****

**WEB 技术**

**实验报告**

|  |  |
| --- | --- |
| 学生姓名 | 张子洋 |
| 学 号 | 808221223 |
| 专业班级 | 计科2203班 |
| 指导教师 | 盛羽 |
| 学 院 | 计算机学院 |
| 完成时间 | 2024.12.10 |

计算机学院

2024年7月

**目录**

[一 实践任务描述 4](#_Toc17284)

[二 项目文件结构及功能 6](#_Toc32493)

[三 页面效果展示 7](#_Toc27975)

[四 页面设计图 10](#_Toc26666)

[五 核心技术点 11](#_Toc26666)

[5.1 CSS设计 11](#_Toc21345)

[5.1.1 响应式设计 11](#_Toc13576)

[5.1.2 导航栏样式 11](#_Toc16733)

[5.1.3 卡片设计 11](#_Toc30451)

[5.1.4 动态交互效果 11](#_Toc15135)

[5.2 ECharts图表 11](#_Toc8637)

[5.2.1 ECharts折线图 11](#_Toc28338)

[5.2.2 ECharts雷达图 12](#_Toc30424)

[5.3 高德api 13](#_Toc32002)

[5.3.1 天气插件集成： 13](#_Toc2981)

[5.3.2 动态标记和信息窗口： 13](#_Toc17802)

[5.3.3 互动性： 13](#_Toc19167)

[5.3.4 美学和功能特点： 14](#_Toc13441)

[5.4 SQ lite3数据库 15](#_Toc4776)

[5.4.1 数据库设置和表创建： 15](#_Toc20287)

[5.4.2 使用 bcrypt 进行密码哈希处理： 15](#_Toc23496)

[5.4.3 数据插入： 16](#_Toc22768)

[5.4.4 数据库连接清理： 16](#_Toc31774)

[5.4.5 安全功能（散列密码）： 16](#_Toc31192)

[5.5 multer和localstorage实现编辑 18](#_Toc25934)

[5.5.1 Multer（文件上传和处理） 18](#_Toc26031)

[5.5.2 LocalStorage（数据持久化与交互） 19](#_Toc2612)

[5.6 Session 路由鉴权 25](#_Toc8668)

[5.6.1 Session 身份验证: 25](#_Toc10409)

[5.6.2 鉴权中间件 requireAuth: 25](#_Toc21153)

[5.6.3 登录功能和密码加密: 26](#_Toc25233)

[5.6.4 增加安全性: 26](#_Toc28265)

[六 心得与体会 27](#_Toc10327)

[6.1 环境配置 27](#_Toc28895)

[6.2 技术提升与收获 27](#_Toc18490)

[6.3 前端与后端协作 27](#_Toc13184)

[6.4 不足与改进 27](#_Toc20764)

[参考资料 29](#_Toc32686)

# 一 实践任务描述

## 实验一：HTML+CSS实验

制作个人主页，要求：

1)符合HTML,CSS相关规范；

2)网页内容及布局不限（可参考下图）;

3)不能使用任何框架

## 实验二：JavaScript 实验

在个人主页中增加以下内容：

1)在合适的位置显示家乡当前天气现象

2)直接调用高德Web服务API进行实现

## 实验三：后端程序设计实验

在个人主页中增加以下内容：

1)支持网页访问计数

2)在合适的位置放置编辑链接

3)点击编辑链接进入密码验证界面（密码可预先存于数据库）

4)密码验证通过后进入个人主页内容修改页面

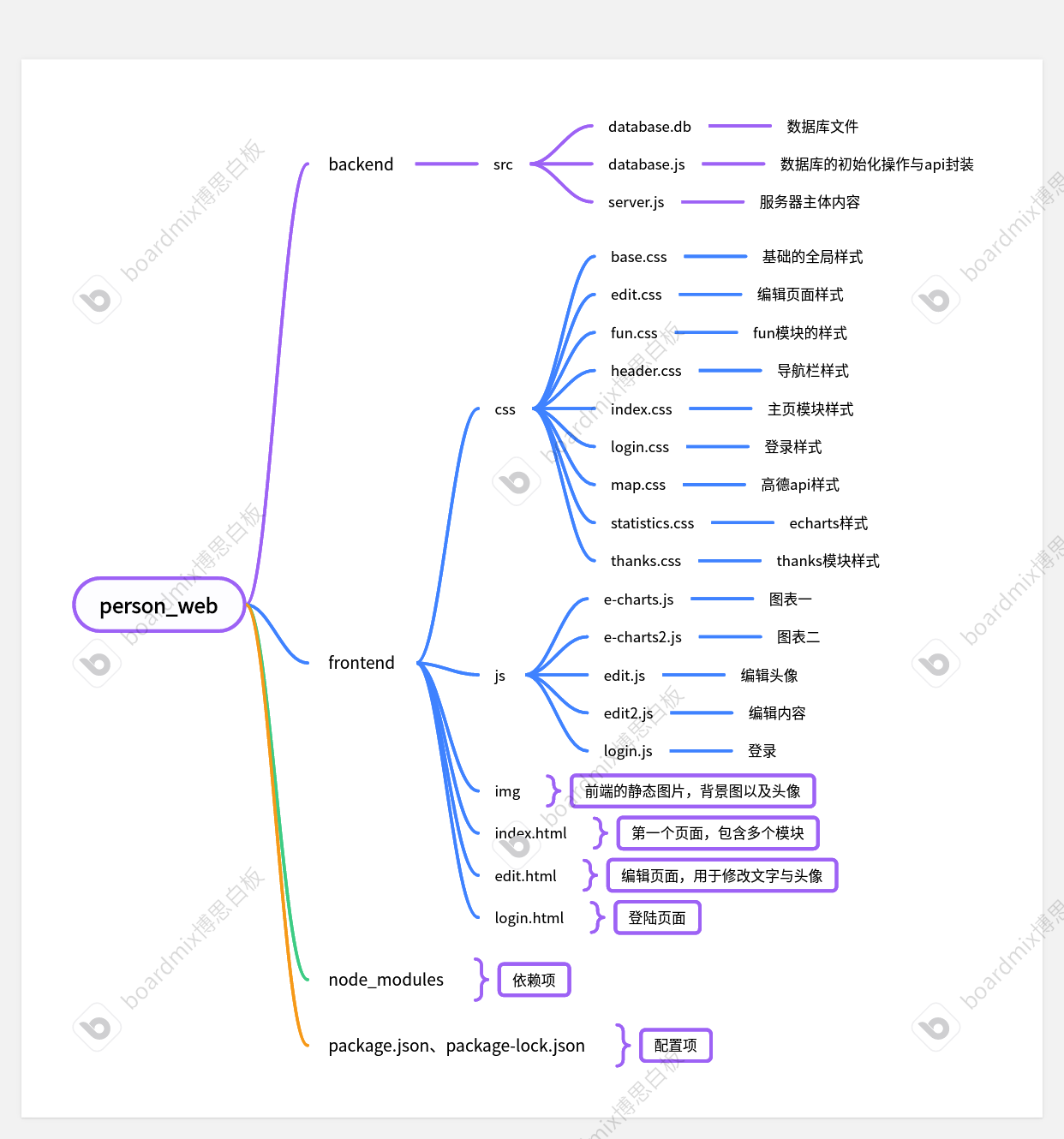
5)个人主页中可修改的字段根据个人主页内容自定义（如：电话、项目经历等），适当选择有代表性字段即可

6)有一个可编辑字段需为列表式的，如项目经历，提供相应的添加、删除、修改操作支持

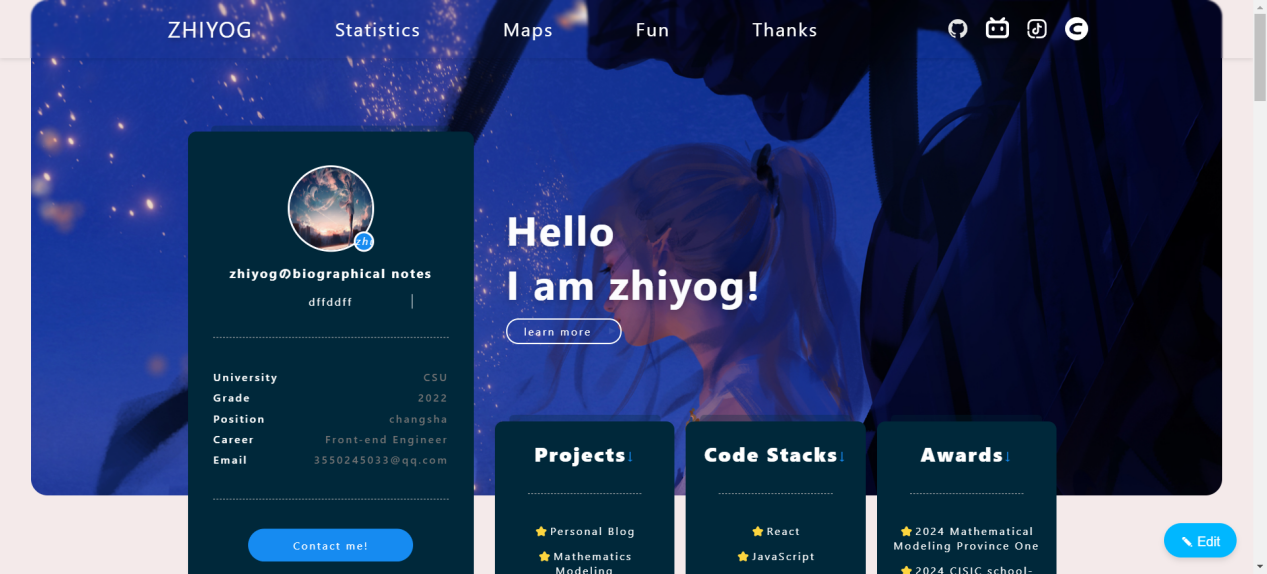
7)支持更改照片

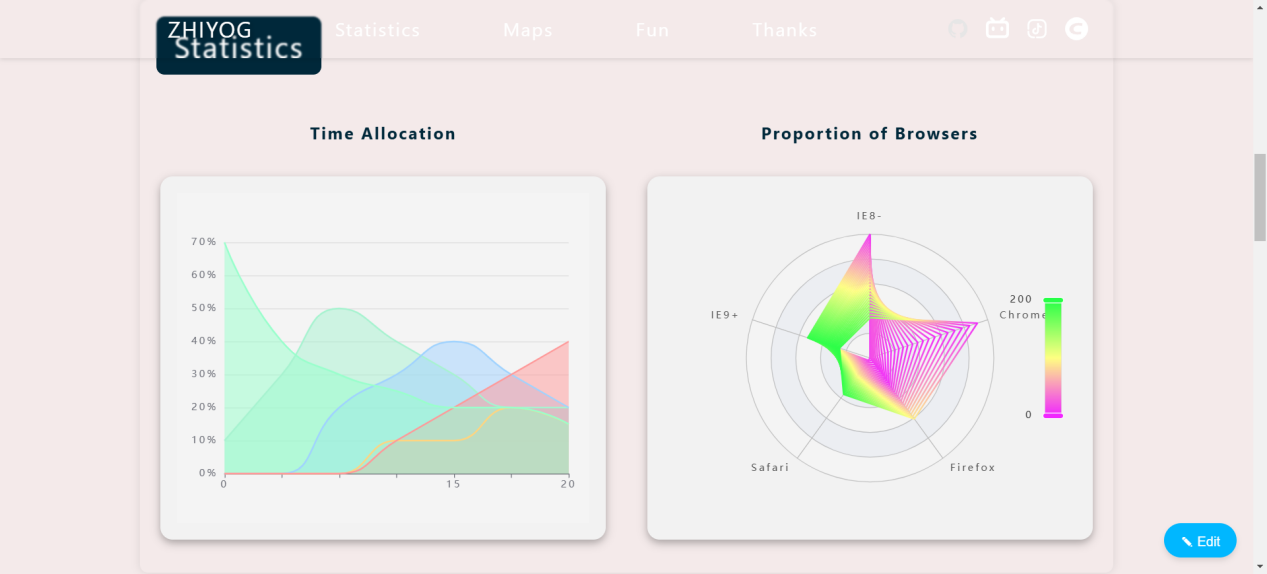
要求：不能使用框架（含JQuery）

# 二 项目文件结构及功能

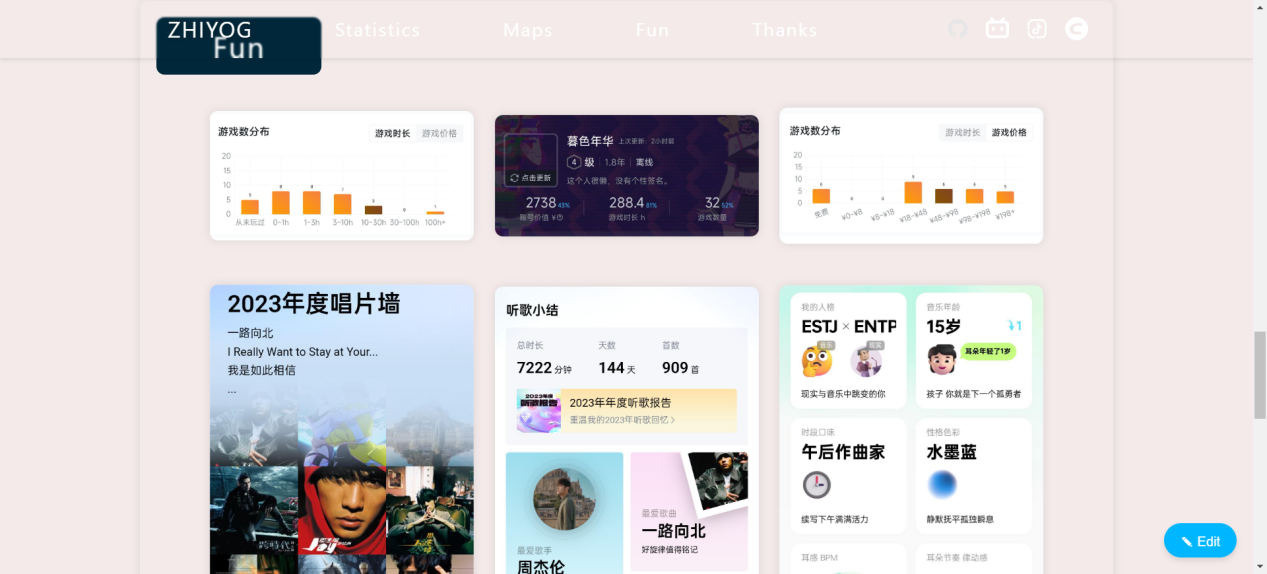
****

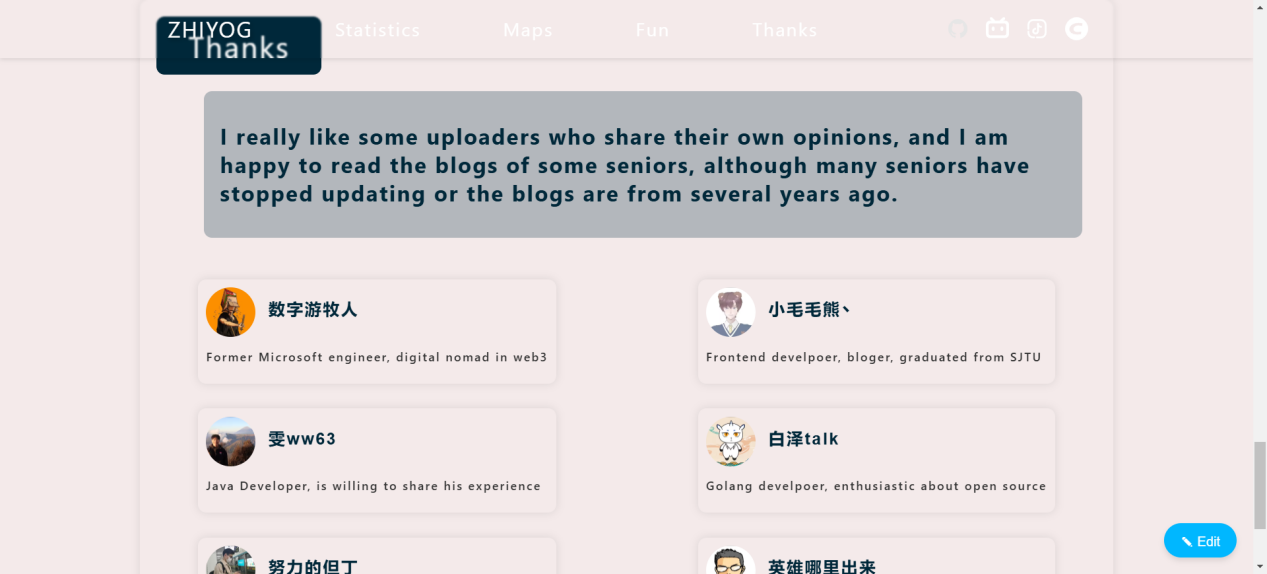
# 三 页面效果展示

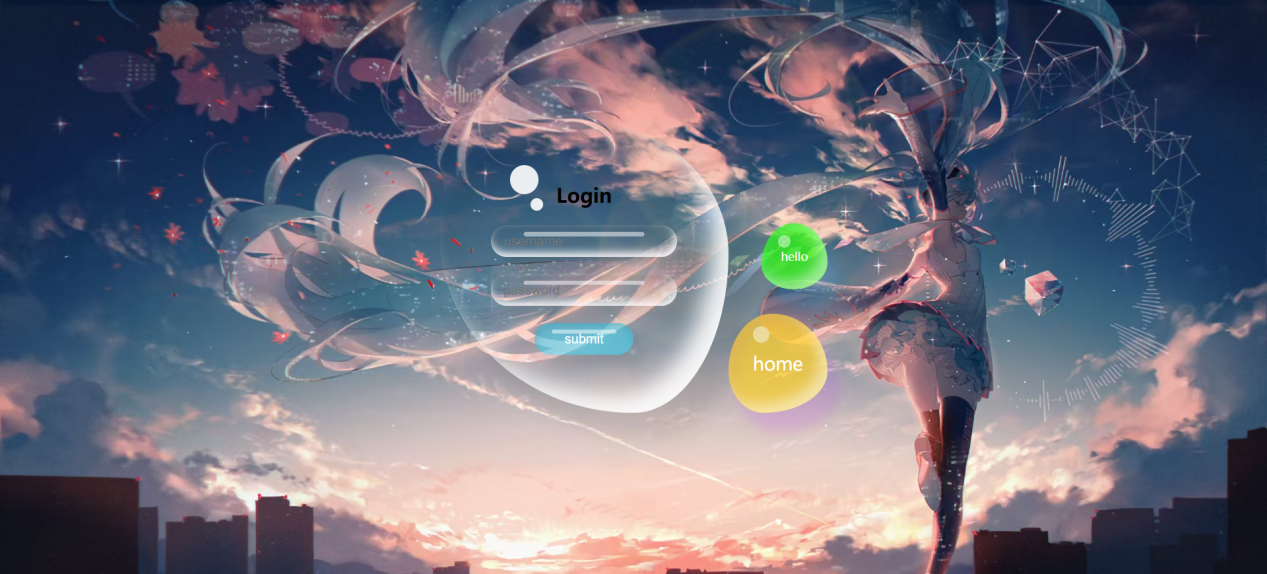


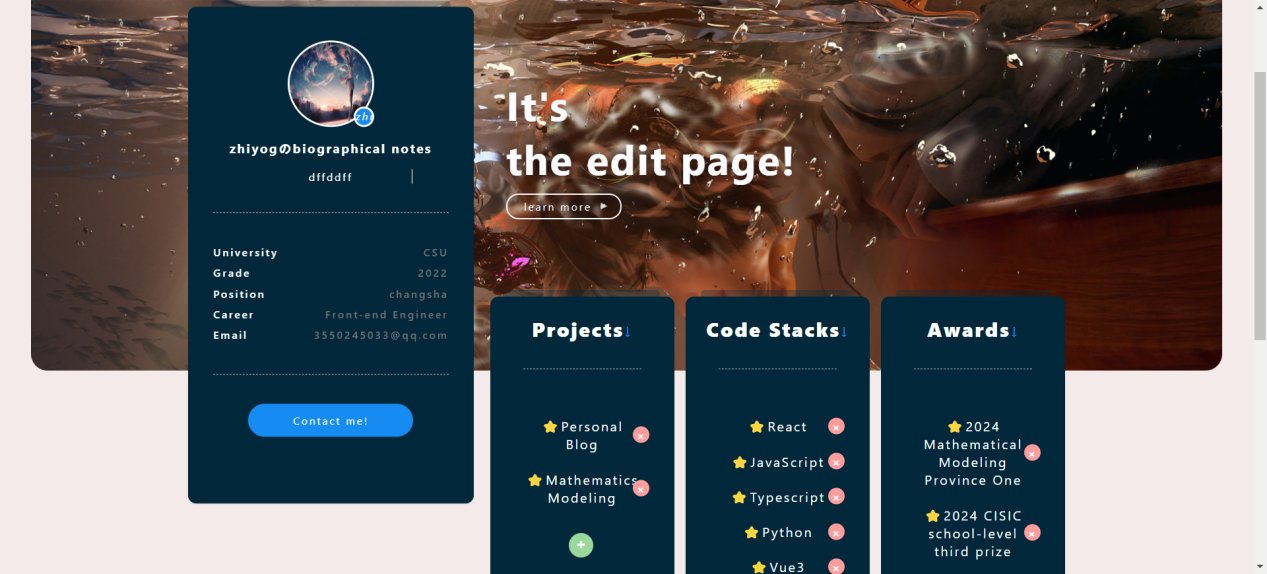




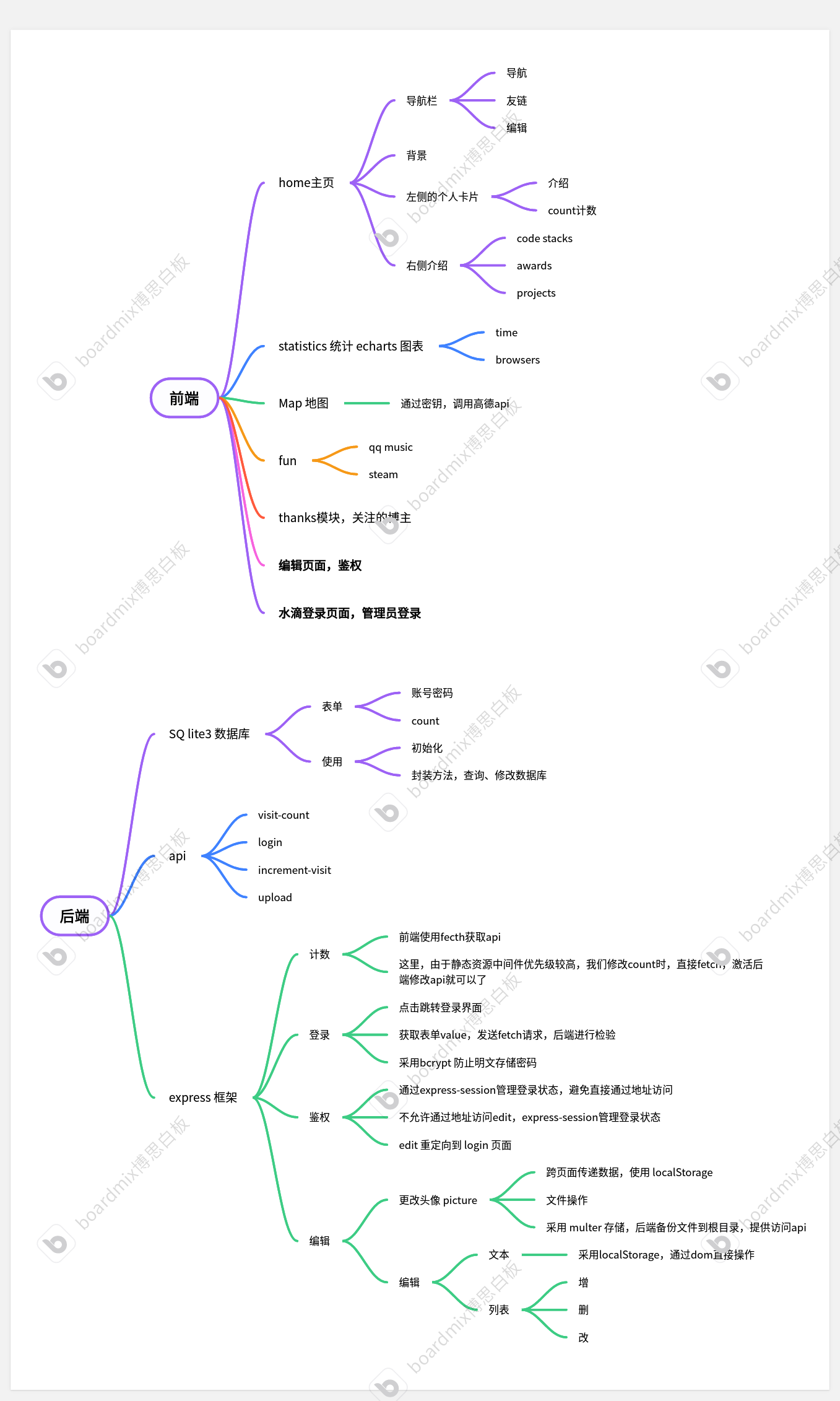








# 四 页面设计图

五 核心技术点

## 5.1 CSS设计

### ****5.1.1 响应式设计****

* 通过 @media 媒体查询调整了不同屏幕尺寸的样式。例如，在手机和小屏设备上调整了导航栏、布局宽度等元素，以确保页面适应各种设备。

### ****5.1.2 导航栏样式****

* .navigation 类定义了背景图像、固定背景、标题样式等，使用了 background-image 和 background-size 属性来确保背景图像在不同屏幕尺寸下的展示效果。
* .navigation .buttom 通过 :hover 增加了按钮的放大效果。

### ****5.1.3 卡片设计****

* .me-card 和 .carbox 提供了卡片和容器的基础样式，包括背景色、阴影、圆角等样式，增强了视觉效果。
* 使用了 box-shadow、border-radius 和 transform 属性来优化视觉体验。

### ****5.1.4 动态交互效果****

* @keyframes 动画在多个地方被使用，如 bmove 和 shine，分别用于按钮指示和卡片的动画效果，增加了页面的动感。
* hover 效果被广泛应用于元素的样式变化，例如按钮和图片的交互。

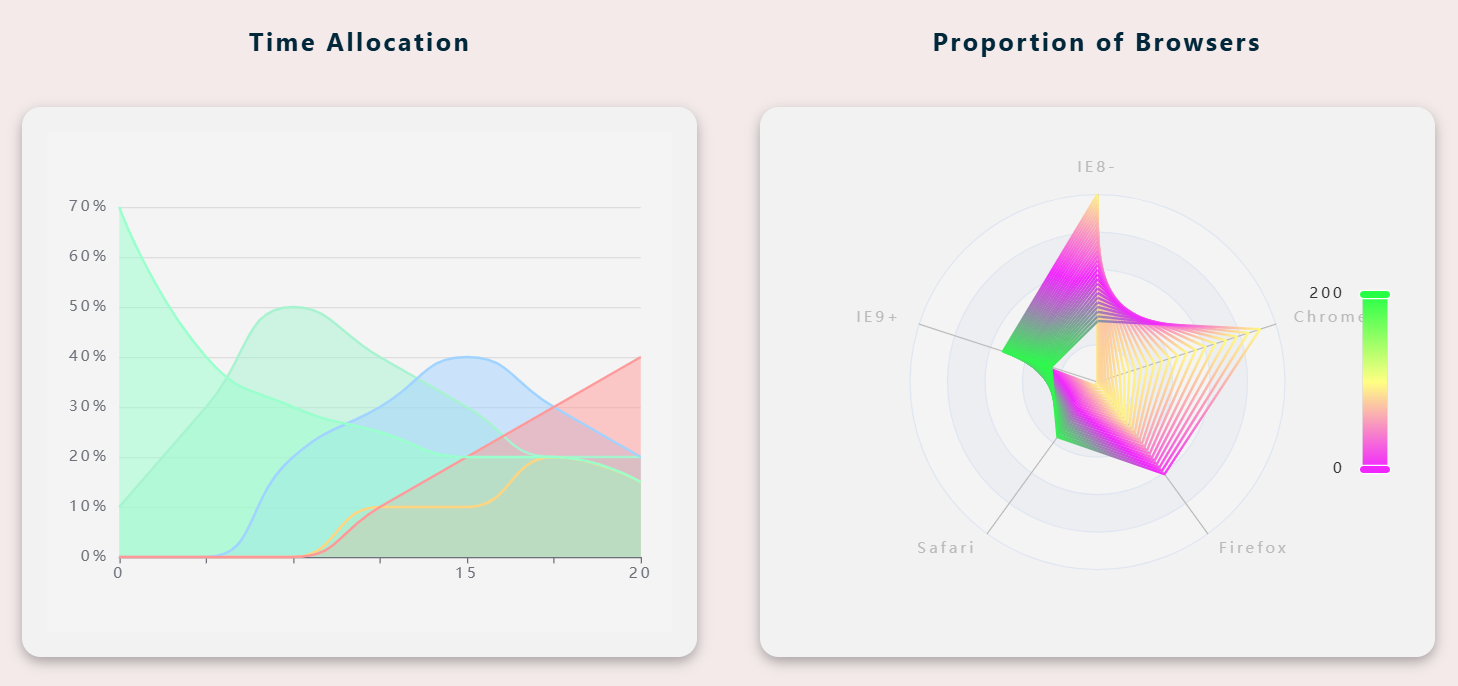
## 5.2 ECharts图表

### 5.2.1 ECharts****折线图****

* **数据设置：**各种活动（学习、音乐、游戏、编码、家庭）的时间分配以及不同时间指标的对应值。
* **类别和颜色：**定义类别（活动）并为每个类别分配不同的颜色，以便更好地进行视觉区分。
* **平滑线条：**每个系列（活动）都用一条平滑的线条表示，并且线下方的区域用半透明颜色填充，以实现平滑的渐变效果。
* **自定义轴标签：** x 轴代表时间指标，标签仅显示 5 的倍数的值。
* **工具提示：**将鼠标悬停在图表点上时，工具提示会显示详细信息，显示特定指数下每个类别的值。
* **响应式设计：**图表设置为根据窗口大小动态调整大小，以在不同的屏幕尺寸上保持其布局。

### 5.2.2 ECharts****雷达图****

* **视觉地图：**此功能设置从绿色到黄色再到紫色的颜色渐变来表示数据值。视觉地图有助于将数值映射到颜色渐变上。
* **雷达指标：**雷达图使用五个指标，每个指标对应一个特定的浏览器（IE，Safari，Firefox，Chrome）。
* **渐变和强调：**雷达图中的数据线呈现渐变效果。当鼠标悬停在数据线上时，线条的宽度和颜色会发生变化，区域会填充半透明颜色。
* **系列生成：**动态生成 28 个数据系列，每个系列代表五个指标的一组值，并且每个系列都有独特的颜色渐变。
* **动态数据生成：**使用模式（例如，减少或增加函数）生成数据值，从而使图表具有不断发展、变化的性质。



## 5.3 高德api

### 5.3.1 ****天气插件集成：****

* **AMap.Weather插件：**该脚本使用AMap Weather插件（AMap.plugin('AMap.Weather', function () {...}）来获取实时天气数据。
* **天气数据检索：**该weather.getLive()方法获取指定城市（本例中为温县）的实时天气数据。
* **天气信息显示：**脚本提取并显示各种天气详细信息。

### 5.3.2 ****动态标记和信息窗口：****

* **创建标记：**使用自定义图标（蓝色标记）在地图中心放置一个标记，并使用偏移量（new AMap.Pixel(-13, -30)）调整位置。
* **信息窗口：**AMap.InfoWindow当用户与标记交互时，创建一个来显示天气信息。
  + 信息窗口的内容是根据天气数据动态生成的。
  + 当标记初始化或悬停在标记位置上时，信息窗口会在地图上打开。

### ****5.3.3 互动性：****

* **标记悬停事件：**当用户将鼠标悬停在标记上时，天气信息会显示在信息窗口中。
  + 该事件marker.on('mouseover', function () {...})用于在标记悬停时打开信息窗口。

### 5.3.4 ****美学和功能特点：****

* **自定义标记图标：**使用自定义标记图标 ( https://webapi.amap.com/theme/v1.3/markers/n/mark\_b.png) 来表示地图上的位置。
* **动态和信息内容：**信息窗口包含动态内容，以清晰的格式（使用 进行结构化布局）向用户提供详细的天气<h4>信息<p>。

    <!-- 地图 -->

    <div class="maps" id="maps">

      <div class="mode">Maps</div>

      <div class="map" id="map"></div>

      <script type="text/javascript">

        window.\_AMapSecurityConfig = {

          securityJsCode: "89b596490cca6364101b36c2d45e9f3e",

        }

      </script>

      <script src="https://webapi.amap.com/loader.js"></script>

      <script type="text/javascript"

        src="https://webapi.amap.com/maps?v=2.0&key=dbcb618758ba071072471d12ea02dcb8"></script>

      <script type="text/javascript">

        var map = new AMap.Map('map', { // 修改这里为 'map'

          resizeEnable: true,

          center: [113.0795, 34.9412],

          zoom: 12

        });

        AMap.plugin('AMap.Weather', function () {

          var weather = new AMap.Weather();

          // 查询实时天气信息

          weather.getLive('温县', function (err, data) {

            if (!err) {

              var str = [];

              str.push('<h4>实时天气</h4><hr>');

              str.push('<p>城市/区：' + data.city + '</p>');

              str.push('<p>天气：' + data.weather + '</p>');

              str.push('<p>温度：' + data.temperature + '℃</p>');

              str.push('<p>风向：' + data.windDirection + '</p>');

              str.push('<p>风力：' + data.windPower + ' 级</p>');

              str.push('<p>空气湿度：' + data.humidity + '</p>');

              str.push('<p>发布时间：' + data.reportTime + '</p>');

              var marker = new AMap.Marker({

                map: map,

                position: map.getCenter(),

                icon: 'https://webapi.amap.com/theme/v1.3/markers/n/mark\_b.png', // 默认蓝色标记图标

                offset: new AMap.Pixel(-13, -30) // 偏移量调整，使图标中心对准位置

              });

              var infoWin = new AMap.InfoWindow({

                content: '<div class="info">' + str.join('') + '</div>',

                isCustom: true,

              });

              infoWin.open(map, marker.getPosition());

              marker.on('mouseover', function () {

                infoWin.open(map, marker.getPosition());

              });

            }

          });

        });

      </script>

    </div>

## 5.4 SQ lite3数据库

### ****5.4.1 数据库设置和表创建：****

* sqlite3模块用于初始化 SQLite 数据库（database.db）。
* 该users表包含、和的列id，username以password确保该表可用于将来的用户身份验证或注册。

### 5.4.2 ****使用 bcrypt 进行密码哈希处理：****

* 该bcrypt库用于在将密码存储到数据库之前对其进行安全哈希处理。这对于确保用户密码安全存储而非以明文形式存储至关重要。
* 该bcrypt.hash方法使用 10 轮盐值来为初始admin用户生成散列密码。

### ****5.4.3 数据插入：****

* 散列密码被插入到users表中，确保敏感数据（如密码）安全存储。
* 该INSERT INTO users查询确保如果用户已经存在（使用INSERT OR IGNORE），则不会插入重复的记录。

### ****5.4.4 数据库连接清理：****

* 完成操作后（成功或错误后），数据库连接将关闭，以避免资源泄漏。

### 5.4.5 ****安全功能（散列密码）：****

* 存储密码的最佳实践，即在使用 存储之前对密码进行哈希处理bcrypt，这使应用程序更加安全，并防止以不安全的格式存储敏感信息。

const sqlite3 = require('sqlite3').verbose();

const db = new sqlite3.Database('database.db');

const bcrypt = require('bcrypt');

// 初始化数据库

// db.serialize(() => {

//   // 创建 users 表

//   db.run(`

//     CREATE TABLE IF NOT EXISTS users (

//       id INTEGER PRIMARY KEY AUTOINCREMENT,

//       username TEXT NOT NULL UNIQUE,

//       password TEXT NOT NULL

//     )

//   `);

//   // 插入默认用户

//   db.run(`

//     INSERT OR IGNORE INTO users (username, password)

//     VALUES ('admin', '123456') -- 修改为你需要的初始用户名和密码

//   `);

//   // 创建 visit\_counts 表

//   db.run(`

//     CREATE TABLE IF NOT EXISTS visit\_counts (

//       id INTEGER PRIMARY KEY AUTOINCREMENT,

//       count INTEGER NOT NULL DEFAULT 0

//     )

//   `);

//   // 初始化访问计数

//   db.run(`

//     INSERT OR IGNORE INTO visit\_counts (count)

//     VALUES (0)

//   `);

// });

 // 加密密码

 // 初始化数据库

 db.serialize(() => {

   // 创建用户表

   db.run(`

     CREATE TABLE IF NOT EXISTS users (

       id INTEGER PRIMARY KEY AUTOINCREMENT,

       username TEXT UNIQUE NOT NULL,

       password TEXT NOT NULL

     )

   `);

   // 加密密码

   const username = 'admin';

   const plainPassword = 'admin123';

   bcrypt.hash(plainPassword, 10, (err, hashedPassword) => {

     if (err) {

       console.error('密码加密失败:', err);

       db.close(); // 在发生错误时也需要关闭数据库

       return;

     }

     // 插入初始化数据

     db.run(

       `INSERT INTO users (username, password) VALUES (?, ?)`,

       [username, hashedPassword],

       (err) => {

         if (err) {

           console.error('数据插入失败:', err);

         } else {

           console.log(`用户 ${username} 数据初始化完成`);

         }

         // 所有操作完成后关闭数据库

         db.close();

       }

     );

   });

 });

## 5.5 multer和localstorage实现编辑

### 5.5.1 ****Multer（文件上传和处理）****

**① 使用 Multer 上传文件**：

* **Multer 集成**：代码利用**Multer**（虽然您提供的代码中没有明确显示，但它通过调用fetch和服务器端处理暗示）来处理服务器上的文件上传。此库对于处理通常用于上传文件的 multipart/form-data 至关重要。
* **图片处理**：用户可以上传头像图片（头像），由服务器处理后保存到服务器的特定路径或云存储服务中。

**② 服务器端交互**：

* **图像上传**：通过元素选择图像文件后<input type="file">，POST将向服务器端点（/upload）发出文件请求。Multer 将处理服务器端的文件解析和存储。
* **服务器响应**：然后服务器以包含新文件路径的 JSON 对象进行响应，然后在客户端使用该对象来更新头像图像。

**③ 持久图像路径**：

* **动态路径处理**：文件上传成功后，服务器返回新的图片路径（filePath），该路径动态设置为头像的图片源（editHeadPicture.src）。
* **错误处理**：如果上传失败，代码会处理错误并提醒用户，从而确保上传过程的稳定性。

### 5.5.2 LocalStorage（数据持久化与交互）

**① 头像图片持久存储**：

* **LocalStorage 持久化**：头像图片上传完成后，服务器返回图片路径后，会将新的图片路径存储在浏览器的 中localStorage。这样可以保证即使刷新或重新访问页面，头像图片也能持久化，无需重新上传。
* **高效存储**：头像图像路径作为字符串存储在localStorage键下"headPictureSrc"，可轻松跨会话检索。

**② 使用 LocalStorage 编辑文本**：

* **可编辑文本元素**：editableText和quoteText元素允许用户单击并编辑文本。新文本存储在 中localStorage，即使页面重新加载后仍会保留。
* **动态文本编辑**localStorage：当用户更新文本时，文本会在用户确认更改后立即保存，确保更新的内容在会话中持久保存。

**③ 高效的列表处理**：

* **在 LocalStorage 中存储列表**：类别（如“项目”、“codeStacks”和“奖项”）以 JSON 数组形式存储在 中localStorage。这可确保列表数据不会在会话之间丢失。
* **添加、编辑和删除项目**：该应用允许用户添加、编辑和删除这些列表中的项目。每个更改都会反映出来，localStorage以便数据保持持久性。

**④ 加载和渲染**：

* **页面加载时加载数据**：加载页面时，localStorage将检索并相应地呈现存储在其中的数据（如头像图像路径和可编辑文本）。这使用户体验无缝衔接，并确保他们不会丢失之前的设置或修改。
* **初始回退**：如果未找到任何数据localStorage，则使用默认值（例如占位符图像），以确保即使用户之前未与页面交互，页面也能按预期运行。

|  |  |  |
| --- | --- | --- |
| 特征 | **Multer** | **Localstorage** |
| 数据类型 | 文件上传（例如图像、文档） | 文本数据（字符串、JSON 数组等） |
| 持久性 | 服务器端存储（图片路径、文件） | 客户端持久性（数据保存在浏览器中） |
| 用例 | 上传和管理文件（图片上传等） | 存储简单数据（文本、列表、偏好） |
| 数据可用性 | 需要服务器来存储和提供文件 | 浏览器中可获取未来所有访问的数据 |
| 安全 | 需要服务器端安全性（例如文件验证、权限） | 仅限于客户端安全（无服务器端验证） |
| 错误处理 | 处理与文件上传失败相关的错误（大小、格式等） | 错误与 localStorage 容量和浏览器支持有关 |
| 实施复杂性 | 需要后端设置（例如，带有 Express 的 Node.js、Multer） | 无需后端，完全由前端处理 |

// 获取span元素

const editableText = document.getElementById('edit\_hover\_text');

// 从localStorage读取并设置初始文本（如果有保存的内容）

if (localStorage.getItem('textContent')) {

    editableText.textContent = localStorage.getItem('textContent');

}

// 添加点击事件，允许用户修改文本

editableText.addEventListener('click', () => {

    const currentText = editableText.textContent;

    const newText = prompt('编辑文本:', currentText);

    if (newText !== null && newText !== currentText) {

        editableText.textContent = newText;

        // 将修改后的文本保存到localStorage

        localStorage.setItem('textContent', newText);

    }

});

const quoteText = document.getElementById('edit\_quote')

if (localStorage.getItem('textQuote')) {

    quoteText.textContent = localStorage.getItem('textQuote');

}

quoteText.addEventListener('click', () => {

    const currentText = quoteText.textContent;

    const newText = prompt('编辑文本:', currentText);

    if (newText !== null && newText !== currentText) {

        quoteText.textContent = newText;

        // 将修改后的文本保存到localStorage

        localStorage.setItem('textQuote', newText);

    }

});

// 加载列表数据并渲染

function loadList() {

    const categories = ['projects', 'codeStacks', 'awards'];

    categories.forEach(category => {

        const list = JSON.parse(localStorage.getItem(category)) || initialData[category];

        const listContainer = document.getElementById(`${category}-list`);

        listContainer.innerHTML = ''; // 清空现有列表

        list

            .filter(item => typeof item === 'string') // 筛选出字符串类型的数据

            .forEach(item => {

                const p = document.createElement('p');

                p.textContent = item; // 确保 item 是字符串

                p.onclick = () => editItem(category, p, item);

                const deleteBtn = document.createElement('span');

                deleteBtn.classList.add('delete-btn');

                deleteBtn.textContent = '×';

                deleteBtn.onclick = (e) => {

                    e.stopPropagation();

                    deleteItem(category, p, item);

                };

                p.appendChild(deleteBtn);

                listContainer.appendChild(p);

            });

    });

}

// 编辑条目

function editItem(category, p, oldText) {

    customPrompt('Edit item:', p.textContent.replace('×', '').trim(), (newText) => {

        if (newText !== null && newText !== oldText) {

            p.textContent = newText;

            // 添加删除按钮

            const deleteBtn = document.createElement('span');

            deleteBtn.classList.add('delete-btn');

            deleteBtn.textContent = '×';

            deleteBtn.onclick = (e) => { e.stopPropagation(); deleteItem(category, p, newText); };

            p.appendChild(deleteBtn);

            // 更新 localStorage 数据

            const list = JSON.parse(localStorage.getItem(category)) || initialData[category];

            const index = list.indexOf(oldText);

            if (index > -1) {

                list[index] = newText;

                localStorage.setItem(category, JSON.stringify(list));

            }

        }

    });

}

function deleteItem(category, element, item) {

    const list = JSON.parse(localStorage.getItem(category)) || initialData[category];

    const updatedList = list.filter(entry => entry !== item); // 移除匹配的项

    localStorage.setItem(category, JSON.stringify(updatedList)); // 更新 localStorage

    element.remove(); // 从 DOM 中移除对应的 <p>

}

// 增加新条目

function addItem(category) {

    customPrompt('Enter new item:', '', (newText) => {

        if (newText) {

            const list = JSON.parse(localStorage.getItem(category)) || initialData[category];

            list.push(newText);

            localStorage.setItem(category, JSON.stringify(list));

            loadList();

        }

    });

}

function customPrompt(title, defaultValue, callback) {

    // 创建遮罩层

    const overlay = document.createElement('div');

    overlay.style.position = 'fixed';

    overlay.style.top = '0';

    overlay.style.left = '0';

    overlay.style.width = '100vw';

    overlay.style.height = '100vh';

    overlay.style.backgroundColor = 'rgba(0, 0, 0, 0.5)';

    overlay.style.zIndex = '9998';

    // 创建弹窗容器

    const promptBox = document.createElement('div');

    promptBox.style.position = 'fixed';

    promptBox.style.top = '50%';

    promptBox.style.left = '50%';

    promptBox.style.transform = 'translate(-50%, -50%)';

    promptBox.style.width = '300px';

    promptBox.style.padding = '20px';

    promptBox.style.backgroundColor = '#fff';

    promptBox.style.boxShadow = '0 2px 10px rgba(0, 0, 0, 0.2)';

    promptBox.style.borderRadius = '8px';

    promptBox.style.zIndex = '9999';

    promptBox.style.fontFamily = 'Arial, sans-serif';

    // 标题

    const titleEl = document.createElement('h3');

    titleEl.textContent = title;

    titleEl.style.margin = '0 0 10px';

    titleEl.style.fontSize = '18px';

    titleEl.style.color = '#333';

    // 输入框

    const input = document.createElement('input');

    input.type = 'text';

    input.value = defaultValue || '';

    input.style.width = '80%';

    input.style.padding = '10px';

    input.style.marginBottom = '10px';

    input.style.border = '1px solid #ddd';

    input.style.borderRadius = '4px';

    input.style.fontSize = '14px';

    // 按钮容器

    const buttonContainer = document.createElement('div');

    buttonContainer.style.textAlign = 'right';

    // 确认按钮

    const confirmButton = document.createElement('button');

    confirmButton.textContent = '确认';

    confirmButton.style.marginRight = '10px';

    confirmButton.style.padding = '8px 12px';

    confirmButton.style.border = 'none';

    confirmButton.style.borderRadius = '4px';

    confirmButton.style.backgroundColor = '#007bff';

    confirmButton.style.color = '#fff';

    confirmButton.style.cursor = 'pointer';

    // 取消按钮

    const cancelButton = document.createElement('button');

    cancelButton.textContent = '取消';

    cancelButton.style.padding = '8px 12px';

    cancelButton.style.border = 'none';

    cancelButton.style.borderRadius = '4px';

    cancelButton.style.backgroundColor = '#6c757d';

    cancelButton.style.color = '#fff';

    cancelButton.style.cursor = 'pointer';

    // 按钮点击事件

    confirmButton.addEventListener('click', () => {

      const result = input.value.trim();

      if (callback) callback(result);

      document.body.removeChild(promptBox);

      document.body.removeChild(overlay);

    });

    cancelButton.addEventListener('click', () => {

      if (callback) callback(null);

      document.body.removeChild(promptBox);

      document.body.removeChild(overlay);

    });

    // 组装元素

    buttonContainer.appendChild(confirmButton);

    buttonContainer.appendChild(cancelButton);

    promptBox.appendChild(titleEl);

    promptBox.appendChild(input);

    promptBox.appendChild(buttonContainer);

    document.body.appendChild(overlay);

    document.body.appendChild(promptBox);

    // 自动聚焦输入框

    input.focus();

  }

// 初始化加载页面内容

loadList();

## 5.6 Session 路由鉴权

### ****5.6.1 Session 身份验证****:

①express-session 中间件：

* 用来存储用户会话数据，确保用户在登录后能够维持会话状态，而不需要每次请求都重新登录。
* 配置了一个 secret 字段，确保会话数据被加密并且保持私密。
* cookie 设置了会话的有效期为 **1小时**，即每个用户登录后的会话会持续1小时。
* resave: false 和 saveUninitialized: false 确保不会在每次请求时强制重新保存会话。

② session.user:

* 登录成功后，将用户信息（如 username）存储在 session 中，标识该用户已登录。
* 这样，在用户访问需要认证的页面时，可以通过会话判断用户是否已登录。

### ****5.6.2 鉴权中间件**** requireAuth:

①requireAuth 中间件用于对特定的路由进行访问控制

* 如果用户的 session.user 不存在（即未登录），则会重定向到登录页面 (/login.html)。
* 如果用户已登录（session.user 存在），则允许访问后续的路由。

**② 保护页面**:

* 例如，/edit.html 页面使用了 requireAuth，确保只有登录用户才能访问。
* 这种方法确保了对于敏感操作或编辑页面，未登录的用户无法直接访问。

### ****5.6.3 登录功能和密码加密****:

**① 登录 API (**/api/login**)**：

* 使用 **bcrypt** 对密码进行加密处理和比对，确保用户密码的安全性。
* 登录成功后，将用户信息存储在 session 中，以后请求都可以通过 session 判断用户是否已登录。

**② 密码验证**:

* 登录请求通过查询数据库获取对应的用户名，并使用 bcrypt.compare 来验证输入的密码与存储在数据库中的哈希密码是否匹配。
* 如果验证成功，则将 username 存储在 session.user 中，以便后续请求可以通过会话验证用户身份。

### ****5.6.4 增加安全性****:

* **Session 过期时间**：通过设置会话 cookie 的有效期为 **1小时**，避免用户会话一直有效，增加了安全性。如果用户长时间没有操作，会话会自动过期。
* **密码加密**：使用 bcrypt 加密用户密码，而不是存储明文密码。即使数据库泄露，用户密码也不会被直接暴露。

// 配置 session 中间件

app.use(

  session({

    secret: 'your\_secret\_key', // 替换为随机的密钥字符串

    resave: false,

    saveUninitialized: false,

    cookie: { maxAge: 60 \* 60 \* 1000 }, // 会话持续时间（1小时）

  })

);

// 中间件：验证用户是否已登录

function requireAuth(req, res, next) {

  if (!req.session.user) {

    return res.redirect('/login.html'); // 如果没有登录，重定向到登录页面

  }

  next();

}

# **六 心得与体会**

## 6.1 环境配置

node环境 node官网

express框架 npm install express

bcrypt加密 npm install bcrypt

SQ lite3 npm install sqlite3

multer下载 npm install multer

## 6.2 技术提升与收获

* **技术能力提升**：通过具体的项目实践，学会了如何使用各种技术（如Node.js、Express、Multer等）来解决实际问题。比如，利用中间件来处理用户会话，实现了用户鉴权功能，学习了如何保护路由以及如何处理文件上传等。
* **工具和框架的应用**：在项目中使用了bcrypt进行密码加密、sqlite3数据库操作等，这些技术提升了我对Web后端开发中安全性和数据持久化的理解。

## 6.3 前端与后端协作

* **前后端交互**：通过fetch进行前后端数据交换，实现了头像上传功能，并且利用localStorage进行前端数据的持久化，提升了用户体验。
* **数据存储与同步**：通过前后端的配合，实现了访问计数的持久化存储和更新，前端通过API获取访问数据，并展示出来，增强了页面的互动性。

## 6.4 不足与改进

* **性能优化**：尽管项目功能实现完整，但在高并发的情况下，如何优化数据库操作、文件上传等环节，提升系统的性能，仍然是一个需要进一步探索的问题。
* **用户体验**：在用户体验方面，未来可以增加更多的交互性功能，比如修改用户资料、展示用户的详细信息等，使得系统更加完备。
* localstorage的使用，图片修改multer上传

# 参考资料

1. 高德开放平台 <https://lbs.amap.com/>
2. Jabin Peng个人主页 <https://www.zpzpup.com/>
3. WeiTingting个人主页 <http://123.56.144.92/>
4. 数字游牧人samuel主页 <https://www.qzq.at/>
5. Echarts可视化 <https://echarts.apache.org/handbook/zh/get-started/>
6. [https://juejin.cn/post/7242127432203173948?searchId=20241114104732F14886292F96EA0A881](https://juejin.cn/post/7242127432203173948?searchId=20241114104732F14886292F96EA0A8801)