移动电子商务技术课程报告

实验目的

本次实验使用了vue框架，及其配套生态element-ui等各种强大组件，加上基于JavaScript、css、HTML语言的学习，实现了一个简易的移动电子商务网站。一方面，我们了解到了一个电子商务系统的设计目标、设计需求以及实现的方法，能够从系统的观点出发分析，规划，设计和实施一个简易的移动电子商务网站；另一方面，我们学习了vue这样一个深经考验、在实战生产环境中应用广泛的 JavaScript 框架，它能够更好、更轻松地处理许多web应用的场景；最后一方面，我们还体会到了前后端分离下互联网项目开发的优势、掌握了相应的技术栈，如 Ajax 异步请求等。

实验内容

本课程以实战能力的学习为主，重点介绍了前端框架Vue并配合实际项目案例进行学习，整个过程可以分为如下三个阶段：

1. 在课程开始之初，我们首先学习了实验所需的相关基础，主要有以下方面：
    1. 了解常见的电子商务网站，以及了解相关的机制，某些问题的处理，比如淘宝的界面到底都怎么实现，淘宝是怎么处理"双十一"这样的高并发时段的问题的。
    2. 学习前端web的设计实现，包括一个网页页面从设计、实现再到渲染的过程，了解它是如何先把一个页面做出，再把一个页面做好的过程。
    3. 学习JavaScript语言，其是一种轻量级、解释型、即时编译型的语言，主要用于Web界面功能的开发。 学习了HTML语言，这是一种超文本标记语言，通过一系列标签使得网络文档格式统一，连接分散的Internet资源。
    4. 学习基于标准HTML，css，JavaScript的vue编程框架模式，逐步步入实战学习的过程。
    5. 学习在Node.js中使用JavaScript完成服务端的开发，能够支撑起一个电子商务网站的具体服务功能
2. 第二阶段，我们查看和学习了一些完成的具体实例，包括一个仓库管理的微信小程序，了解到了一些小程序的开发，以及其中设计实现的细节。

    老师带着我们自顶向下的分析了整个项目的实现，首先分析了 整体间各个部分的关系，然后分析了其中代码实现的具体功能细节，了解了很多实例设计中所需要 做的内容，从中了解到实现一个完整的Web功能都有哪些需要做，并且怎么做。 再此基础上，从第一阶段到第二阶段的过程中，老师都给我们了许多能够自学的网络平台和其它资源，让我们能够从理论和实践两方面同步学习设计一个电子商务网站的具体内容。包括从其他实现的项目功能以及开源代码、教学书籍、教学网站中得到我们完成自己设计需要的能力和内容。

3. 第三阶段就是具体完成一个电子商务网站了。老师提供给我们一个电子商务网站的模板，或者说雏形，我们需要做的工作是在这个雏形的基础上做自己的添加处理和修改，将这个电子商务网站雏形，通过结合自己理解中的电子商务网站应该有的界面功能，做设计和具体的实现，最终使得形成了一个界面良好的电子商务网站，达到了本课程的学习目的。

   本课程主要学习目的是学习一个电子商务网站界面是什么样的，又是怎么实现的，如何使用vue框架结合一些其他的JavaScript代码，完成电子商务网站的界面，主要的设计和实现在网站界面的前端功能上，后端要求比较小，数据量也不大，由于侧重点的考虑，后端并不是一个完善的实现，某些操作和数据并没有连接数据库。

实验过程和页面展示

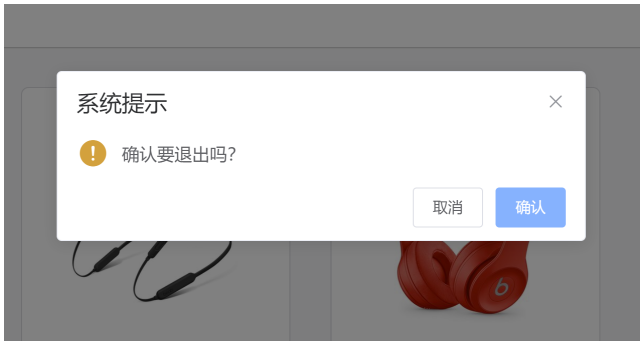
本次实验是做一个电子商务网站的前端界面，后端服务器主要是辅助前端正常运行测试的功能，做的并不复杂，所以功能测试和设计都以前端为主，分阶段的叙述整个实验过程：

1. 首先先查看已经有的电子商务网站的雏形，尽管叙述的时候说是一个雏形，但是这个雏形已经具备不少功能，包括显示商品、显示其图片和描述等具体信息，每个商品也有内置的详细描述，还有一些条件筛选功能，以及购物车功能，对于选中的商品可以加入购物车，最后选择购买，这其实已经有一个电子商务网站了基本要求了，考虑我们平时上淘宝的操作，比如我们要买一台电脑，其实也就是先以想要的型号、价格排序、颜色等条件筛选，将备选加入购物车，最后下决定了将某个下单，但是回过头考虑整个电商网站， 这无疑缺少了关键的一个方面，就是一个电子商务网站肯定有大量的客户，区别不同客户的方式是每个客户有一个自己账号，这就说明电子商务网站需要一个登录界面。
2. 根据需求，完成一个登录界面，我们首先思考一个正常的电商网站的登录功能，最开始一个电商网站是没有客户的，新的客户需要注册得到一个专属的账号，然后客户就可以使用这个账号完成登录，从而进一步完成购买的需求。但是无疑，注册功能和账号密码这个对于客户身份的验证功能，无疑需要数据库的连接，也就是后端技术的支持，而这并不是我们学习和实践完成的侧重，所以我们弱化后端需要实现的功能，主要先做一个登录界面出来，忽略掉注册功能，这个登录界面需要输入账号和至少8位的密码（这个检验无疑在前端就可以完成），然后根据用户名“判断”客户，完成登录功能，这就是一个基本的登录界面的完成。

   最终实现效果如下所示：

在首页中，我们也能看到相应的账户信息，购物车被隐藏到了选择栏中：

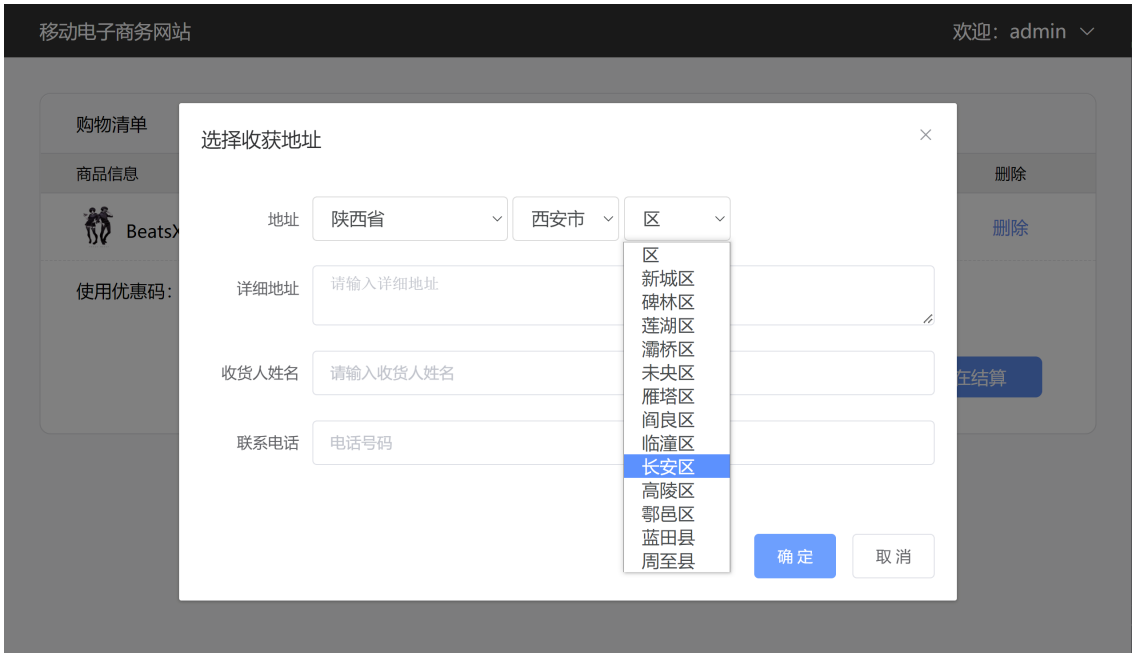
点击退出登录，会弹出确认弹窗：



3. 完成登录功能后，继续考虑我们这个电商网站少了哪些核心功能，回想我们买电脑的过程，我们选好电脑，下单在付款前，还需要进行一个重要的操作，我们需要填写我们的收货地址，这不同于线下购买，商家想要将东西交到你手上，无疑需要获取你的地址。 点击现在结算后，就能看到以上一个画面，这就像是我们平时购买东西的界面一个，完善这个界面，你就可以等待你心想的商品发货了（虽然在这个网站"购买"的东西永远不可能发货。）

   最终实现效果如下：

   当点击现在结算按钮后，会弹出一个地址信息对话框，按要求填写后即可购买该商品



实验具体实现

登录页面的实现

首先需要添加路由拦截功能，对于主页、购物车等页面，只有登录后才能显示，具体实现如下：

```
router.beforeEach((to, from, next) => {
    if (to.meta.isLogin) {
        let token = store.state.userinfo.token;
        console.log(token)
        if (token) {
            next()
        } else {
            next('/login')
        }
    } else {
        next()
    }
    window.document.title = to.meta.title;
});
```

如果前端拿到了token值，则说明已登录，否则跳转到登录页面

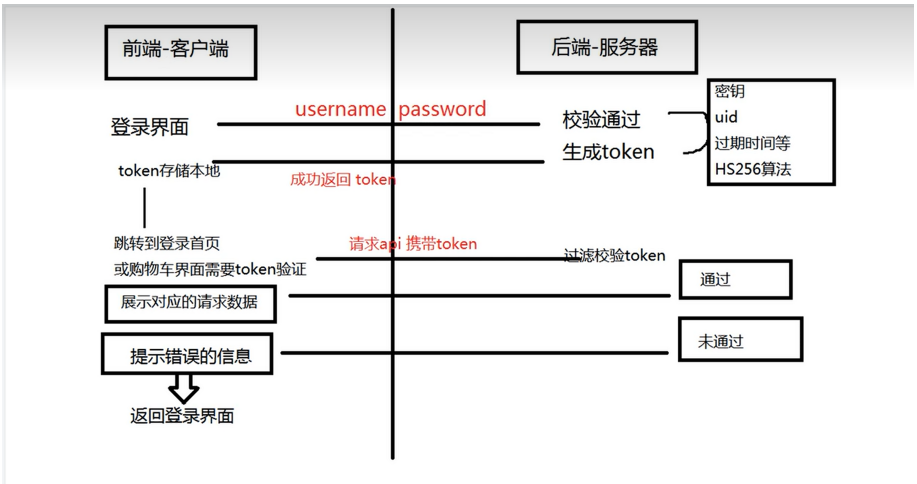
登录页面使用了element-ui组件进行美化，并添加了表单验证规则，要求用户密码长度必须在8-12之内：

```
var validatePassword = (rule, value, callback) => {
  if (value === "") {
    callback(new Error("请输入用户密码"));
  } else if (value.length < 8 || value.length > 13) {
    callback(new Error("长度必须在8-12之内!"));
  } else {
    callback();
  }
};
```

在点击提交按钮之后，需要访问后端，因此在后端需要添加一个登录接口，其功能主要是根据用户输入的信息生成token值返回到前端：

```
//登录接口
app.get('/shopping/login', function (req, res) {
    let user = req.query.user;
    let pwd = req.query.pwd;
    // 查询数据库--省略
    const token = jwt.sign({ user, id: 123 }, secret.secretKey, { expiresIn: 20 })
    res.send({
        info: 'success',
        status: 200,
        token
    })
})
```

关于token的作用如下所示，主要作为用户标识，可以使用jsonwebtoken包生成：



前端拿到token后需要存入vuex中，以方便其他组件获取，因此在vuex中增加了相应的用户字段以及对应的方法：

```
const store = new Vuex.Store({
    state: {
        productList: [],
        cartList: [],
        userinfo: {
            username: '',
            token: null,
        }
    },
```

```
// 设置token
setUser(state, payload) {
    state.userinfo = payload;
},
// 清空用户信息
clearUser(state) {
    state.cartList = [];
    state.userinfo = {
        username: '',
        token: null,
    }
},
```

另外，这里还涉及一个小细节，就是如何从token字段中解析出所需的用户名，这里使用了jwt-decode包，当然其不会解析出密钥和过期时间。

最后，为了使得页面在刷新之后，用户登录状态保持不变，这里使用了session会话存储用户信息以实现数据持久化，在main.js中判断其是否存在，设置好vuex：

```
let userinfo = sessionStorage.getItem('userinfo')
if (userinfo) {
    userinfo = JSON.parse(userinfo);
    store.commit('setUser',userinfo)
}
```

整个用户登录过程代码如下所示：

```
submitForm(formName) {
  this.$refs[formName].validate((valid) => {
    if (valid) {
      request.SetSrvHost();
      request.get_login_token({
        user:this.loginForm.username,
        pwd:this.loginForm.password
      })
      .then(res=>{
        // console.log(jwt(res.data.token));
        let obj = {
            token:res.data.token,
            username:jwt(res.data.token).user
        }
        this.$store.commit('setUser',obj)
        // 数据持久化
        sessionStorage.setItem('userinfo',JSON.stringify(obj))
        this.$router.push('/list')
      })
```

做完登录页面之后，我们可以在主页上显示用户信息，并设置退出登录按钮

为了美化起见，我将购物车隐藏到了个人信息的选择栏下：

```
<el-dropdown @command="handleUser">
    <div class="header-title">
        欢迎：{{ this.$store.state.userinfo.username }}
        <i class="el-icon-arrow-down el-icon--right"></i>
    </div>
    <el-dropdown-menu slot="dropdown">
        <el-dropdown-item command="cart" class="header-menu-cart">
            购物车
            <span v-if="cartList.length">{{ cartList.length }}</span>
        </el-dropdown-item>
        <el-dropdown-item command="exit">退出登录</el-dropdown-item>
    </el-dropdown-menu>
</el-dropdown>
```

点击退出登录按钮，还会弹出确认栏，防止误触：

```
else if (comm == "exit") {
    this.$confirm("确认要退出吗？", "系统提示", {
    confirmButtonText: "确认",
    cancelButtonText: "取消",
    type: "warning",
    }).then(() => {
        this.$store.commit('clearUser');
        sessionStorage.removeItem('userinfo');
        console.log('退出成功');
        this.$router.push('/login');
    });
}
```

地址管理页面的实现

地址管理页面也使用了element-ui中的对话框组件进行美化，其中的地址选择栏还使用了v-distpicker包：

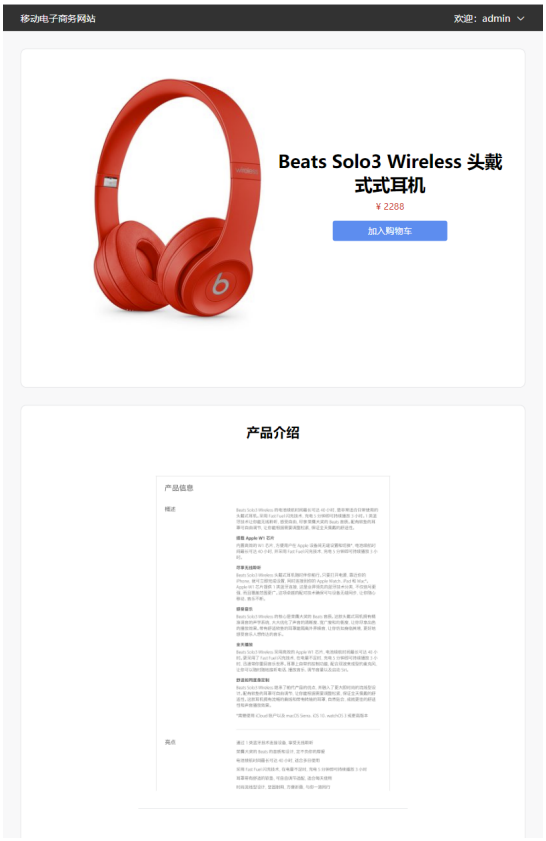
```
<el-dialog
title="选择收获地址" width="700px"
:append-to-body="true" :visible.sync="showAddressInfoFlag">
<el-form label-width="100px" :model="addressForm">
    <el-form-item label="地址">
        <v-distpicker
        :province="select.province" :city="select.city" :area="select.area">
        </v-distpicker>
    </el-form-item>
    <el-form-item label="详细地址">
        <el-input type="textarea"  v-model="addressForm.address"
            placeholder="请输入详细地址" autocomplete="off"></el-input>
```

对话框的显示使用showAddressInfoFlag变量进行控制，当其为True时，显示对话框，否则进行隐藏，这也是element-ui已封装好的功能。

```
initForm(){
    this.select = { province: '', city: '', area: '' };
    this.addressForm = {
        account: "",
        name: "",
        qualification:"",
    };
},
showAddressWin() {
    this.initForm();
    this.showAddressInfoFlag = true;
},
```

其他细节

在商品详情页面中，原程序的处理逻辑是把所有图片加载到相应页面，且图片存放在前端，我将其搬到了后端，并实现了商品与产品介绍的一一对应：



实验总结

通过本次实验，我们对vue的掌握更加熟练，也学习到了用户登录时前后端交互的整个过程，能够使用vue框架配合相应的组件写出较为美观的界面，深入了解到一个电商类网站所需的功能以及实现方法、步骤，积累了良好的项目经验。特别在实践过程中，了解到了element-ui、vant等等优秀的Vue组件库，帮助我们快速上手开发，也使得界面更加美观。总之，我深深体会到了Vue框架的优异以及其生态系统的完善，对电子商务系统也有了一个较为清晰的认识，收获颇多。