



服务外包项目实训报告

（ 2023/ 2024 学年 第 1 学期）

题 目： 仿www.booking.com网站

**专 业**  软件工程（嵌入式培养）

**学 生 姓 名**  叶信托

**班 级 学 号**  B20031908

**指 导 教 师**  沈殷涛/ 成小惠

**指 导 单 位** 北京中软国际教育科技股份有限公司

南京邮电大学计算机学院

**日 期**  2023.10.30 - 2023.11.10

**仿www.booking.com网站**

1. **概论**
   1. 课题目标：

本次项目实训中，我们选择的课题是“仿www.booking.com网站”。在项目中，我主要负责“爆款旅游目的地”和“查收邮件”这两个Vue界面的代码编写与逻辑交互。在这个过程中，用到以下技能和知识：

* 前端基础技术：HTML、CSS和JavaScript。
* 前端进阶开发框架和技术：Node.js、Vue2.0、Element-ui桌面端组件库。
* 后端基础技术：Axios库、请求异步加载、JavaSE等。
* 逻辑交互：学会在多个Vue界面间进行路由跳转与数据交互。

通过这个项目，我深入了解和实践现代前端开发的技术和方法，并将学会设计复杂的用户界面和交互，处理用户交互逻辑，并通过团队合作来开发高质量的软件。这将大幅度提升我的编程技能，并为将来的职业发展做好准备。

* 1. 任务分析：

在我所负责的方面，主要面临如下任务：

* 分析和理解Booking.com网站的前端界面和功能。
* 开发主界面组件：包括设计和布局HomeView主页面的第三个组件Comp3、实现用户交互功能以及展示“爆款旅游目的地”和“宣传图”。
* 开发“查收邮件”界面：包括路由跳转、数据交互、异常处理等内容。
* 撰写文档，包括实验报告和其他相关文档，记录实验过程、设计决策、实现细节和测试结果。
  1. 内容描述：

（1）在任务1“分析和理解Booking.com网站的前端界面和功能”中，主要包含如下内容：

* 分析和理解Booking.com网站的前端界面和功能：
* 研究Booking.com网站的页面布局、交互设计和功能。确定需要模仿的主要界面和功能元素。

（2）在任务2“开发主界面组件：包括设计和布局HomeView主页面的第三个组件Comp3、实现用户交互功能以及展示“爆款旅游目的地”和“宣传图”。”中，主要包含如下内容：

* 使用Node.js、Vue2.0、Element-ui桌面端组件库、HTML、CSS和JavaScript等前端框架和技术实现仿造Booking.com的界面。
* 该界面用于展示国内各地热门旅游目的地图片。在页面设计采用分层嵌套组件结构：父组件HomeView\_Com5用于整体布局,包括标题和子组件的容器子组件Country1和Country2用于分割显示不同区域的图片内容
* 通过JavaScript向后端服务器发起Axois请求，并从后端获取图片资源，展示在界面中。

（3）在任务3“开发“查收邮件”界面：包括路由跳转、数据交互、异常处理等内容。”中，主要包含如下内容：

* 界面显示一个背景图片和相关文字信息，包括标题、说明、按钮等。
* 获取上一个Vue界面传递的邮箱地址作为参数，并显示在界面上。
* 通过调用后端接口获取图片的URL，并显示在界面上。
* 点击返回登录界面的按钮，跳转到主界面。

（4）在任务4“撰写文档，包括实验报告和其他相关文档，记录实验过程、设计决策、实现细节和测试结果。”中，主要包含如下内容：

* 测试和调试：进行系统级和单元级测试，确保实现在功能和性能方面正常工作。使用调试工具和技术解决可能出现的问题和错误。
* 撰写文档：编写实验报告，记录实验过程、设计决策、实现细节和测试结果。撰写其他相关文档，如用户手册或技术文档，以便其他人能够理解和使用我们的实现。

1. **设计与实现**

在我主要负责“爆款旅游目的地”和“查收邮件”这两个Vue界面中，“爆款旅游目的地”界面主要基于Vue2.0框架和Element-ui组件库，结合HTML，CSS，JavaScript技术实现；“查收邮件”界面主要通过Aoixs异步加载、JavaSE等技术实现。程序的整体架构、主要功能流程、主要数据结构的设计、数据设计以及部分关键代码如下所示：

2.1.程序的整体架构

（1）“查收邮件”界面

本界面的代码使用Vue.js框架进行开发，包含了模板、脚本和样式部分。

* 模板部分：使用了Vue.js的模板语法，定义了界面的结构和布局。通过绑定数据和事件处理函数，实现了动态展示和交互功能。
* 脚本部分：使用了axios库进行前后端数据交互。在created生命周期钩子函数中，获取路由参数中的邮箱地址，并通过调用后端接口获取图片的URL。根据接口返回的数据，更新urls数据项，从而实现图片的动态加载和显示。另外，定义了navigatorToLog方法，在点击返回登录界面按钮时进行路由跳转到主界面。
* 样式部分：使用了scoped关键字将样式限定在当前组件中。样式部分定义了界面元素的样式和布局，以及一些交互效果的样式。

此架构基于Vue.js框架和axios库，通过前后端数据交互和事件处理，实现了邮箱链接确定界面的展示和交互功能。

（2）“爆款旅游目的地”界面

本界面通过Vue.js的模板语法和组件化开发，实现了主界面HomeView\_Com5和两个子界面Country1、Country2的代码架构。这种架构使得代码更加模块化和可维护，方便团队协作开发和代码重用。

* 主界面HomeView\_Com5通过引入子组件country1和country2，在页面上展示了两个子界面的内容。这种组件的嵌套关系使得界面结构清晰，各个组件之间的交互也更加灵活。同时，主界面还包含了一些静态文本信息，以引导用户了解爆款目的地的特点。
* 在数据获取方面，使用了axios库进行HTTP请求，通过调用后端接口获取图片的URL。这个过程在子界面的mounted生命周期钩子函数中进行，确保在组件挂载后立即获取数据。获取到的URL被绑定到组件的数据项urls上，实现了图片的动态展示。这样，每次页面加载或数据更新时，界面能够根据最新的URL信息动态呈现不同的爆款目的地图片。
* 代码还定义了组件的数据、方法和样式。每个组件都有自己的作用域样式，通过使用`scoped`属性，确保样式只应用于当前组件，避免样式冲突。这样可以更好地组织和管理组件的样式，提高代码的可读性和可维护性。

综上所述，通过组件化开发和数据动态绑定，实现了界面的模块化和数据驱动，为用户提供了一个直观而丰富的爆款目的地浏览体验。

2.2.主要功能的流程

（1）“查收邮件”界面

首先，在组件创建时获取传递的邮箱参数，然后在组件挂载后发送HTTP请求获取图片URL数据，并根据返回的数据进行成功或失败的处理。最后，定义了返回登录界面的方法，并设置了一些样式来美化界面。

具体流程如下：

* 使用Vue.js的模板语法构建界面，包括图片展示、文本信息和按钮等元素。使用CSS设置样式属性，如位置、字体大小和颜色。
* 引入axios库进行HTTP请求，并定义组件的data属性，包括mailbox、name和urls等数据项。
* 在created函数中，获取传递的RegisteredMailbox\_参数并将其赋值给mailbox属性，用于显示已注册的邮箱地址。
* 在mounted函数中，使用axios库发送HTTP GET请求，请求后端接口获取图片的URL。请求成功后，根据返回的数据中的code属性进行成功或失败的处理。如果code为1，将URL数据赋值给urls属性；如果code不为1，打印错误信息并弹出提示框。

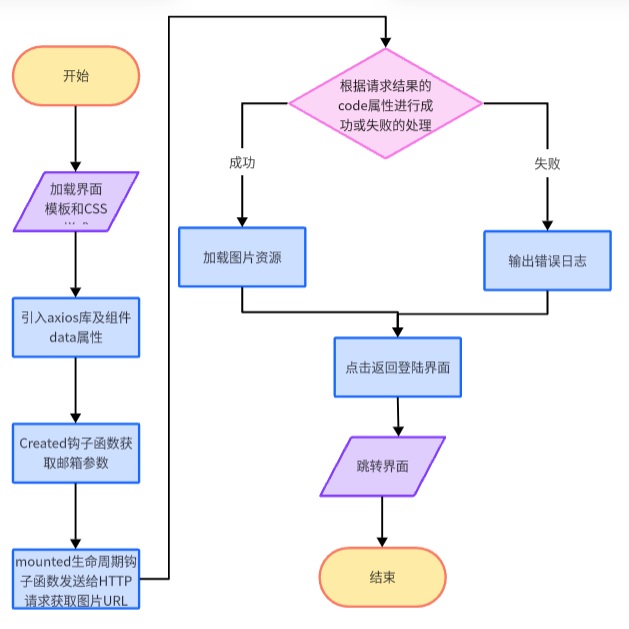


图1 “查收邮件”功能的流程图

* 在methods部分，定义了navigatorToLog方法，用于点击按钮返回登录界面。通过this.$router.push方法进行路由跳转，传递参数IsLogIn为true。
* 在style部分，使用scoped属性限定样式作用域，确保样式仅应用于当前组件。

1. “爆款旅游目的地”界面

该界面的主要功能是通过HTTP请求获取图片的URL，并将这些URL展示在界面上。它采用了Vue组件化开发的方式，使用了Element UI的网格布局，并对图片和布局进行了样式设置。每个子界面通过发送不同的HTTP请求来获取不同的图片URL，实现了展示不同国家或地区旅游目的地。主要功能流程如下：

* 主界面（HomeView\_Com5）的模板中定义了一个包含标题和介绍信息的区域，以及两个子组件（country1和country2）的展示区域。子组件是通过自定义的Vue组件引入的。
* 子界面Country1的模板中使用了Element UI的网格布局（el-row和el-col），展示了两张图片。图片的URL通过发送HTTP请求获取，并通过v-bind指令绑定到img标签的src属性上。
* 子界面Country1的脚本部分使用了axios库发送HTTP请求，请求的URL为'api/urls'，并传递了一个参数page，值为"Com5\_sub1"，用于指定获取的图片URL的来源。获取到的图片URL存储在`urls`数组中，然后在界面上展示。
* 子界面Country1的样式部分定义了一些样式规则，如设置图片的圆角、最小高度和宽度，以及调整布局和间距等。
* 子界面Country2的模板和脚本部分与Country1类似，但展示了三张图片，并在第一个图片上添加了一个链接，点击后可以跳转到其他页面。
* 在子界面的mounted钩子函数中，使用axios发送HTTP请求获取图片URL。如果请求成功（response.data.code === 1），则将返回的URL存储在`urls`数组中；否则，打印错误信息并弹出提示框。

2.3.主要数据结构的设计

（1）“查收邮件”界面

在该界面中，使用了一个名为`data`的对象来管理组件的数据。这个对象包含了以下属性：

* `mailbox`：用于存储已注册的邮箱地址。通过将传递的`RegisteredMailbox\_`参数赋值给`mailbox`，可以在界面中显示已注册的邮箱地址，提醒用户查收验证邮件。
* `name`：表示当前页面的名称。在发送HTTP请求获取图片URL时，会将当前页面的名称作为参数传递给后端接口。这个参数的作用是确保获取到与当前页面相关的正确的图片URL。
* `urls`：用于存储获取到的图片URL。在mounted生命周期钩子函数中，通过发送HTTP请求获取后端接口返回的数据，并将其中的图片URL赋值给`urls`属性。这样，界面中的图片标签就可以通过绑定`urls`属性来展示相应的图片。

基于这样的数据结构设计，能够有效地管理和使用邮箱地址、页面名称以及图片URL等数据，使界面能够正确地显示和展示相关信息。

（2）“爆款旅游目的地”界面

爆款目的地展示界面由一个主界面（HomeView\_Com5）和两个子界面（Country1和Country2）组成。

* 主界面（HomeView\_Com5）是整个界面的入口，展示了爆款目的地的标题和介绍信息。该界面通过引用子组件（country1和country2）来展示不同的国家或地区的旅游目的地。主界面的代码中，使用了一些内联样式来设置标题和介绍的样式。
* 子界面Country1展示了一个使用了Element UI的网格布局（el-row和el-col）的图片展示区域。通过发送HTTP请求获取图片的URL，并将其展示在界面上。子界面的代码中，使用了axios库来进行HTTP请求，并在mounted钩子函数中发送请求。获取到的图片URL存储在`urls`数组中，然后通过v-bind指令绑定到img标签的src属性上。
* 子界面Country2也展示了一个图片展示区域，但使用了稍微不同的布局方式。同样，它也发送HTTP请求获取图片URL，并将其展示在界面上。该界面使用了el-row和el-col来创建一个水平排列的图片区域，并在第一个图片上添加了一个链接，点击后可以跳转到其他页面。

综上所述，该爆款目的地展示界面通过Vue组件化开发的方式，结合HTTP请求和样式设置，实现了展示不同国家或地区旅游目的地的功能。

2.4.数据的设计

我所负责的“查收邮件”和“爆款旅游目的地”界面并没有涉及到数据的具体设计和存储，主要集中在界面展示、界面导航以及通过Axois方法异步从后端获取数据方面。

2.5.部分关键的代码

（1）“查收邮件”界面

部分关键的代码有如下几个方面：

* Created生命周期钩子函数：在这个钩子函数中，通过使用this.$route.params来获取传递的RegisteredMailbox\_参数，并将其赋值给mailbox属性。从而在界面中显示已注册的邮箱地址，提醒用户查收验证邮件。代码如下：

created(){  
 this.mailbox = this.$route.params.RegisteredMailbox\_;  
},

* Mounted生命周期钩子函数。在这个钩子函数中，使用axios库发送HTTP GET请求，向后端接口请求获取图片的URL。通过传递params参数，将当前页面的名称传递给后端接口，确保获取与当前页面相关的正确的图片URL。关键代码如下：

mounted() {  
 axios.get('api/urls',{  
 params:{page:this.name}  
 })  
 .then(response => {  
 //请求处理

})  
 .catch(error => {  
 //异常处理  
 });  
},

* 在处理请求成功或失败的回调函数中，首先检查返回的数据中的code属性。如果code属性的值为1，表示获取URL成功，将URL数据赋值给urls属性，以便在界面中展示相应的图片。如果code属性的值不为1，表示获取URL失败，会打印错误信息并弹出提示框，提醒用户获取URL失败。关键代码如下：

axios.get('api/urls',{  
 params:{page:this.name}  
})  
 .then(response => {  
 if (response.data.code === 1) {  
 // alert("获取成功！");  
 // url获取成功的处理  
 this.urls = response.data.data;  
 } else {  
 // url获取失败的处理  
 console.log(response.data.msg);  
 alert("获取失败！");  
 }  
 })  
 .catch(error => {  
 console.error('Error:', error);  
 alert('查询失败');  
 });

* navigatorToLog方法。这个方法是在点击按钮时触发的，用于返回登录界面。使用this.$router.push方法进行路由跳转，跳转到名为'home'的路由，并传递参数IsLogIn为true。这样，用户可以返回登录界面，并且系统可以根据IsLogIn参数来执行相应的登录操作，关键代码如下：

methods:{  
 navigatorToLog(){  
 // 用户存在，进行路由跳转  
 this.$router.push({  
 name: 'home',  
 params: { IsLogIn: true }  
 });  
 },  
},

通过以上的代码部分，能够获取传递的邮箱地址参数，请求并获取相关的图片URL，处理成功或失败的情况，并提供了返回登录界面的功能，能够完整地实现所需的功能和交互流程。

（2）“爆款旅游目的地”界面

该程序的关键代码部分主要用于发送HTTP请求获取图片URL，并将获取到的URL绑定到页面上的img标签上进行展示。在子界面的mounted钩子函数中发送请求，并通过axios库处理请求的响应，将获取到的URL存储在`urls`数组中，然后通过Vue的数据绑定语法将URL绑定到img标签的src属性上。同时，通过CSS样式设置图片的样式和布局。以下是程序的关键代码解释和说明：

* 在主界面的模板中，使用了Vue的组件引入方式，引入了Country1和Country2两个子组件。

import country1 from '../components/sub\_components/Com5\_sub1.vue'  
import country2 from '../components/sub\_components/Com5\_sub2.vue'

* 在子界面Country1的脚本部分，使用了axios库发送HTTP GET请求。请求的URL为'api/urls'，并通过`params`参数传递了一个名为`page`的参数，值为"Com5\_sub1"，用于指定获取的图片URL的来源。

mounted() {  
 axios.get('api/urls',{  
 params:{page:this.name}  
 })  
 .then(response => {  
 if (response.data.code === 1) {  
 // alert("获取成功！");  
 // url获取成功的处理  
 this.urls = response.data.data;  
 } else {  
 // url获取失败的处理  
 console.log(response.data.msg);  
 alert("获取失败！");  
 }  
 })  
 .catch(error => {  
 console.error('Error:', error);  
 alert('查询失败');  
 });  
},

* 子界面Country1的模板部分中，使用了Vue的数据绑定语法`:src="urls[0]"`将获取到的图片URL绑定到img标签的src属性上，实现图片的展示。

<el-row :gutter="15">  
 <el-col :span="60"><div class="grid-content ">  
 <img :src="urls[0]" class="threeline">  
 </div></el-col>  
 <el-col :span="60"><div class="grid-content">  
 <img :src="urls[1]" class="threeline">  
 </div></el-col>  
</el-row>

* 子界面Country1的样式部分定义了一些样式规则，如设置图片的圆角、最小高度和宽度，以及调整布局和间距等。

<style scoped>  
.bg-purple-light {  
 background: #e5e9f2;  
}  
.grid-content {  
 border-radius: 4px;  
 min-height: 280px;  
 width: 100%;  
  
}  
.threeline{  
 border-radius: 4px;  
 width: 100%;  
 height: 280px;  
 margin-top: 0%;  
}  
</style>

* 子界面Country2的代码结构和Country1类似，只是展示了三张图片，并在第一个图片上添加了一个链接。发送HTTP请求获取图片URL的方式和处理响应的逻辑也相同。

mounted() {  
 axios.get('api/urls',{  
 params:{page:this.name}  
 })  
 .then(response => {  
 if (response.data.code === 1) {  
 // alert("获取成功！");  
 // url获取成功的处理  
 this.urls = response.data.data;  
 } else {  
 // url获取失败的处理  
 console.log(response.data.msg);  
 alert("获取失败！");  
 }  
 })  
 .catch(error => {  
 console.error('Error:', error);  
 alert('查询失败');  
 });  
},

1. **系统测试**

（1）“查收邮件”界面

* 界面展示：



图2 “界面展示”功能测试

* 按钮点击测试：

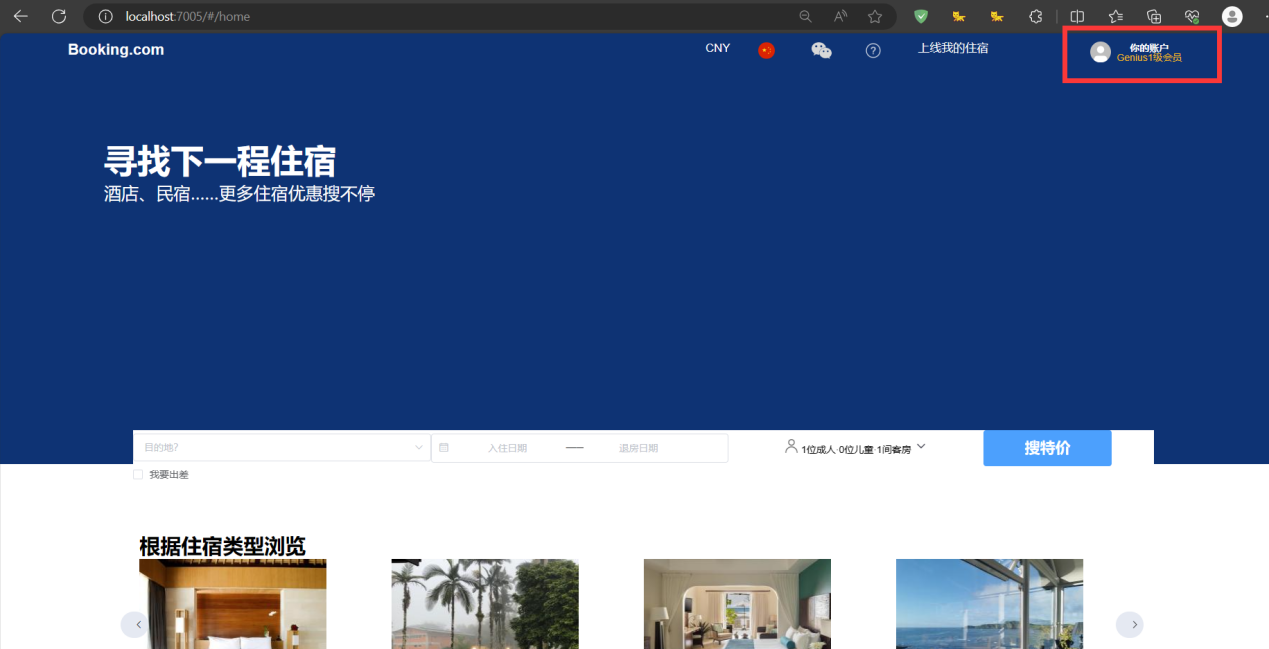


图3 “按钮点击”功能测试

（2）“爆款旅游目的地”界面

* 界面展示：

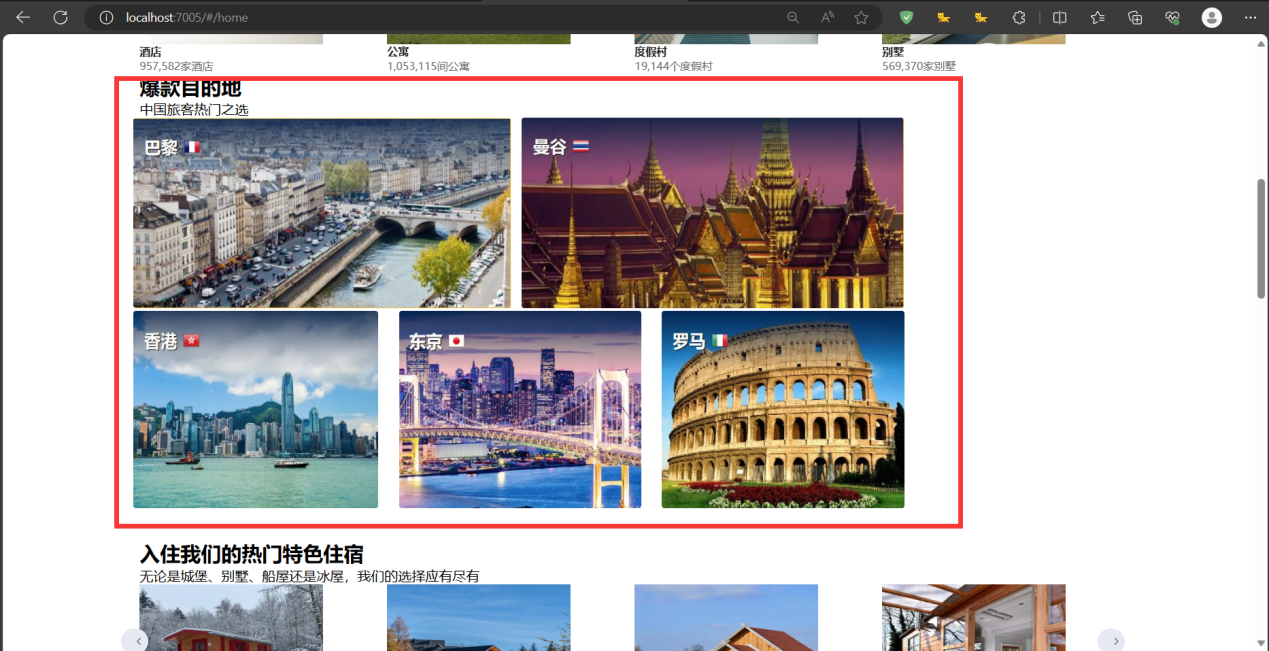


图4 “界面展示”功能测试

（3）结果分析：

测试结果良好，2个界面的功能均正常实现。

1. **评价与结论**

我的项目心得体会如下：

1. 学会使用组件化开发：使用Vue.js框架进行组件化开发，将页面划分为多个组件，每个组件负责不同的功能和展示。这样可以提高代码的可重用性和可维护性。

2. 学习了Axios请求：使用Axios库进行HTTP请求，通过调用API接口获取数据。合理使用Promise的链式调用，处理请求成功和失败的情况，并在控制台打印错误信息或弹出提示框。

4. 复习了数据绑定和动态渲染：通过Vue.js的数据绑定特性，将数据与页面元素进行关联。在mounted钩子函数中发起请求获取数据，并将数据赋值给对应的属性，从而实现动态渲染页面的效果。

5. 学会使用异常处理和错误提示：通过捕获异常和控制台输出错误信息，可以帮助调试和排查问题。使用弹出框组件进行错误提示，提醒用户请求失败或操作出错的情况。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 评分内容 | 评分细则 | | 优秀 | | 良好 | 中等 | 差 |
| 平时成绩（20%） | 学习态度；遵守实验室规定； （10分） | |  | |  |  |  |
| 考勤情况（10分） | |  | |  |  |  |
| 设计开发成果和验收答辩  （60%） | 程序演示与项目介绍：课题功能丰富，程序运行结果正确；界面友好，设计合理，创新性强。（10分） | |  | |  |  |  |
| 程序开发与实现能力；工作量饱满；代码质量和运行性能良好，算法正确；（30分） | |  | |  |  |  |
| 答辩时内容阐述清晰准确；问题回答正确（20分） | |  | |  |  |  |
| 设计报告质量（20%） | 报告文档书写格式规范、排版美观、文字流畅（10分） | |  | |  |  |  |
| 内容正确详实、结构合理、反映系统设计流程（10分） | |  | |  |  |  |
| 评分等级 |  | | | | | | |
| 指导教师  签名 |  | 日期 | | 2023-11-12 | | | |
| 备注 | 评分等级有五种：优秀、良好、中等、及格、不及格 | | | | | | |