体系结构

团队第二次作业大纲

设计部分

**需求情况**

1. 系统主要功能列表、非功能需求列表、项目约束

**系统主要功能**

* 余票查询:根据时间、地点等输入信息检索车次和剩余座位等信息
* 车票购买:从检索到的车辆中选定指定车次，购买车票、选择座位
* 车票改签、退票:对未出行的车票进行改签、退票等操作
* 查询订单:查询用户历史已出行订单和未出行订单
* 账户管理:对个人账户信息进行管理，增删常用乘车人
* 时刻查询:提供列车运行时刻表和列车实时的正晚点信息

**ASRs scenarios**

互操作性

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 系统 |
| 刺激 | 系统请求和第三方系统交互 |
| 制品 | 第三方系统 |
| 环境 | 系统运行时或构建时 |
| 响应 | 系统成功拿到第三方系统的服务 |
| 响应度量 | 交互请求成功率大于99.9% |

易用性

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 终端用户 |
| 刺激 | 想要使用系统完成操作 |
| 制品 | 系统 |
| 环境 | 系统运行时 |
| 响应 | 能够正确引导用户完成操作  提供撤销或取消功能，在用户操作过程中识别并纠正用户错误  错误发生时的恢复 |
| 响应度量 | 用户完成单项操作的时间小于等于2min  用户操作成功率应该大于等于99%  错误恢复时间小于5min |

性能

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 用户，第三方系统 |
| 刺激 | 非周期性的数据请求，操作请求 |
| 制品 | Broker模块，服务器模块，request bus |
| 环境 | 系统正常运行时，系统高负载时 |
| 响应 | 系统执行用户请求的操作；处理用户数据；返回数据 |
| 响应度量 | 请求响应延迟时间不超过5s  请求丢失的比例不超过0.001% |

Scalability 场景：硬件升级

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 开发人员 |
| 刺激 | 系统硬件需要升级，如可能增加服务器数量，或者对单个服务器进行硬件的修改 |
| 制品 | 系统硬件 |
| 环境 | 系统在构建中或者已经上线 |
| 响应 | 对系统硬件进行淘汰或更新  修改项目使之适应新的硬件环境  测试更改对于系统造成的影响  发布维护变更 |
| 响应度量 | 升级时间不应超过8小时  升级所影响的代码量不应超过2% |

Scalability 场景：服务器数据库升级

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 开发人员 |
| 刺激 | 服务器数据库需要进行扩容或者分布式处理 |
| 制品 | 服务器，数据库 |
| 环境 | 系统在构建中或者已经上线 |
| 响应 | 对数据库进行升级处理  修改服务端响应代码  测试修改后性能 |
| 响应度量 | 升级时间不应超过8小时  升级所影响的代码量不应超过3% |

1. operational scenarios ，ASR：以scenarios的形式展现？
2. **ADD设计过程**，两个架构分别多次迭代

每次4个步骤：选择元素、选择ASR、策略候选表和设计决策、本次的结果也就是确定下一次迭代的输入。每次迭代产生一个逐步细化的模块图

设计完成之后自然会产生总的模块图，再画下面俩（也许这里module视图就画完了）

1. 两种体系结构设计分别画一个module视图，一个component-connector视图

每个图需要介绍：

* 图片
* 每一个模块、元素的功能介绍，包括元素之间的大概接口说明
* 设计决策的过程、设计的合理性分析证明，有理有据，应对后面别组的测评
* 大佬组文档中有可变性指南，即各种处理是设计时选择还是运行时选择，如何应对需求、硬件、数据等的变更。可以参考写一写

**选出一个架构，再去设计实现**的情况：

1. 两个系统架构设计的优缺点分析、对比
2. 架构的选择原因，设计上的决策，体系结构模式、tactic的使用
3. UML类图，展示类和之间的关系
4. component/connector到实现类之间的匹配关系，类图形式

**其他**

1. 好像是测评的内容，包括utility tree（质量属性效用树，用来获取ASR的方法），敏感点、tradeoff点
2. 项目过程中的挑战、最佳实践经验、学习与收获
3. 每个人的分工与贡献

原型部分

1. 代码
2. 可执行程序
3. 部署使用说明