**金刚石文档**

**[项目总结]**

本项目的项目总结由组员的每个人的总结组成，如下：

1. 刘奕博：

在这次敏捷开发的项目当中，我负责的是前端开发，具体的工作主要是注册、找回密码界面和团队空间相关功能。

我遇到的第一个难点是css的样式，有的时候明明在自己的电脑上显示达到了预期效果，换了别人的电脑就大相径庭。在研究与请教之下，我发现将长宽及位置使用比例进行规定很大程度上能够避免这类问题（相对于使用像素进行定义）。

第二个难点是对vue特性和js、html语言不太熟练从而很多效果都没有能够很好的事件完成。举一些例子，譬如页面间传递参数，我只会使用jQuery来进行传递，而同组成员使用路由的params进行传递，这样不利于代码的阅读和维护，改的样式统一也增添了工作量。再比如html有的容器不能放有的标签，js的foreach不能break……这些都是我开发之前不甚了了而又想当然最终造成了很大困境的弯路。

我认为，在这次项目中做的较好的部分就是整体的交互逻辑，使用体验和UI设计这些相对“润物细无声”的地方。所有的ui都适配了互动，用户几乎所有的“我觉得这个功能应该点这里”几乎都确实如此。这除了要归功于elementUI的强大与美观，vue的功能丰富，还离不开组长和同侪们的不断设计和努力。

这次我的收获也十分充裕。从技术层面看，我对于vue，webpack、vuerouters、elementUI有了基本的了解，也掌握了初步的应用和开发能力。特别是elementUI，封装了美观的界面和方便的表单验证和事件管理，让我感到web前端开发原来可以如此的美妙。其次，对于使用github进行团队协作也有了掌握——相较于上次使用社交媒体软件群聊进行同步，git的同步，版本管理功能原来是如此强大……其次，从项目的层面来看，我也对于敏捷开发有了初步的认知：不拘泥于文档，开发测试同步，在小规模开发团队之中的站会沟通……这样的工作流程也极大地促进了项目的成功。最后，在组长的不断指导和帮助下，我对于前端开发的技术近乎开天辟地的取得了进步，更是重新对开发、测试工作燃起热情，这对我的帮助是不可估量的。

同样，我们本次的项目也有遗憾。就我来说，我认为最大的不足在于组件化的缺乏，折旧对于代码的测试、维护、迭代和阅读造成了极大地不便。在后期，许多功能相似的页面也有着不少相似的代码，这无疑是一种浪费。对于我来说，造成这种不足的原因并非是类设计的不成功，而是技术能力相对不足，组件间通信的技术和语法还没有熟练掌握，导致了这种妥协的发生，希望下次可以进行规避。

1. 叶天宇：

在本次的网站前后端开发中，我主要完成的是前端代码的编写，前端代码主要作用时完成页面的布局和渲染以及页面逻辑的设计。

在本次前端的开发中，我主要负责编辑器部分的相关内容和部分其他页面的优化问题。

在编辑器页面首先遇到的问题就是富文本编辑器的选择以及实现图片文字共同上传的相应问题。因此，在考虑多种编辑器后，我们选择了mavon富文本编辑器，这是一种支持markdown语法的富文本编辑器，在编辑器中可以实现markdown语法进行编写，同时，提供预览功能。在图片的处理上，我们富文本编辑器提供传输本地图片的接口，先将图片上传到服务器端，再在编辑器的文本内容中导入url链接，获取图片的路径，从而完成图片的上传和显示。

在本次前端开发的过程中同样还有前端页面样式布局的问题，如何调整页面的结构层次，如何实现美观简约的布局风格是我们在本次前端设计中很重要的一个问题，我们在设计中使用了功能强大的elementUI进行设计，借助了其他的第三方库来进行相应的页面信息功能的完善，同时也借鉴了主流网站的页面布局，进行本次编辑器开发的页面设计。

1. 张涵珂：

在本次项目中，我作为组长，负责了任务的分配、后端大部分api的编写、数据库设计、前后端服务器部署与维护、前端部分页面的一个修改和优化。

首先第一个比较棘手的点就是后端数据库的设计，因为前后端分离，后端需要先自行初步确定好一些想法，并且以此设计出数据库，才能进行api的编写，这样做的难点就是，如果这些初步的想法和之后的实践存在冲突，就需要在之后对数据库表进行大幅的修改与添加同时对已经写好的api进行相应的修改，这样就给开发带来了很大的阻力。所以我们在第二天开始，相关表的设计，都会通过与前端同学的交流沟通过再进行进一步字段的设计，这样很好的避免了相关的问题。

第二个比较棘手的点就是前端所需数据的多样性。因为往往我们需要通过一个接口来给前端返回丰富的数据进而避免二次请求浪费时间同时消耗服务器资源。所以api编写时返回的数据就显得非常重要，但由于python提供的查询函数有限，联合查询更是不能无法提供全套数据，只能提供相关单个数据，如果想要返回更多数据，就需要进行非常多联合查询，在response的编写也同样非常浪费时间。所以我们通过使用django rest framework框架的使用，利用serializers序列化来进行对返回数据的序列化，同时可以在序列化的时候加上相关外键，就能把外键对应表的所有数据一起序列化返回给前端，让前端能够拥有丰富的数据，同时后端代码量也很少。

最后一个难点就是后端api编写过多的一个问题。我们这次为前端提供了非常多api当然也是全部使用了，但是问题就是很多api的代码非常类似，但由于不同的请求需要不同的返回数据，所以也需要再写一个api，这样虽然前端请求起来非常方便，但后端编写就显得非常冗余。所以代码的复用显得非常重要，我们这次就将很多操作抽离出方法进行使用，例如token的验证和文档id的一个验证，通过方法来减少冗余程度。前端有时为了复用接口需要在一个操作中进行两次请求，所以为了这个问题的解决，我们也需要将两个api合为一个，缩短请求所需时间，所以这也是导致api很多的一个原因。这个问题还有待解决。

以上就是我的一个简单总结，都是和后端相关的，希望在之后的开发中能够做得更好。