Sprint 回顾会议记录摘要（Run-Max 项目 - 客户分数排名服务）

## 一、会议基础信息

- 会议主题：Run-Max 项目第1个Sprint回顾（聚焦 .NET Core 客户分数排名服务开发）

- 会议时间：2025年10月1日 4：30-5：00

- 参会人员：Rhys-Dai（测试leader）、Oscar（后端开发工程师）、Li Xuanye（后端开发工程师）、Luo longjie（后端开发工程师）

- 核心目标：完成「客户分数更新」「排名查询」接口开发及基础内存数据结构设计，适配 bootcamp.pdf 需求

二、Sprint成果回顾

1. 已完成工作：

- 实现 `Update Score` 接口核心逻辑（客户新增、分数增减、分数≤0时移除），需求文档业务规则；

- 设计 `Dictionary+SortedSet` 内存数据结构，支持客户ID快速查询与排名实时排序；

- 完成 GitHub 仓库（tick111/Run-Max）初始化，上传基础代码

- 执行功能测试用例32条，修复高危Bug 4个（如参数校验异常、排名计算错误）。

2. 未完成工作：

- `Get by customerid` 接口（查询客户及周边排名）开发滞后，因“排名定位算法优化”耗时超预期；

- 高并发压力测试未启动，受 Git 推送同步问题（远程代码冲突）影响，测试环境代码未及时更新。

三、关键亮点

1. 技术方案落地：

- 提前调研 `SortedSet` 自定义比较器，确保“分数降序+ID升序”排名逻辑一次通过，减少返工；

- 针对 需求文档“无持久化”要求，采用内存双结构设计，接口平均响应时间≤200ms，满足性能预期。

2. 协作效率：

- 每日站会聚焦“阻塞问题”，快速解决2次 Git 权限问题（Rhys-Dai 账号无推送权限，dev小组在pull主分支成功后无法push到各自的分支上面），保障开发进度；

- 测试提前介入需求评审，输出接口测试用例框架，开发过程中同步执行冒烟测试，提前暴露3个逻辑漏洞。

四、待改进问题与解决方案

|  |  |  |  |
| --- | --- | --- | --- |
| **问题类型** | **具体问题描述** | **解决方案** | **负责人** |
| 开发效率 | Get by customerid 接口排名定位算法耗时超预期 | 1. 引入 “排名缓存列表” 优化查询（每 10 秒同步 SortedSet 数据）；2. 参考开源项目排名定位方案，减少算法复杂度 | Oscar |
| 版本控制 | 多次出现 Git 推送冲突（远程代码未同步） | 1. 每日开发前强制执行 git pull origin main；2. 开发新功能创建 feature 分支（如 feature/rank-query），避免直接在 main 分支修改 | 全体开发及测试 |
| 测试进度 | 根据需求文档编写基本功能模块测试用例 | 代码开发处于初期阶段，根据开发push的代码先进行单元测试 | Rhys |

五、行动计划（下个Sprint优化）

1. 技术层面：优先完成 `Get by customerid` 接口开发，集成排名缓存优化方案；启动高并发测试（模拟1000次/秒请求），验证内存数据结构稳定性。

2. 协作层面：制定《Git 分支管理规范》，并在团队内同步培训；每日下班前10分钟，开发同步代码提交状态，避免冲突堆积。

3. 风险预案：提前与 GitHub 仓库管理员确认权限（避免再次出现403错误）；准备备用测试环境，防止主环境代码同步问题阻塞测试。

六、会议决议

- 下次会议需重点跟进：排名缓存方案效果、Git 规范执行情况、集成测试实现。