
| 更新用户资料 | edit\_profile.php |
| 用户登录 | login.php |
| 展示用户资料 | profile.php |
| 用户注册 | register.php |
| 找回（重置）密码 | reset\_password.php |

* src

| Function | Nom de fichier |
| --- | --- |
| 生成验证码 | captcha.php |
| 用于数据库连接的基本配置 | config.php |
| API 校验用户登录 | login.php |
| API 用户登出 | logout.php |
| API 校验用户注册 | register.php |
| API 重置用户密码 | reset\_password.php |
| API 更新用户个人资料 | update\_profile.php |

* + api

| Function | Nom de fichier |
| --- | --- |
| API 增加新的TAG | add\_tag.php |
| API 返回文章详情 | article.php |
| API 返回文章总数 | article\_count\_view.php |
| API 为文章增加或者移除TAG | article\_tag.php |
| API 按照选择的TAG，返回分页后的文章列表（任意包含） | articles.php |
| API 按照选择的TAG，返回分页后的文章列表（全包含） | articles\_all\_tags.php |
| API 新建文章 | creat\_article.php |
| API 删除文章 | delete\_article.php |
| API 返回文章包含的所有TAG | get\_article.php |
| API 返回标题中包含关键字的所有文章的分页列表 | search\_articles.php |
| API 返回所有TAG（按照热度降序） | tags.php |
| API 更新文章 | update\_article.php |

# 功能实现

## 基本浏览

### 首页

首页是网站的主页，用于展示文章的浏览功能，包括：

* 按照更新时间排序的文章列表（默认）
* 按照TAG分类的文章列表
* 导航到搜索页
* 导航到登录页（未登录时）
* 导航到个人信息页（已登录时）

#### 文章列表展示

首页展示的文章列表按照文章的更新时间排序，默认展示最新更新的文章。每篇文章的显示格式包括文章标题、更新日期、创建日期、作者以及相关标签。用户可以通过点击文章标题访问文章的详细页面。

#### 文章标签分类展示

首页包含一个动态生成的文章标签列表。这些标签通过调用后端接口 src/api/tags.php 获取，并以按钮形式显示在页面上。用户可以点击这些标签按钮来过滤文章列表，根据选择的标签来显示相关文章。

##### 标签切换逻辑

有一个切换按钮 Switch to All-Inclusive 允许用户在包含所有选择标签的文章（“全部包含”模式）和包含任一选择标签的文章（“任一包含”模式）之间切换。这个功能通过前端逻辑实现，改变按钮的状态和类名来表示当前模式。

##### API 调用说明

* **加载标签**：当页面加载时，loadTags 函数会调用 src/api/tags.php API 以获取所有可用的标签。此 API 返回标签的列表，每个标签用于生成页面上的一个按钮。
* **筛选文章**：当用户点击标签按钮筛选文章时，根据当前的标签筛选模式（全部包含或任一包含），页面会调用不同的 API：
  + **全部包含模式**：调用 src/api/articles\_all\_tags.php，传递选中的标签ID，返回包含所有这些标签的文章。
  + **任一包含模式**：调用 src/api/articles.php，传递选中的标签ID，返回包含任一这些标签的文章。

#### 分页功能

文章列表支持分页浏览，底部有分页控制按钮包括“上一页”和“下一页”。用户还可以通过输入页码直接跳转到指定页面或选择每页显示的文章数量（5, 10, 15, 20）。这些功能通过调用分页接口来动态加载指定页面的文章数据。

#### 导航到搜索页

提供一个按钮链接到 search\_results.php，允许用户跳转到搜索页面进行文章搜索。

#### 用户登录与个人信息访问

* **未登录时**：如果用户未登录，首页顶部会显示一个登录链接，指向 user/login.php，允许用户登录。
* **已登录时**：如果用户已登录，首页顶部会显示用户的用户名，并提供一个链接到 user/profile.php，允许用户访问和编辑个人信息。

### 技术实现细节

* **后端会话管理**：使用 PHP 的 session\_start() 管理用户会话。
* **前端动态内容加载**：使用 Axios 库通过 AJAX 请求与后端 API 交互，动态加载文章列表和标签。
* **前端页面导航**：使用 JavaScript 控制页面导航逻辑，如文章分页和页面跳转。
* **安全性**：通过 PHP 的 htmlspecialchars 函数处理输出，防止 XSS 攻击。

### 搜索页

文章搜索页允许用户通过输入关键词来搜索文章标题，并展示搜索结果。以下是详细的功能说明和技术实现细节：

#### 搜索输入和按钮

* **搜索框**：页面提供一个输入框，用户可以在此输入要搜索的文章标题的关键词。
* **搜索按钮**：旁边有一个搜索按钮，用户点击后会根据输入的关键词进行搜索。搜索结果会在下方的列表中动态显示。

#### API 调用说明

* **搜索文章**：当用户点击搜索按钮后，页面会调用 src/api/search\_articles.php API，传递用户输入的查询关键词、当前页码和每页显示的项目数作为参数。该 API 返回与查询条件匹配的文章列表。
* **响应处理**：前端通过 Axios 发送 GET 请求，接收到的数据用于动态生成每篇文章的详细显示，包括标题、更新日期、创建日期、作者和标签。

### 分页功能

搜索结果支持分页显示，具体功能如下：

* **分页控件**：页面底部有分页控制按钮，包括“上一页”、“下一页”和“跳转到页”输入框，以及一个下拉选择框用于选择每页显示的文章数（5, 10, 15, 20项）。
* **动态加载**：用户可以通过点击“上一页”和“下一页”按钮浏览不同的搜索结果页。用户还可以直接输入一个页码，点击“跳转”按钮快速跳转到指定页。

### 用户登录状态显示

* **未登录时**：如果用户未登录，页面顶部会显示一个“登录”链接，点击后跳转到登录页面。
* **已登录时**：如果用户已登录，页面顶部会显示用户的用户名，点击用户名可以跳转到用户的个人资料页面。

### 技术实现细节

* **前端动态内容加载**：使用 Axios 库通过 AJAX 请求与后端 API 交互，动态加载搜索结果。
* **会话管理**：使用 PHP 的 session\_start() 管理用户会话，检查用户是否已登录，并据此显示用户信息或登录链接。
* **安全性**：使用 PHP 的 htmlspecialchars 函数处理用户输入和显示输出，防止 XSS 攻击。
* **用户交互**：通过 JavaScript 监听页面元素（如按钮和输入框）的事件，实现用户交互的响应逻辑。

### 文章详情

生成的文章列表中的每篇文章都可以点击标题进入文章详情页。 文章详情页为用户提供了查看文章全文和相关信息的功能，并为管理员提供了编辑和删除文章的选项。以下是详细的功能说明和技术实现细节：

#### 页面布局

* **标题和元信息**：页面显示文章的标题、作者、标签、创建时间和更新时间。
* **文章内容**：文章的正文内容显示在页面的主体部分，支持基本的HTML格式，如换行。

#### API 调用

* **获取文章详情**：页面加载时，通过调用 ../../src/api/article.php API 获取文章的详细信息。此 API 需要传递文章ID作为参数，返回的数据用于填充页面上的各个部分。

### 管理功能（仅限管理员）

#### 编辑文章

* **编辑文章表单**：如果用户是管理员，页面将显示一个“编辑文章”按钮。点击此按钮将通过表单提交的方式导向 update\_article.php，其中包含文章ID的隐藏字段。
* **编辑标签表单**：同样地，还有一个“编辑标签”按钮，点击后导向 article\_tag\_choice.php，用于编辑文章的标签。

#### 删除文章

* **删除按钮**：管理员还可以看到一个“删除文章”按钮。点击此按钮将触发删除操作，经用户确认后，调用 ../../src/api/delete\_article.php API 删除文章。
* **删除确认**：点击删除按钮时，会弹出确认对话框，确认后才执行删除操作，以防误操作。

### 技术实现细节

* **权限控制**：页面在服务器端检查用户的登录状态和管理员状态。如果用户未登录，则重定向到登录页面。如果用户不是管理员，则不显示管理功能。
* **前端动态内容加载**：使用 Axios 库通过 AJAX 请求与后端 API 交互，动态加载文章详情。
* **用户交互**：通过 JavaScript 监听事件和处理用户操作，如点击按钮执行搜索、编辑和删除操作。
* **安全性**：使用 PHP 的 htmlspecialchars 函数处理显示的用户生成内容，以防止 XSS 攻击。

## 用户管理

### 注册

注册页存在一个表单可以输入用户名、密码以及邮箱。 当用户提交时，会将表单以POST提交给src/register.php，该API会检查提交是否合法。

一个合法的提交必须满足：

* 用户名必须至少有5个字符，并且不存在重名用户
* 密码必须至少有8个字符，并且同时包含数组和字母
* 邮箱合法

如果提交不合法，会以GET方式返回注册页面，注册页面会检查是否有error字段，并会按照关键字生成相关错误提示。 否则会展示欢迎界面，并在数秒后转到登录界面。

### 登录

登录页面存在一个表单要求用户输入用户名、密码以及验证码，并且存在一个注册按钮以及表示“我忘记密码”的链接。

点击注册按钮会导航至注册页面。 点击“我忘记密码”链接，会导航至重置密码页面。 点击验证码图片会刷新验证码。

验证码时在src/captcha.php生成的，正确的密钥保存于$\_SESSION中。

当用户提交登录表单时，会将表单以POST提交给src/login.php，该API会检查提交是否合法。

一个合法的提交必须满足： - 验证码正确 - 用户名与密码匹配

当用户提交不合法时，会以GET方式返回登录页面，登录页面会检查是否有error字段，并会按照关键字生成相关错误提示。

否则会跳转至首页。

#### 管理员

管理员用户的并无特殊的登录或者管理页面，但是管理员用户可以修改文章，删除文章，以及为文章编辑TAG。

管理员用户的标志是$\_SESSION['is\_admin']为true。

如果用户是管理员，会在文章详情页展示编辑tag、编辑文章和删除文章按钮，以及在个人信息页展示查看LOG分析的链接。

并不需要担心普通用户通过修改前端代码来获取管理员权限，因为API会检查用户是否为管理员。

### 个人信息

#### 查看个人信息

个人信息页面会展示用户的用户名，邮箱，是否为管理员，以及注册时间。并且会生成指向修改个人信息的链接、登出的链接以及浏览LOG分析的链接。

#### 修改个人信息

修改个人信息页面存在一个表单，要求用户输入新的用户名、邮箱。

首先会查询用户的原始信息，作为默认值展示在表单中。

当用户提交修改表单时，会将表单以POST提交给src/update\_profile.php，该API会检查提交是否合法。

一个合法的提交必须满足： - 用户名必须至少有5个字符，并且不存在重名用户 - 邮箱合法

如果提交不合法，会以GET方式返回修改个人信息页面，修改个人信息页面会检查是否有error字段，并会按照关键字生成相关错误提示。

否则会提示修改成功，并在数秒后转到个人信息页面。

### 修改密码

修改密码页面存在一个表单，要求用户输入用户名、新密码。

当用户提交修改密码表单时，会将表单以POST提交给src/reset\_password.php，该API会检查提交是否合法。

一个合法的提交必须满足： - 密码必须至少有8个字符，并且同时包含数组和字母

如果提交不合法，会以GET方式返回修改密码页面，修改密码页面会检查是否有error字段，并会按照关键字生成相关错误提示。

否则会提示修改成功，并在数秒后转到登录页面。

### 退出登录

退出登录会清空$\_SESSION以及$\_COOKIE并且跳转至首页。

$\_SESSION = array();  
session\_destroy();

setcookie("username", "", time()-3600, "/");

## 论坛管理

### 创建文章

对于所有登录用户，均可以创建文章，创建文章时需要填写标题，内容，并在创建成功后自动成为文章的第一位作者。

权限检查是两部分，一部分是页面展示，另一部分是API检查。

在页面展示时，会检查用户是否登录，如果未登录，会跳转至登录页面。

<div class="user-info">  
 <?php if (isset($\_SESSION['username'])): ?>  
 <a href="../user/profile.php"><?php echo htmlspecialchars($\_SESSION['username']); ?></a>  
 <?php else:  
 header('Location: ../user/login.php');  
 endif; ?>  
</div>

在API检查时，会检查用户是否登录，如果未登录，会返回错误信息。

$author\_id = $\_SESSION['user\_id'] ?? null;  
if (!$author\_id) {  
 echo json\_encode(['success' => false, 'message' => 'login required']);  
 exit;  
}

创建文章页允许登录的用户创建新的文章。以下是详细的功能说明和技术实现细节：

#### 文章创建表单

* **标题输入**：页面包含一个输入框，用于输入文章的标题。此输入框为必填项。
* **内容输入**：还有一个文本区域供用户输入文章的内容。这也是一个必填项。
* **提交按钮**：一个按钮用于提交文章。点击此按钮后，页面将通过 JavaScript 函数 submitArticle 发起创建文章的请求。

#### 技术实现细节

##### 前端表单处理

* **输入验证**：JavaScript 用于在前端验证标题和内容的输入。如果用户试图提交空的标题或内容，将显示一个警告消息，并阻止表单提交。
* **表单提交**：使用 JavaScript 的 submitArticle 函数处理表单的提交逻辑。该函数检查输入，构建一个包含标题和内容的 GET 请求，并发送到后端的 API。

##### API 调用

* **创建文章**：当用户点击“提交文章”按钮，JavaScript 会调用后端的 ../../src/api/create\_article.php API。此 API 接受标题和内容作为参数，创建新的文章记录。
* **成功与错误处理**：API 调用成功后，用户会收到成功的提示，并被重定向到新创建的文章的详情页面。如果创建失败或发生错误，用户将收到相应的错误消息。

##### 安全与维护

* **输入清理**：使用 PHP 的 htmlspecialchars 函数对显示的用户名进行处理，防止跨站脚本攻击（XSS）。
* **错误处理**：JavaScript 中的错误处理确保在发生网络或服务器错误时给用户清晰的反馈。

### 编辑文章

对于所有管理员用户，均可以修改文章。

权限检查是两部分，一部分是页面展示，另一部分是API检查。

在页面展示时，会检查用户是否为管理员，如果是管理员，会展示编辑tag、编辑文章和删除文章按钮。 否则不会展示这些按钮。

在API检查时，会检查用户是否为管理员，如果不是管理员，会返回错误信息。

if (!isset($\_SESSION['user\_id'])) {  
 http\_response\_code(401); echo json\_encode(['message' => 'login required']);  
 exit;  
}  
  
if (!isset($\_SESSION['is\_admin']) || $\_SESSION['is\_admin'] != 1) {  
 http\_response\_code(403); echo json\_encode(['message' => 'Access Denied: You are not an admin']);  
 exit;  
}

#### 页面布局与功能

##### 表单初始化

* **文章ID验证**：页面加载时首先检查是否有文章ID传递。如果没有文章ID，系统将显示错误消息，并将用户重定向回上一个页面。
* **表单填充**：如果存在文章ID，页面将通过调用后端API获取文章的当前详情，并自动填充到表单中，以便用户进行编辑。

#### 技术实现细节

##### 数据获取

* **获取文章详情**：使用 JavaScript 的 fetch 方法调用 ../../src/api/article.php，传递文章ID以请求当前文章的详细信息。API响应成功后，将文章的标题和内容填充到表单中，准备进行编辑。
* **错误处理**：如果获取文章详情失败，页面会显示错误消息，并允许用户返回上一页重新尝试。

##### 表单提交

* **提交更新**：表单使用 AJAX 方法提交，避免页面重新加载。用户编辑标题和内容后，点击“提交更改”按钮将通过 axios 发送 GET 请求到 ../../src/api/update\_article.php。请求包括文章ID、更新的标题和内容。
* **更新反馈**：提交后，根据服务器响应向用户显示成功或错误消息。如果更新成功，页面将重定向到文章的详情页面。

### 安全与维护

* **用户权限验证**：页面加载前，服务器端脚本检查用户是否有权进行编辑操作，确保只有有权限的用户可以编辑文章。
* **输入清理**：提交的数据通过后端脚本进行清理和验证，防止注入攻击和其他安全威胁。

### 删除文章

对于所有管理员用户，均可以删除文章。

权限检查是两部分，一部分是页面展示，另一部分是API检查。

在页面展示时，会检查用户是否为管理员，如果是管理员，会展示编辑tag、编辑文章和删除文章按钮。 否则不会展示这些按钮。

在API检查时，会检查用户是否为管理员，如果不是管理员，会返回错误信息。

if (!isset($\_SESSION['user\_id'])) {  
 http\_response\_code(401); echo json\_encode(['message' => 'login required']);  
 exit;  
}  
  
if (!isset($\_SESSION['is\_admin']) || $\_SESSION['is\_admin'] != 1) {  
 http\_response\_code(403); echo json\_encode(['message' => 'Access Denied: You are not an admin']);  
 exit;  
}

本功能直接集成在文章详情页中，管理员用户可以在文章详情页点击“删除文章”按钮来删除文章。

function deleteArticle() {  
 const articleId = new URLSearchParams(window.location.search).get('id');  
 if (confirm("Are you sure you want to delete this article?")) {  
 axios.get(`../../src/api/delete\_article.php?article\_id=${articleId}`)  
 .then(function (response) {  
 alert(response.data.message);  
 if (response.data.message.startsWith('Success')) {  
 // back  
 window.history.back();  
 }  
 })  
 .catch(function (error) {  
 console.error('Error deleting the article:', error);  
 alert('An error occurred while deleting the article');  
 });  
 }  
 }

### 为文章编辑TAG

对于所有管理员用户，均可以为文章编辑TAG。

权限检查是两部分，一部分是页面展示，另一部分是API检查。

在页面展示时，会检查用户是否为管理员，如果是管理员，会展示编辑tag、编辑文章和删除文章按钮。 否则不会展示这些按钮。

在API检查时，会检查用户是否为管理员，如果不是管理员，会返回错误信息。

if (!isset($\_SESSION['user\_id'])) {  
 http\_response\_code(401); echo json\_encode(['message' => 'login required']);  
 exit;  
}  
  
$userId = $\_SESSION['user\_id'];  
  
if ($\_SERVER['REQUEST\_METHOD'] != 'POST') {  
 http\_response\_code(405); echo json\_encode(['message' => 'Request method must be POST']);  
 exit;  
}  
  
if (!isset($\_SESSION['is\_admin']) || $\_SESSION['is\_admin'] != 1) {  
 http\_response\_code(403); echo json\_encode(['message' => 'Access Denied: You are not an admin']);  
 exit;  
}

if (!isset($\_SESSION['user\_id'])) {  
 http\_response\_code(401); echo json\_encode(['message' => 'login required']);  
 exit;  
}  
  
if (!isset($\_SESSION['is\_admin']) || $\_SESSION['is\_admin'] != 1) {  
 http\_response\_code(403); echo json\_encode(['message' => 'Access Denied: You are not an admin']);  
 exit;  
}

## 会话管理

### $\_SESSION

当用户登录时，会将用户的用户名保存在$\_SESSION中，以便在用户访问其他页面时，可以知道用户是否已经登录，并可以获知其他信息，如用户名，是否是管理员等。

if (password\_verify($password, $hashed\_password)) {  
 $\_SESSION = array();  
 $\_SESSION['loggedin'] = true;  
 $\_SESSION['username'] = $username;  
 $\_SESSION['user\_id'] = $id;  
 $\_SESSION['email'] = $email;  
 $\_SESSION['is\_admin'] = $is\_admin;  
 // other code  
 }

当用户退出登录时，会清空$\_SESSION。

$\_SESSION = array();  
session\_destroy();

### $\_COOKIE

当用户登录时，会将用户的用户名保存在$\_COOKIE中。 当在时限内再次访问网站时，会自动填充用户名。

if (password\_verify($password, $hashed\_password)) {  
 // other code  
 setcookie('username', $username, time() + 3600, '/');  
 // other code  
 }

当用户主动退出登录时，会清空$\_COOKIE。

setcookie("username", "", time()-3600, "/");

当用户筛选TAG时，会将用户的TAG选择保存在$\_COOKIE中。

if (isset($\_POST['tag\_ids'])) {  
 $tag\_ids = $\_POST['tag\_ids'];  
 setcookie('tag\_ids', $tag\_ids, time() + 3600, '/');  
}

## 分页功能

分页逻辑实现于数据库层，php只能获取一页的数据以及符合条件的数据条目数量。

这里展示按照TAG分类（包含任意一个即可）的文章列表的分页逻辑实现。

-- Create stored procedure:  
-- Query article list based on a list of tag\_ids  
-- Articles associated with Ids included in tagIds  
-- With pagination  
DROP PROCEDURE IF EXISTS FetchArticlesByTagsWithPaging;  
  
DELIMITER //  
  
CREATE PROCEDURE FetchArticlesByTagsWithPaging(  
 IN tagIds VARCHAR(255), -- Comma-separated list of tag\_ids  
 IN pageNum INT,  
 IN pageSize INT,  
 OUT totalArticles INT  
)  
BEGIN  
 DECLARE offset INT;  
 SET offset = (pageNum - 1) \* pageSize;  
 -- Helper table:  
 -- Articles associated with Ids included in tagIds  
 WITH FilteredArticles AS (SELECT at.article\_id  
 FROM article\_tags at  
 WHERE FIND\_IN\_SET(at.tag\_id, tagIds)  
 GROUP BY at.article\_id)  
 SELECT SQL\_CALC\_FOUND\_ROWS alv.article\_id,  
 alv.title,  
 alv.authors,  
 alv.created\_at,  
 alv.updated\_at,  
 alv.tags  
 FROM article\_list\_view alv  
 JOIN FilteredArticles fa ON alv.article\_id = fa.article\_id  
 ORDER BY alv.updated\_at DESC  
 LIMIT pageSize OFFSET offset;  
  
 SELECT FOUND\_ROWS() INTO totalArticles;  
END //  
  
DELIMITER ;

## 日志功能

### 日志记录

在本项目中，我们使用了文件日志的方式记录了用户的操作日志以及文章的操作日志。

User:1 Action:Login IP:127.0.0.1 Host:LAPTOP-V8J4LDQC OS:Windows Browser:Chrome AT:2024-05-06 22:04:49  
User:1 Action:Login IP:127.0.0.1 Host:LAPTOP-V8J4LDQC OS:Windows Browser:Chrome AT:2024-05-06 22:15:49  
User:1 Action:UpdateProfile AT:2024-05-06 22:16:40  
User:6 Action:Register AT:2024-05-06 22:35:26  
User:6 Action:Login IP:127.0.0.1 Host:LAPTOP-V8J4LDQC OS:Windows Browser:Chrome AT:2024-05-06 22:40:17  
User:6 Action:ResetPassword AT:2024-05-06 22:40:42

在用户日志中，我们记录了用户的登录、注册、修改个人信息、重置密码等操作。 当用户登录时，我们记录了用户的IP地址、主机名、操作系统、浏览器等信息。

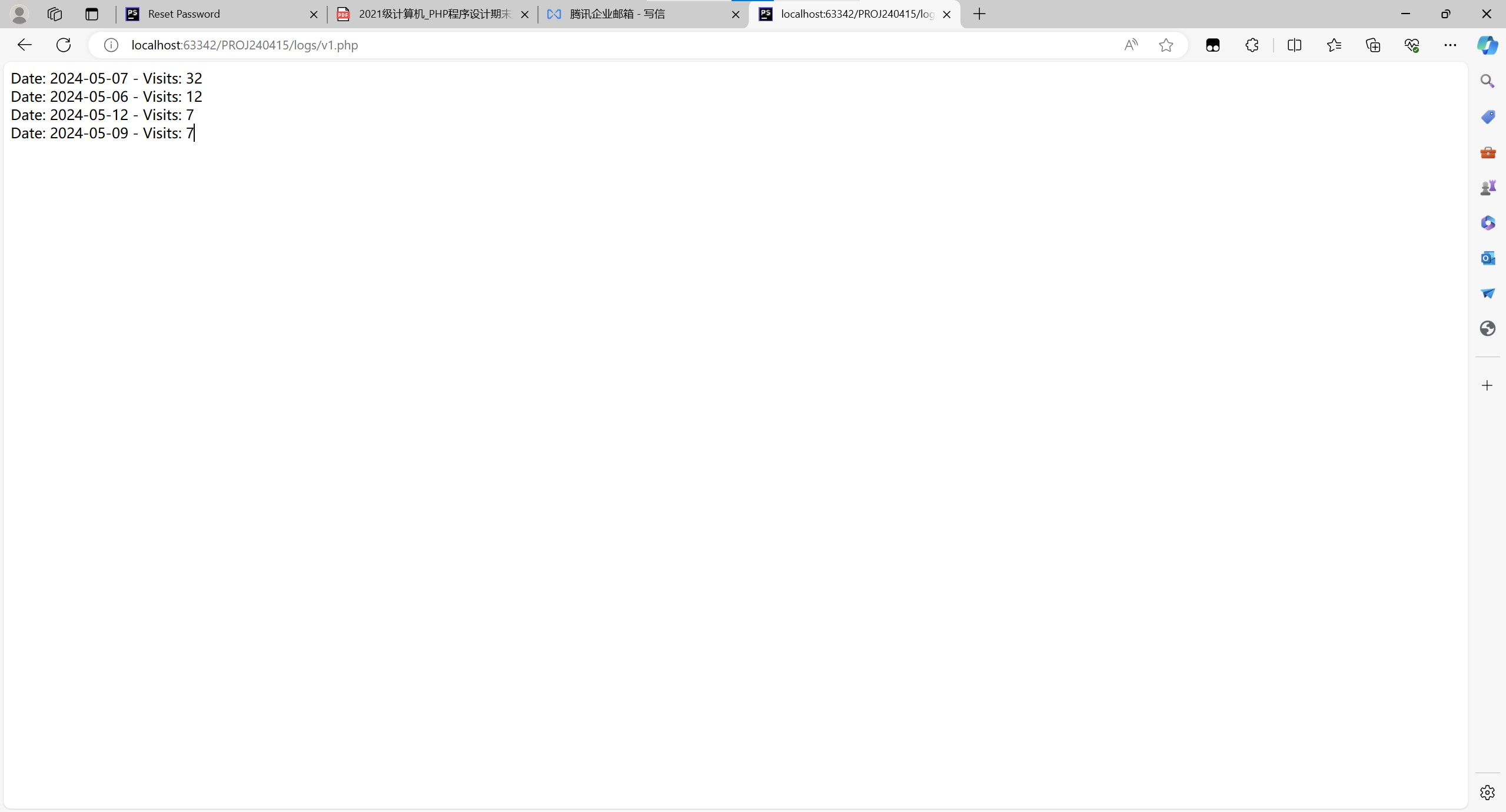
User:1 Article:18 AT:2024-05-06 22:42:47 Action:Read  
User:1 Article:35 AT:2024-05-06 22:43:01 Action:Create  
User:1 Article:35 AT:2024-05-06 22:43:02 Action:Read  
User:1 Article:35 AT:2024-05-06 22:43:08 Action:Update  
User:1 Article:35 AT:2024-05-06 22:43:13 Action:Delete  
User:1 Article:1 AT:2024-05-07 02:38:11 Action:Read

在文章日志中，我们记录了用户对文章的操作，包括阅读、创建、修改、删除等操作。

### 日志分析

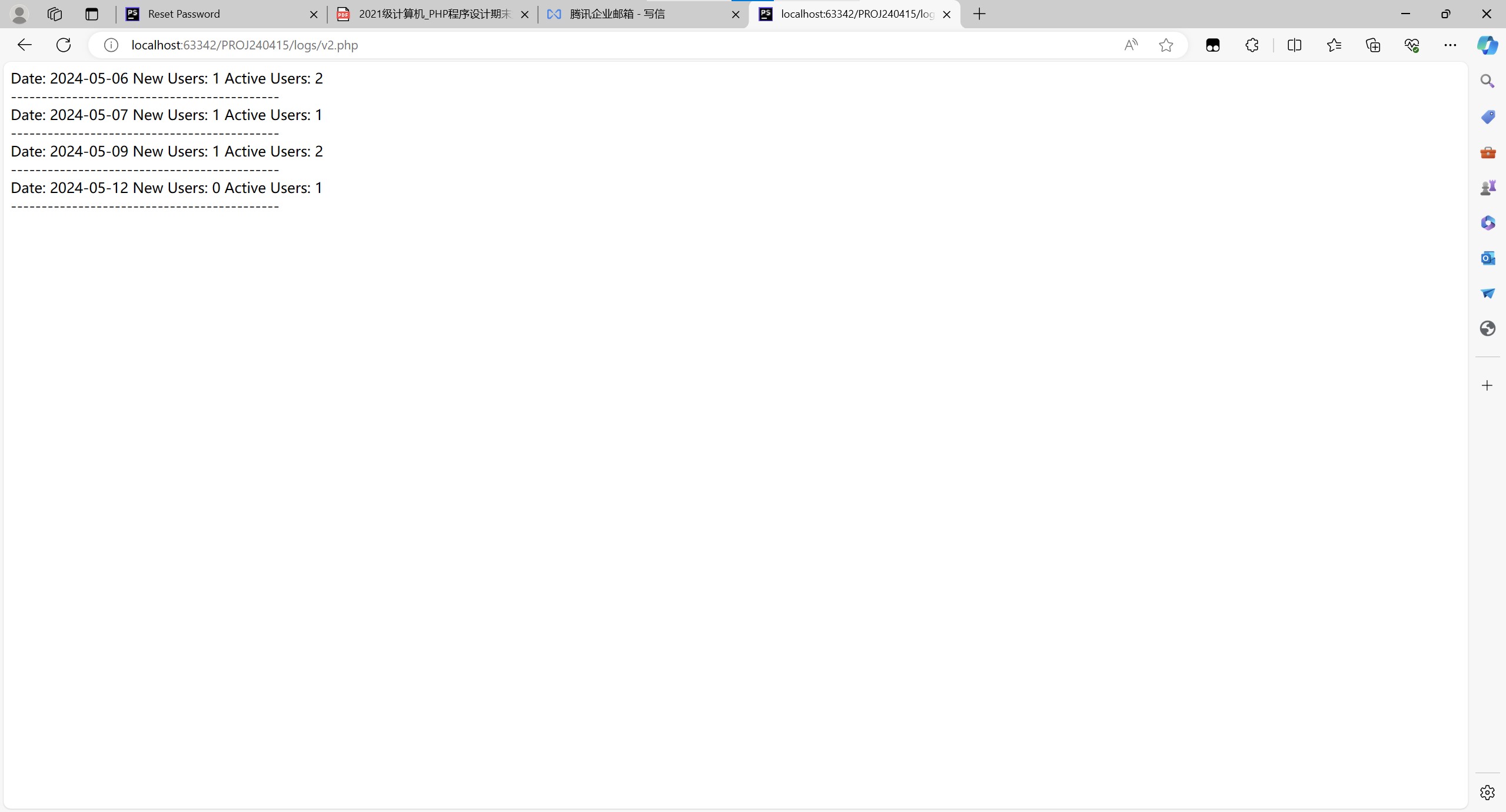
我们编写了一系列的PHP脚本，用于对日志进行分析。

v1.php: 每日流量分析



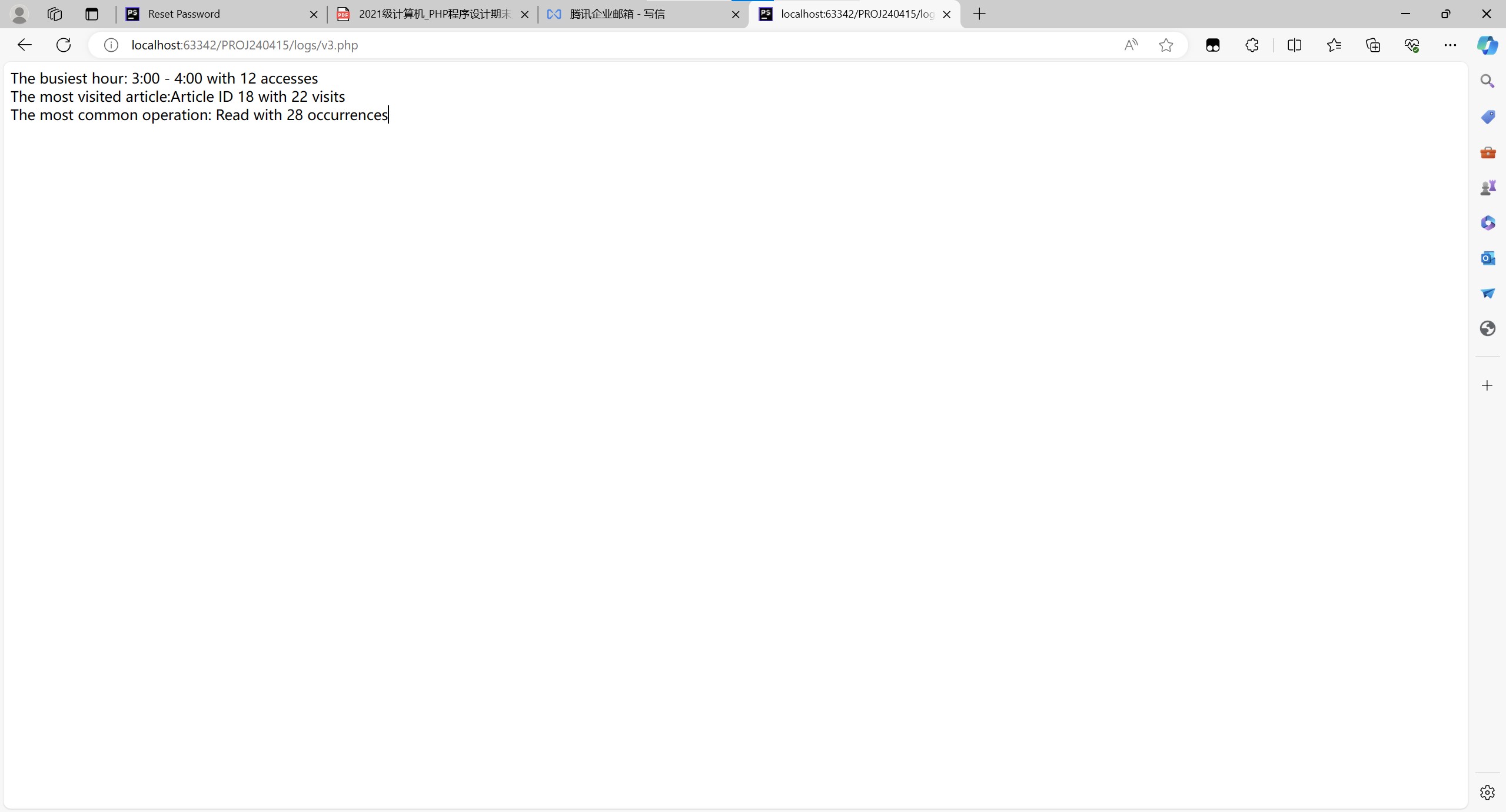
v1

v2.php: 每日新增/活跃用户分析



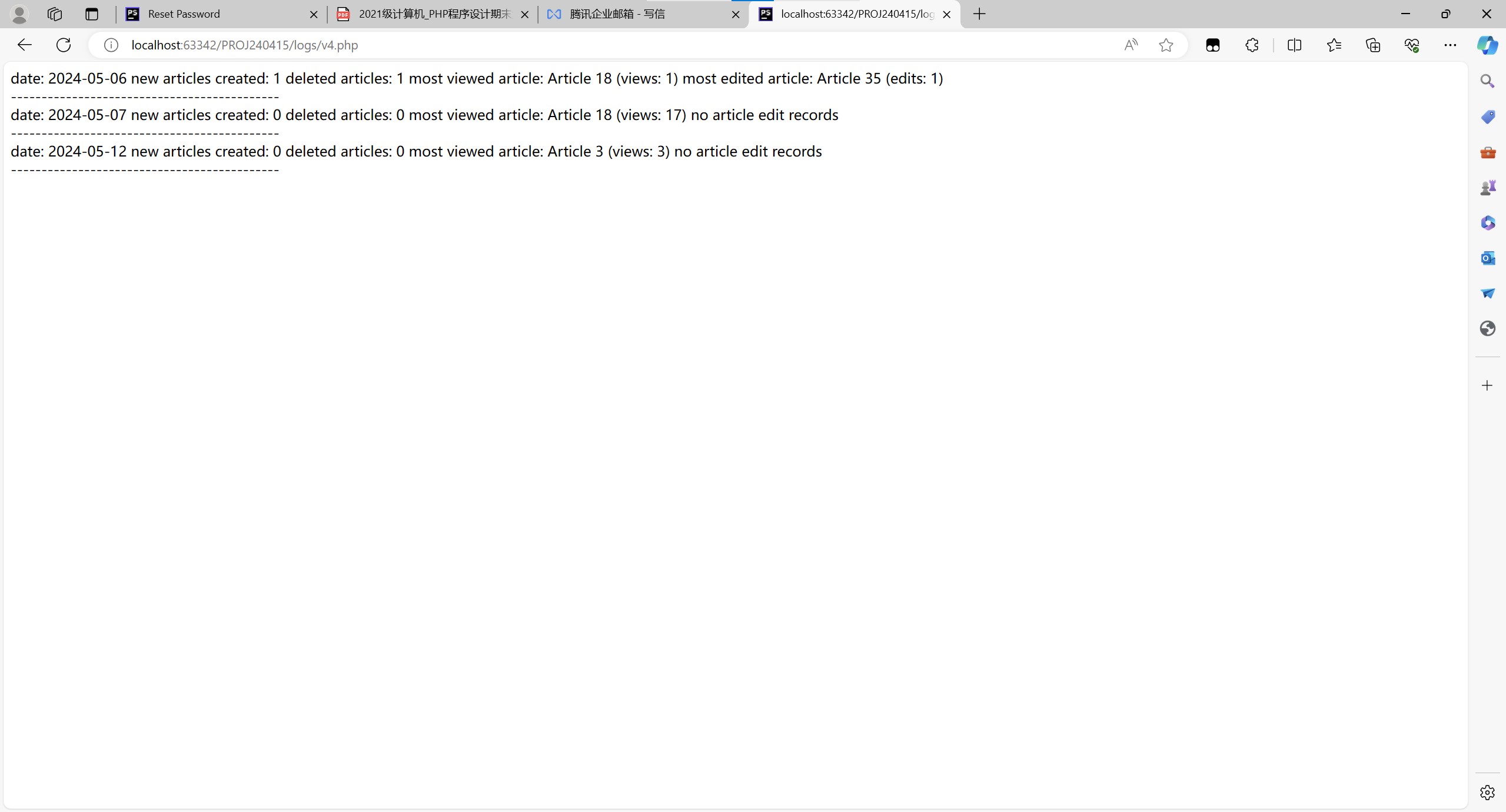
v2

v3.php: 最热时段/页面/操作分析



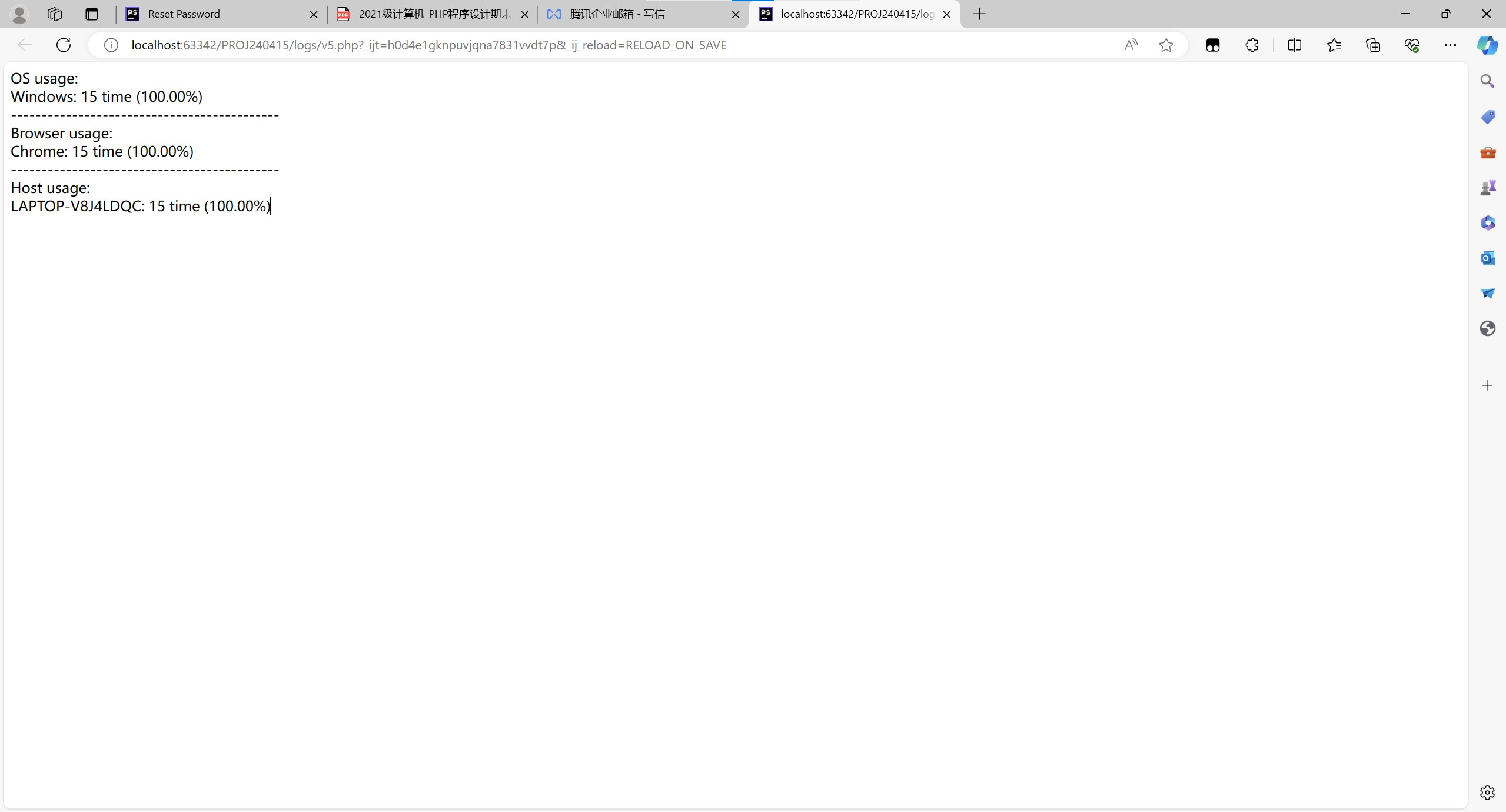
v3

v4.php: 每日文章统计



v4

v5.php: 用户惯用分析



v5

## 安全性设计

### 密码加密

密码加密使用了PHP内置的password\_hash函数，使用PASSWORD\_DEFAULT算法加密。

$hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);  
$stmt = $conn->prepare("INSERT INTO users (username, password, email) VALUES (?, ?, ?)");  
$stmt->bind\_param("sss", $username, $hashed\_password, $email);

### 登录验证码

<?php  
// header to make image stream as png  
  
$permitted\_chars = 'ABCDEFGHJKLMNPQRSTUVWXYZabcdefghjkmnpqrstuvwxyz23456789';  
$length = 4; $captcha\_text = substr(str\_shuffle($permitted\_chars), 0, $length);  
  
$\_SESSION['captcha'] = $captcha\_text;  
  
$image = imagecreatetruecolor(120, 40);  
$background\_color = imagecolorallocate($image, 255, 255, 255);  
$text\_color = imagecolorallocate($image, 0, 0, 0);  
imagefilledrectangle($image, 0, 0, 120, 40, $background\_color);  
  
$line\_color = imagecolorallocate($image, 64, 64, 64);  
for ($i = 0; $i < 6; $i++) {  
 imageline($image, mt\_rand(0, 120), mt\_rand(0, 40), mt\_rand(0, 120), mt\_rand(0, 40), $line\_color);  
}  
  
for ($i = 0; $i < 1000; $i++) {  
 imagesetpixel($image, mt\_rand(0, 120), mt\_rand(0, 40), $text\_color);  
}  
  
$font = '../ttf/CascadiaMono.ttf'; imagettftext($image, 20, 0, 30, 30, $text\_color, $font, $captcha\_text);  
  
// Output the image

验证码是在src/captcha.php生成的，正确的密钥保存于$\_SESSION中。 生成的验证码图片会在登录页面展示，并且点击验证码图片会刷新验证码。 验证码图片的生成逻辑为4个随机字符，以及一些干扰线和点。

### SQL注入防范

#### 应用层

为了防止SQL注入，我们使用了mysqli的预处理语句绑定参数的方式，而不是直接拼接SQL语句。

以下是一个用户登录API的示例：

$sql = "SELECT user\_id, username, password, email, is\_admin FROM users WHERE username = ?";  
$stmt = $conn->prepare($sql);  
$stmt->bind\_param("s", $username);  
$stmt->execute();  
$stmt->bind\_result($id, $username, $hashed\_password, $email, $is\_admin);  
$stmt->fetch();

#### 数据库层

在数据库层，我们大量使用了存储过程来执行SQL语句，而不是直接在PHP中拼接SQL语句。

以下是一个更新文章TAG列表的API的示例：

$stmt = $conn->prepare("CALL UpdateArticleTags(?, ?, ?)");  
$stmt->bind\_param("iss", $article\_id, $add\_tags, $remove\_tags);  
$result = $stmt->execute();