# 前端功能及实现文档

本文档描述了 PDFSharingWebAPI 项目前端应用的功能、页面结构以及使用 HTML, CSS 和 JavaScript 实现这些功能并与后端 Web API 交互的原理。

## 页面功能概述

1. **main.html (登录页面)**:
   * 显示一个视频背景。
   * 包含用户登录表单（用户名和密码）。
   * 提供跳转到注册页面的链接。
2. **register.html (注册页面)**:
   * 显示一个动态粒子背景动画。
   * 包含用户注册表单（用户名和密码）。
   * 提供跳转到登录页面的链接。
3. **bookList.html (书籍列表页面)**:
   * 显示当前登录用户的用户名和注销按钮。
   * 提供文件上传区域（拖拽或选择文件）。
   * 包含上传文件的名称和描述输入框。
   * 显示上传状态提示。
   * 动态加载并显示 PDF 书籍列表，每本书籍卡片包含：
     + 文件名和描述。
     + 上传时间。
     + 在线阅读按钮。
     + 删除按钮。
     + 生成分享链接的选项（时效时长和单位）。
     + 生成后显示的分享链接和复制按钮。
4. **share.html (文件分享预览页面)**:
   * 根据 URL 中的分享 Token 获取文件信息和内容。
   * 显示文件名、描述和分享链接的剩余有效时间（或永久有效）。
   * 使用 PDF.js 在线预览 PDF 文件。
   * 提供全屏预览按钮。
   * 显示当前分享链接和复制按钮。
   * 提供下载 PDF 文件的按钮。
   * 显示加载状态或错误信息。

## 前端实现原理

前端应用主要使用原生 HTML, CSS 和 JavaScript 实现，通过 Fetch API 与后端 Web API 进行异步通信。

### 1. 用户认证和状态管理 (js/script.js)

* **原理**: 使用 JWT (JSON Web Token) 进行认证。登录成功后，后端返回 JWT Token，前端将其存储在浏览器的 localStorage 中。每次需要访问受保护的后端接口时，从 localStorage 读取 Token 并将其添加到请求的 Authorization: Bearer <Token> 头中。
* **实现**:
  + setJwtToken(token): 将 JWT Token 存储到 localStorage 中，键名为 jwtToken。
  + getJwtToken(): 从 localStorage 中读取 JWT Token。
  + removeJwtToken(): 从 localStorage 中移除 JWT Token（用于注销）。
  + isAuthenticated(): 检查 localStorage 中是否存在有效的 JWT Token，判断用户是否已认证。
  + getUsernameFromToken(): 解析 JWT Token 的 Payload 部分（Base64 解码并解析 JSON），提取用户名信息。此函数已修正，可以识别不同键名（包括完整的 URI）。
  + registerUser(username, password): 向 /api/auth/register 发送 POST 请求注册用户。
  + loginUser(username, password): 向 /api/auth/login 发送 POST 请求登录，成功后调用 setJwtToken 存储 Token 并跳转页面。
  + logoutUser(): 调用 removeJwtToken 移除 Token 并跳转到登录页面。
  + DOMContentLoaded 监听器: 在 bookList.html 页面加载时，检查用户是否已认证，如果未认证则强制跳转到登录页；如果已认证，则显示用户名并为注销按钮绑定事件。

### 2. 书籍列表页面 (bookList.html, css/list-style.css, js/list-script.js)

* **功能实现**:
  + **文件上传**:
    - HTML 结构包含一个拖拽区域 (.drop-zone) 和一个文件选择输入框 (#fileInput)。
    - js/list-script.js 监听拖拽事件 (dragover, dragleave, drop)，通过添加/移除 CSS 类 (.dragover) 来改变拖拽区域的样式。
    - 监听文件选择输入框的 change 事件，获取选中的文件。
    - handleFile(file) 函数处理选中的文件，检查文件类型（必须是 PDF），存储文件对象，并自动填充文件名输入框。
    - 点击“上传文档”按钮时，调用 uploadFile() 函数。
    - uploadFile() 函数构建 FormData 对象，包含文件名、描述和文件数据。
    - 使用 Fetch API 向 /api/BookList 发送 POST 请求，并在请求头中携带 JWT Token 进行认证。
    - 根据后端响应显示上传状态 (.status 元素)。
  + **书籍列表显示**:
    - 页面加载时 (DOMContentLoaded) 调用 fetchAndRenderBooks() 函数。
    - fetchAndRenderBooks() 函数使用 Fetch API 向 /api/BookList 发送 GET 请求，并在请求头中携带 JWT Token。
    - 成功获取书籍列表数据后，调用 renderBooks(books) 函数。
    - renderBooks(books) 函数遍历书籍数据，动态生成 HTML 结构 (.book-card)，并插入到 #bookList 容器中。
    - 为每个书籍卡片上的按钮（删除、分享）和链接（在线阅读）附加事件监听器 (attachButtonListeners())。
  + **在线阅读**:
    - 书籍卡片中的“在线阅读”按钮是一个链接 (<a> 标签)，其 href 属性直接指向后端提供文件内容的 API (/api/BookList/{filename})。
    - target="\_blank" 属性使其在新标签页中打开 PDF 文件，浏览器通常会使用内置的 PDF 阅读器或 PDF.js 等插件进行预览。
  + **删除文件**:
    - 删除按钮绑定了点击事件，调用 handleDelete(fileName) 函数。
    - handleDelete(fileName) 函数弹出确认对话框。
    - 确认删除后，使用 Fetch API 向 /api/BookList 发送 DELETE 请求，请求体中包含要删除的文件名，并在请求头中携带 JWT Token。
    - 删除成功后，重新调用 fetchAndRenderBooks() 刷新列表。
  + **生成分享链接**:
    - 每个书籍卡片包含分享时效的输入框和选择框，以及“生成分享链接”按钮。
    - 点击“生成分享链接”按钮时，调用 generateShareLink(fileName, duration, unit, bookCardElement) 函数。
    - generateShareLink() 函数获取用户选择的时效和单位，使用 Fetch API 向 /api/share/generate 发送 POST 请求，请求体包含文件名、时效和单位，并在请求头中携带 JWT Token。
    - 成功生成分享链接后，后端返回链接 URL。前端获取该 URL，显示隐藏的分享链接区域 (.share-link-area)，并将 URL 填充到输入框中。
  + **复制分享链接**:
    - 分享链接区域包含一个输入框和一个“复制”按钮。
    - 点击“复制”按钮时，调用 copyShareLink(inputElement) 函数。
    - copyShareLink() 函数尝试使用现代 navigator.clipboard.writeText API 复制文本。
    - 如果 navigator.clipboard 不可用（例如在非 HTTPS 环境），则回退到使用 document.execCommand('copy') 的备用方法。
  + **样式**: css/list-style.css 提供了书籍列表、上传区域、按钮和状态提示的样式，包括响应式调整。

### 3. 分享页面 (share.html, css/style.css (内联样式), js/share-script.js)

* **功能实现**:
  + **获取分享信息和加载 PDF**:
    - 页面加载时 (DOMContentLoaded)，从 URL 参数中获取分享 Token。
    - 调用 fetchFileInfoAndLoadPdf() 函数。
    - fetchFileInfoAndLoadPdf() 函数首先使用 Fetch API 向 /api/share/status/{token} 发送 GET 请求，获取文件信息（文件名、描述、过期时间）。此接口允许匿名访问。
    - 成功获取文件信息后，显示文件名和描述，并调用 updateRemainingTime() 更新剩余时间显示。
    - 如果分享不是永久有效，启动定时器 (setInterval) 每秒更新剩余时间。
    - 然后，使用 Fetch API 向 /api/share/file/{token} 发送 GET 请求获取 PDF 文件的 Blob 数据。此接口也允许匿名访问。
    - 获取到 Blob 数据后，使用 PDF.js 库 (pdfjsLib.getDocument()) 加载 PDF。
    - 加载成功后，遍历 PDF 的每一页，创建 Canvas 元素并使用 PDF.js 渲染页面内容，将其添加到 #pdfViewer 容器中。
    - 显示全屏预览按钮。
    - 在整个过程中，使用 showLoading() 和 showError() 显示状态和错误信息。
  + **剩余时间倒计时**:
    - updateRemainingTime(expirationTimeUtc) 函数计算当前时间与过期时间之间的差值，并格式化为“天、小时、分、秒”的形式显示在 #remainingTime 元素中。
    - 特殊处理永久有效的分享链接。
    - 如果分享过期，显示过期消息并调用 showError()。
  + **全屏预览**:
    - 全屏按钮绑定点击事件，调用 toggleFullscreen() 函数。
    - toggleFullscreen() 函数使用 Element.requestFullscreen() API (及其浏览器前缀版本) 来切换 #pdfViewer 元素的全屏状态。
    - CSS (#pdfViewer:fullscreen 等伪类) 调整全屏时的布局，使用 Flexbox 将 PDF 内容居中显示。
  + **复制分享链接**:
    - 分享页面URL本身就是分享链接。
    - 复制按钮绑定点击事件，调用 copyCurrentShareLink() 函数。
    - copyCurrentShareLink() 函数获取当前页面的 URL (window.location.href)，并使用与书籍列表页面相同的逻辑（优先使用 Clipboard API，回退到 execCommand）复制到剪贴板。
  + **下载 PDF**:
    - 下载按钮绑定点击事件，调用 handleDownload() 函数。
    - handleDownload() 函数构建指向 /api/share/file/{token} 的下载 URL。
    - 创建一个临时的 <a> 元素，设置 href 为下载 URL，download 属性为文件名。
    - 将临时 <a> 元素添加到页面，模拟点击触发下载，然后移除该元素。
  + **样式**: share.html 中的内联 CSS 和 css/style.css 共同控制页面的布局和样式，包括响应式调整和全屏时的内容居中。

### 4. 登录/注册页面 (main.html, register.html, css/style.css, css/canvas.css, js/script.js, js/canvas.js)

* **功能实现**:
  + **登录/注册表单**:
    - HTML 包含标准的表单结构 (.auth-form) 和输入框 (input[type="text/password"])。
    - js/script.js 在 DOMContentLoaded 中为登录和注册表单绑定 submit 事件监听器。
    - registerUser() 和 loginUser() 函数分别处理表单提交，向后端认证 API 发送请求。
  + **背景视频**:
    - main.html 使用 <video> 标签嵌入视频，.video-container 提供响应式布局。
  + **背景粒子动画**:
    - register.html 使用 <canvas> 元素作为背景。
    - js/canvas.js 在 DOMContentLoaded 中初始化 Canvas 和粒子系统。
    - Particle 类定义粒子的属性（位置、大小、颜色、速度等）。
    - initParticles() 创建粒子数组。
    - connectParticles() 绘制粒子之间的连线。
    - animate() 函数是动画循环，清空 Canvas（使用半透明填充实现拖尾效果），更新粒子位置，绘制粒子和连线，并使用 requestAnimationFrame 实现平滑动画。
    - 监听鼠标移动和窗口大小改变事件，更新鼠标位置和重置粒子系统。
  + **样式**: css/style.css 提供通用的基础样式、表单样式和响应式布局。css/canvas.css 提供 Canvas 的固定定位和尺寸样式，确保其作为背景。

## 总结

整个项目通过 ASP.NET Core Web API 提供后端服务，负责文件存储、用户认证和分享链接管理。前端使用纯 HTML, CSS 和 JavaScript 构建用户界面，通过异步请求与后端 API 交互。JWT 用于用户认证，确保只有登录用户才能访问书籍列表和生成分享链接等功能。分享功能通过生成唯一的 Token 实现匿名访问和时效控制。PDF.js 库在前端负责 PDF 文件的加载和渲染。