1. **性能**

因该项目目标用户群体数量稳定且不会太多，因此本项目并未对高并发做相应的优化，如果考虑高并发情况，可以从以下方面考虑进行优化：

1、服务器响应时间:

-优化数据库：使用索引、优化查询语句、使用更高效的数据存取策略。

-应用缓存：实现数据缓存策略，如Redis缓存频繁访问的数据。

-异步处理：对于耗时的数据处理任务，采用异步方式处理，使用RabbitMQ,kfaka等中间件。

1. 负载均衡

-软件负载均衡，使用nginx等软件。

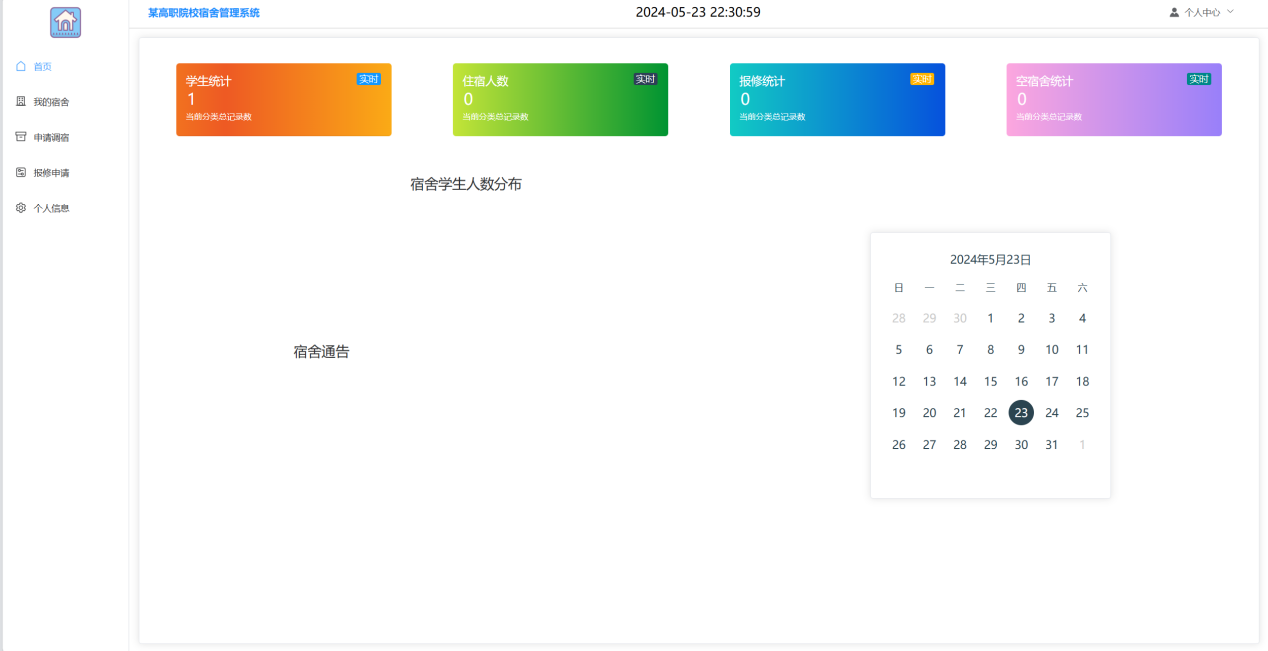
-DNS负载均衡，通过DNS解析策略将流量分配到多个服务器。当用户请求解析域名时，DNS根据负载均衡算法返回不同的IP地址，将用户分散到不同的服务器上。提高服务器面对高并发时的性能。

1. 分布式数据库

-主从复制+读写分离+负载均衡

-主从分离+垂直分区+水平分区 等实现方式

1. **可用性**



本项目在前端设计上实现了简洁高效的设计，在用户的使用简易型，有效性，效率等方面都十分突出

且几乎每个关键操作都有二次确认，在容错度方面也十分显著

如果需要进一步提高可用性，可以考虑以下策略：

1. 对移动端进行进一步适配，目前该项目在移动端表现仍不甚理想
2. 添加新用户引导，进一步增加用户上手简易度
3. 添加反馈界面，对用户不满处进行改进