项目报告

## 项目名称

南京大学讲座管理系统

## 团队成员

丁霄汉 131250207 倪小凡 131250185

陈云龙 131250181 罗瑶 131250177

吴超月 131250168 曾婧 131250159

邹卓晋 131250158 梁思宇 131250129

## 主要功能

用户管理（注册，登录，用户权限管理，个人信息修改）

学生功能（讲座推荐、查看讲座信息，报名，退选，评价，讨论讲座，为收费讲座支付入场费，收看允许录像的讲座的直播和回放）

管理员功能（导入、修改讲座信息，强制报名退选，讲座反馈，讨论区管理）

## 使用场景

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **系统名称** | **子系统** | **预期输出** | **备注** |
| 1 | 用户管理 | 注册 | 该账户已被注册  账户注册成功 | 用户注册登记 |
| 2 | 用户管理 | 登录 | 登录成功/失败 |  |
| 3 | 用户管理 | 用户权限管理 | 权限修改成功/失败 | 管理员可查看并修改用户权限 |
| 4 | 用户管理 | 个人信息修改 | 信息修改成功/失败 | 用户可修改其个人信息 |
| 5 | 学生系统 | 讲座推荐 | 学生登录后在首页上显示针对他的推荐讲座 | 学生的院系、课程等信息从教务网导入，系统根据这些信息得出推荐讲座 |
| 5 | 学生系统 | 查看讲座信息 | 当前/个人的讲座信息 |  |
| 6 | 学生系统 | 报名参加讲座 | 报名成功/失败 | 若名额已满，则学生报名失败 |
| 7 | 学生系统 | 刷校园卡签到 | 签到成功 | 只有在限定的时间内才能成功签到 |
| 8 | 学生系统 | 退选讲座 | 退选成功 |  |
| 9 | 学生系统 | 评价讲座 | 给讲座的打分和评价 | 讲座结束后学生可评价讲座 |
| 10 | 学生系统 | 讨论讲座 | 发帖/回帖 | 学生在讲座的讨论区讨论讲座 |
| 11 | 学生系统 | 讲座支付 | 支付成功  校园卡余额不足 | 对于收取入场费的讲座，通过校园卡系统支付入场费 |
| 12 | 学生系统 | 讲座直播 | 讲座直播视频 | 仅限允许摄像的公开讲座 |
| 13 | 学生系统 | 讲座回放 | 讲座回放视频 | 仅限允许摄像的公开讲座 |
| 14 | 管理员系统 | 导入讲座信息 | 讲座导入成功/失败 | 若讲座信息不合法，则讲座导入失败 |
| 15 | 管理员系统 | 修改讲座信息 | 讲座信息修改成功/失败 | 若修改后信息不合法，则修改失败 |
| 16 | 管理员系统 | 强制报名 | 报名成功 | 强制某些用户报名某个讲座 |
| 17 | 管理员系统 | 强制退选 | 退选成功 | 强制某个用户退选某个讲座 |
| 18 | 管理员系统 | 讲座反馈 | 显示讲座反馈信息  显示学生参与情况 | 讲座结束后根据签到名单和学生评价形成讲座反馈。 |
| 19 | 管理员系统 | 讨论区管理 | 删除/置顶讨论内容，禁言用户 |  |

## NFR和ASR

#### 安全性

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 来自内部/外部的经过了授权/未经过授权的个人或系统 |
| **刺激** | 试图修改/删除数据，访问系统服务，降低系统服务的可用性 |
| **制品** | 系统服务、系统中的数据 |
| **环境** | 在线/离线，联网/断网，连接有防火墙或直接连接到了网络上 |
| **响应** | 对用户进行验证；加密用户的账户信息；阻止未授权用户访问；自动侦测攻击，受到攻击后通知管理员并锁死数据访问 |
| **响应度量** | 未认证用户无法访问数据和发布控制指令  受到攻击后1min内将信息发送给管理员并锁死数据访问  数据被恶意修改/删除后可以在10min内进行恢复 |

## 可用性

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 来自授权的个人用户 |
| **刺激** | 希望进入系统报名参与讲座时系统无法提供正确的反馈 |
| **制品** | 系统的报名参加讲座子系统 |
| **环境** | 联网状态 |
| **响应** | 记录错误日志并从错误中恢复 |
| **响应度量** | 出现此种情况的概率小于1%  记录错误日志并从错误中恢复的时间不多于5s |

## 互操作性

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 系统发起/收到与校园卡系统/教务网系统/现场PC端视频采集系统的请求/响应 |
| **刺激** | 希望与上述的系统进行请求/响应 数据交换 |
| **制品** | 希望进行互操作的上述系统 |
| **环境** | 希望进行互操作的上述系统正在运行 |
| **响应** | 请求被正确获取并成功交换数据 |
| **响应度量** | 交换信息数据成功的概率不低于99% |

## 可修改性

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 开发者 |
| **刺激** | 开发者希望修改系统用户界面、数据标准、控制逻辑等 |
| **制品** | 代码 |
| **环境** | 设计、开发系统时 |
| **响应** | 需要修改的模块被正确的修改，并不影响其他功能的实现 |
| **响应度量** | 每个模块的修改可以在2人月内完成  修改预算不超过总预算的10%  不影响无关的系统功能 |

## 性能

性能场景1：负载

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 访问系统的学生个人用户 |
| **刺激** | 学生用户希望在系统上进行讲座报名操作/查看讲座直播 |
| **制品** | 系统报名参加讲座子系统/讲座直播子系统 |
| **环境** | 联网状态 |
| **响应** | 系统能够正常工作，为每一个用户提供响应 |
| **响应度量** | 系统应该允许500个用户同时进行正常的访问、报名操作/观看讲座直播  用户访问所需页面的时间不超过0.1s |

性能场景2：容量

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 系统 |
| **刺激** | 系统希望进行讲座视频的存储 |
| **制品** | 存储视频数据的系统数据库 |
| **环境** | 运行时 |
| **响应** | 系统正确、完整、及时的存储大量讲座视频数据 |
| **响应度量** | 系统至少能够存储50T的视频数据 |

性能场景3：实时性

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 访问系统的学生个人用户 |
| **刺激** | 学生用户希望在系统上观看讲座直播 |
| **制品** | 系统讲座直播子系统 |
| **环境** | 联网状态 |
| **响应** | 系统直播视频清晰流畅，延时短 |
| **响应度量** | 系统直播延时不超过5s |

## 可维护性

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 系统开发者 |
| **刺激** | 系统开发者希望修改加密密钥 |
| **制品** | 系统用户个人信息模块 |
| **环境** | 系统维护时 |
| **响应** | 及时、方便的进行修改，并对其他模块不造成影响 |
| **响应度量** | 修改时间不超过1人日 |

## 备选设计决策

## 安全性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 对用户密码等信息进行加密 | 保护用户信息的常用方式。 |
| 定期更换密钥 | 更有效的保护用户信息，但是增加维护负担 |
| 在系统服务收到访问时验证用户身份，授权给用户应有的权限 | 有利于保证系统服务仅受到合法访问，但是会增加访问逻辑的复杂度 |
| 自动攻击侦测 | 增加系统服务器负担 |
| 受到攻击后记录攻击日志，提醒管理员并锁住数据 | 影响其他正常用户的可用性 |

## 可用性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 命令/响应 | 一个组件发出一个命令，并希望在预定义的时间内收到一个来自审查组件的响应。与对所有进程发出命令的远程错误探测器相比，这种策略所使用的通信带宽更少 |
| 心跳 | 一个组件定期发出一个心跳信息，另一个组件收听该信息。可以在起到心跳的同时传递数据。 |
| 主动冗余（热启动） | 所有的冗余组件都以并行的方式对事件作出响应。错误发生时，使用该战术的系统的停机时间通常是几毫秒，因为备份是最新的，所以恢复所需的时间就是切换时间。 |
| 被动冗余 | 一个组件（主要的）对事件作出响应，并通知其他组件（备用的）必须进行的状态更新。该战术依赖于能够可靠地接管工作的备用组件。停机时间通常为几秒钟。 |
| 备件 | 备用件是计算平台配置用于更换各种不同的故障组件。当出现故障时，必须将其重新启动为适当的软件配置，并对其状态进行初始化。该战术的停机时间通常为几分钟。 |

## 互操作性（我也不懂啥意思）

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 在提供的服务列表中定位所需要的服务 | 主要针对所需的服务较多且有层次性的情况 |
| 使用控制机制来管理、协调服务的调用 | 针对所要服务的访问需求量较多的情况 |
| 抽取并剪裁接口 | 针对请求响应交换的数据不一致情况 |

## 可修改性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 划分模块 | 可以有效缩小要修改的范围 |
| 抽象通用服务 | 将多个模块都需要的服务提取出来，形成可供多个模块共同使用的通用服务 |
| 添加接口 | 将功能和实现分离，隔离需要更改的模块 |
| 运行时注册 | 支持即插即用操作，但需要管理注册的额外开销 |

## 性能

性能场景1：负载

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 维持计算的多个副本 | 需要考虑如何使副本保持一致和同步 |
| 增加可用资源 | 使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络。但是会增加成本 |
| 区分校内校外网络接入 | 可以充分利用高速廉价的内网资源，但是增加了系统的复杂性 |
|  |  |

性能场景2：容量

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 增加额外的高性能存储资源 | 提高成本 |
| 使用分布式数据库服务器 | 提高了开发和维护的技术门槛，需要保证数据的一致性和正确性 |

性能场景3：实时性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 使用RTMP(Real Time Messaging Protocol)协议 | 适合长时间播放，延时较低，但是又累积延时；是Adobe开发的协议，无法再iPhone中兼容 |
| 使用HLS（ HTTP Live Streaming）协议 | 兼容性比RTMP好，但是延时比其高 |

## 可维护性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 独立存储密钥数据的模块 | 便于密钥变更，既有利于可维护性又有利于安全性 |
| 模块化设计，高内聚低耦合 | 控制变更范围，成熟的设计原则，优点无需多言 |

## 两种备选架构设计

#### SOA

###### 模块视图

###### 组件连接件视图

#### MVC

###### 模块视图

###### 组件连接件视图

## 两种设计的权衡与选择

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | SOA | | MVC | |
|  | pros | cons | pros | cons |
| 安全性 |  | SOA架构的松耦合性和开放性带来安全性的问题：当信任体系存在漏洞或不统一时，可能存在非授权组织非法获取信息的情况。本系统涉及到安全性攸关的校园卡服务，使用SOA会增加安全方案的复杂度。 | 针对用户管理，个人信息和登录注册Model访问数据时加入加密解密逻辑，保证数据安全性。 |  |
| 可获得性 | 在可获得性方面，两种架构均对于报名退选功能设置备份，设置心跳机制监测主模块，心跳中断则切换至备用模块。另一方面，所有数据有冗余备份，以防意外，也便于崩溃发生后的恢复。在可获得性的方面，两种架构没有特别大的差别 | | | |
| 互操作性 | 本系统涉及到即将建设的系统和已存在的校园卡系统和教务系统，可能存在平台和异构问题。选择接口而不是语言具体的类以及基于消息交互，极大的提高了系统的互操作性。 |  |  | 在MVC架构中，费用支付Model，签到Model，讲座推荐Model均需要直接使用到外部系统提供的接口。可能存在接口难以使用的情况。 |
| 可修改性 | SOA不同服务之间保持了一种无依赖的低耦合关系；服务本身是通过统一的接口定义语言来描述具体的服务内容，并且很好地封装了底层的具体实现。所以，当发生修改时，服务之间不会影响，并且低层实现不会影响调用。 |  | MVC架构耦合性低，MVC的三个部件是相互独立，改变其中一个不会影响其它两个，这样具有良好的可修改性。 | 本系统具有与教务系统和校园卡交互的接口，当外部接口定义发生变化时，需要修改本系统的Model的实现逻辑。 |
| 负载 | 两种架构均采用相同的负载策略：对于报名退选功能设置备份，对于直播功能设置内外网两种接入策略等。两种架构没有特别大的差别。 | | | |
| 容量 | 两种架构均采用分布式数据库来保证容量，没有特别大的差别。 | | | |
| 实时性 |  | SOA架构的性能稍低，主要是因为SOA的分布性质和web服务协议的开销。 | MVC架构在实时性上没有特别的表现。 |  |
| 可维护性 | 服务带来低耦合性和很好的模块性。 |  | 分离视图层和业务逻辑层也使得WEB应用更易于维护和修改。 |  |

两种模式在满足ASR方面没有大的差别，所以我们最终选择了团队成员较为熟悉、开发起来成本较低的MVC模式。