1. 技术选项及理由

网页端：

Web开发框架使用基于vue库的模式，开发语言使用javaScript。

选用Vue的原因在于其本身提供了 MVVM 数据绑定和一个可组合的组件系统，具有简单、灵活的 API。从技术上讲， Vue.js 集中在 MVVM 模式上的视图模型层，并通过双向数据绑定连接视图和模型。实际的 DOM 操作和输出格式被抽象出来成指令和过滤器。

由于Vue的组件化特性，使得开发出来的页面具有组件性，方便代码的重用和查看。

服务器端：

Web开发框架使用Flask，开发语言使用Python。服务器端采用经典的MVC架构。服务器采用Linux，数据库采用MySQL，Web服务器采用Flask自带的服务器。

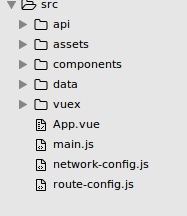
使用Flask的原因首先是该框架轻量易用，同时Python语言的简洁特性能够方便快速的开发和迭代，减少了后期维护和继续开发的代价。

服务器端采用ＭＶＣ架构使得整个服务器的层次比较清晰，同时也便于维护和拓展。我使用Flask自带的数据库采用MySQL主要便于数据库的设。我们使用SQLAlchemy进行数据库和web服务器的连接，快捷的实现了数据库数据和Python对象的直接转换。。

1. 架构设计

网页端：

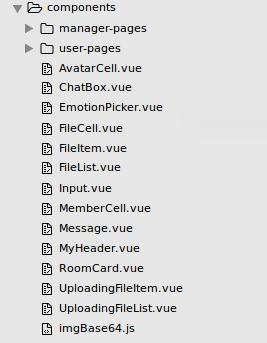
网页端的代码结构如下图



api：代码文件主要是用于封装一些业务逻辑的api，诸如登录验证之类的，同时还封装了基于fetch这一异步网络请求的api从而实现了代码里面复用；

assert：放置一些图片相关的静态资源

components:放置了所有组件的组件代码，如下

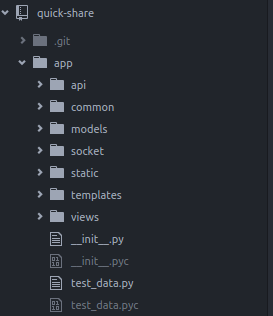


其中为了方便区分用户级别和管理员级别的代码将用户的页面组件都放于user-pages，管理员的页面组件都放于manager-pages;其余些可以共用的组件的都放于components目录下。

Vuex: 用于放置共享状态变量处理的代码文件

服务器端：

服务端的代码结构如下图



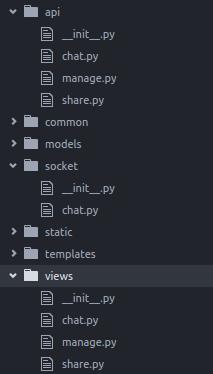
属于典型的MVC结构，其中models里面放的是数据库的结构，static和templates里面放的是静态页面。剩下的用于controller的内容比较庞大，我根据接口类型的不同，将controller的才分成api，socket和views三个部分。其中api返回的是json类型的数据，socket返回的是socket类型的数据，views则返回的主要是页面数据。

1. 模块划分

在整体的模块划分上，我采用的是MVC的这种竖直的结构，但是由于MVC结构的特性，controller的部分难免变得比较臃肿，为了能够减少controller部分的臃肿的状态，我将不同类型的结构分门别类，在各个类型的接口中，还是按照各个业务模块继续分类。通过采用这样一种扁平化分类的方法，降低了controller部分的臃肿。

在各个接口类型中，我根据每个类型的业务需求，把他们各个独立分类，虽然可能会有些许冗余，但是确实便于了代码的管理。

业务模块如下：



1. 软件设计技术

网页端：

使用的主要技术如下：

vue: 用于构建交互式的 Web 界面的库。它提供了MVVM数据绑定和一个可组合的组件系统，具有简单、灵活的 API。

Socket.io: 将Websocket和轮询(Polling)机制以及其它的实时通信方式封装成了通用的接口，极大方便了实时通讯的编程

vuex: Vuex 是一个专门为 Vue.js 应用所设计的集中式状态管理架构,使用它可以更好地组织代码，以及把应用内的的状态保持在可维护、可理解的状态。

服务器端：

使用的主要技术如下：

Flask: Python的微型框架，简单易用，有非常优雅的路由函数编写方式，同时可以非常便捷的使用插件，也有数量丰富的插件库。

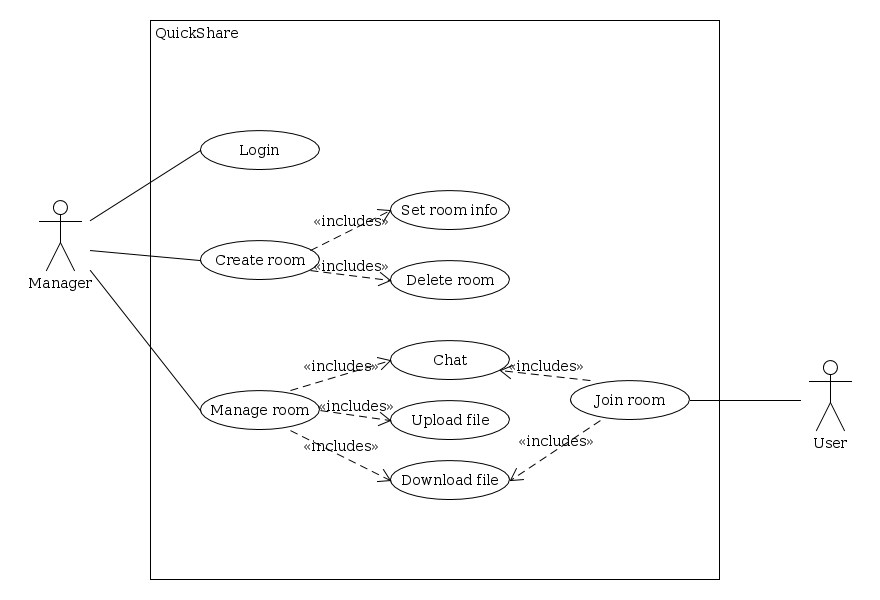
Socket\_IO：支持多种socket引擎，极大方便socket的程序开发，但是也有调试不便等缺点。

Sqlalchemy：数据库访问插件，能够直接将数据库表自动转换成python数据结构。

Redis：缓存数据库，能够方便的存储一些临时数据于内存中，加快了数据访问，同时也减小了真正数据库的存储负担，但是也加大了数据同步和保存的负担。

1. ＵＭＬ图

用例图：



领域模型：

