项目需求分析会议纪要

## 一、会议基本信息

- \*\*会议名称\*\*：客户分数排名后端服务项目需求分析会

- \*\*会议时间\*\*：2025年10月1日3：00-3：30

- \*\*会议地点\*\*：线上会议（teams）

- \*\*主持人\*\*：Rhys,Dai（测试leader）

- \*\*参会人员\*\*：

- 技术部：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）

- 测试部：Rhys,Dai（测试负责人）

- \*\*记录人\*\*：Rhys,Dai（测试负责人）

- \*\*会议议题\*\*：

1. 解读开发文档中项目核心业务需求与规则

2. 明确非功能需求的落地标准与约束条件

3. 梳理功能需求中接口设计的细节与边界场景

4. 确认需求理解偏差，形成统一需求认知

## 二、会议核心内容

### （一）业务需求与规则解读

1. \*\*核心业务模型确认\*\*（Oscar主导）：

- 明确客户标识规则：客户通过唯一的 int64 类型 CustomerID 识别，无其他身份约束，任意正整数均可作为有效 ID 。

- 分数规则同步：客户初始分数默认为 0，分数为 decimal 类型；仅分数＞0 的客户参与排行榜竞争，分数≤0 时需移除出排行榜 。

- 排名逻辑共识：排行榜按“分数降序→CustomerID 升序”排序，分数最高者排名 1，分数相同则 CustomerID 较小的客户排名更靠前；客户分数实时更新时，排名需同步实时调整。

- 结合文档示例验证：以文档中 10 个客户的分数与排名数据为例（如 CustomerID 15514665 分数 124 排名 1，CustomerID 76786448 与 1745431 分数均为 100 但 ID 小的排名更靠前），确认排名逻辑无理解偏差 。

2. \*\*争议点讨论与结论\*\*：

- 争议：客户分数更新后变为 0 时，是否保留在内存中？

- 结论：参考需求“仅分数＞0 参与竞争”，分数=0 时客户不参与排行榜，但无需从内存中删除（仅分数＜0 时彻底移除），后续若分数再次更新为正，可直接恢复参与排名，无需重新创建客户数据。

### （二）非功能需求落地标准

1. \*\*技术栈与存储约束\*\*（Luo longjie）：

- 明确技术框架：仅限使用 .NET Core 开发。

- 存储方案确认：无需持久化，所有客户数据、分数及排名仅保存在内存中，服务重启后数据可丢失，无需设计数据恢复机制 。

2. \*\*交付要求\*\*：

- 代码需上传至 GitHub 仓库，仓库需包含完整源代码（含 Controller、实体类、数据处理逻辑等）、接口文档（README 中说明）及测试用例，确保其他成员可通过仓库地址直接拉取代码并本地运行 。

### （三）功能需求接口细节梳理（Li Xuanye主导）

1. \*\*Update Score 接口（POST /customer/{customerid}/score/{score}）\*\*：

- 参数约束：

- CustomerID：需校验为正 int64 类型，若为负或非数字格式，返回 400 错误（提示“CustomerID 必须为正整数”） 。

- Score：调整值范围为 [-1000, +1000]，正数表示分数增加，负数表示分数减少；若超出范围，返回 400 错误（提示“分数调整范围需在[-1000,1000]内”） 。

- 业务逻辑：

- 客户不存在时：调用接口后自动新增客户，初始分数为本次调整的分数（如调整值为 80，则初始分数 80） 。

- 客户已存在时：分数直接叠加调整值（如原分数 80，调整值-20，最终分数 60），无需支持幂等性（重复调用可能导致分数重复增减） 。

- 响应格式：需返回“更新后的当前分数”，建议补充返回当前排名，便于前端直接展示（如响应体为 `{"customerId": 15514665, "currentScore": 124, "currentRank": 1}`） 。

2. \*\*Get by rank 接口（GET /leaderboard?start={start}&end={end}）\*\*：

- 参数规则：start 为起始排名（包含），end 为结束排名（包含），需校验 start≤end 且均为正整数，否则返回 400 错误；若查询范围超出实际存在的排名（如仅 10 个客户却查询 start=11），返回空数据列表 。

- 性能约束：严格遵循需求“避免在该 API 内排序”，需提前维护已排序的客户列表，接口仅负责按排名区间截取数据，确保响应高效 。

- 响应示例：参考文档中“start=41、end=44”的返回格式，需包含 CustomerID、Score、Rank 三个字段，以 JSON 数组形式返回 。

3. \*\*Get by customerid 接口（GET /leaderboard/{customerid}?high={high}&low={low}）\*\*：

- 参数规则：

- CustomerID：需校验是否存在且分数＞0，不存在或分数≤0 时返回 404 错误（提示“客户不存在或分数≤0，未参与排行榜”） 。

- high/low：可选参数，默认值为 0，需校验为非负整数，若为负数返回 400 错误；high 表示“排名高于该客户的邻居数量”，low 表示“排名低于该客户的邻居数量” 。

- 响应示例：参考文档中“customerid=7777777、high=2、low=3”的返回格式，需包含目标客户及上下游指定数量的客户数据，按排名顺序排列 。

- 性能约束：同 Get by rank 接口，避免接口内排序，依赖提前维护的排序列表快速定位客户及邻居 。

记录人签名\*\*：Rhys，Dai

\*\*成文日期\*\*：2025年10月1日