1. 技术选项及理由

网页端：

Todo

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。。

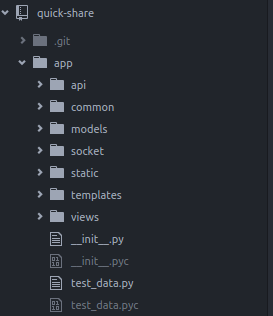
1. 架构设计

网页端：

Todo

服务器端：

服务端的代码结构如下图



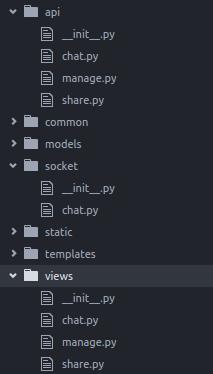
属于典型的MVC结构，其中models里面放的是数据库的结构，static和templates里面放的是静态页面。剩下的用于controller的内容比较庞大，我根据接口类型的不同，将controller的才分成api，socket和views三个部分。其中api返回的是json类型的数据，socket返回的是socket类型的数据，views则返回的主要是页面数据。

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：

Todo

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。

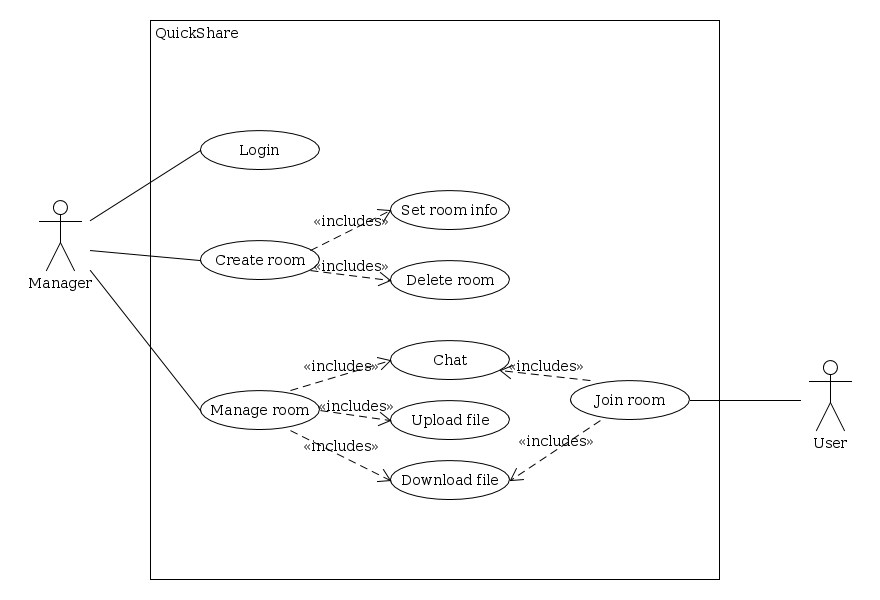
Socket\_IO：支持多种socket引擎，极大方便socket的程序开发，但是也有调试不便等缺点。

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：

