一、CMMI层次成熟度模型简述

CMMI（Capability Maturity Model Integration）是衡量组织软件开发过程成熟度的框架，分为5个层级：

1. 初始级（Level 1：Initial）

- 过程不可预测，依赖个人能力，无标准化流程。

- 项目常出现预算超支、延期和质量问题。

2. 可重复级（Level 2：Managed）

- 建立基本项目管理：需求管理、计划跟踪、配置管理。

- 能复用过往成功经验，但过程仍以项目为单位。

3. 已定义级（Level 3：Defined）

- 组织级标准化流程（如设计模板、编码规范）。

- 过程可量化，并通过培训确保一致性。

4. 量化管理级（Level 4：Quantitatively Managed）

- 基于数据的决策：通过统计方法控制过程质量（如缺陷率、生产率）。

- 设定量化目标并持续监控。

5. 优化级（Level 5：Optimizing）

- 持续改进：利用数据优化流程，主动预防问题。

- 技术创新制度化（如引入自动化测试、AI辅助开发）。

---

二、您的项目过程成熟度评估

基于项目文档（信息摘要、设计文档），评估结果如下：

成熟度：CMMI Level 2（可重复级）

符合Level 2的特征：

1. 需求管理

- 文档中明确划分三类用户需求（老人/子女/社区端），并细化功能模块（如健康监控、服务预约）。

- 进行了竞品分析（对比健康监测、社区管理类产品），定位差异化优势。

2. 项目计划与跟踪

- 任务分工清晰：四名成员按比例承担组织协调（20%~40%）、技术实现（20%~30%）等职责（信息摘要表）。

- 技术选型明确：Vue.js（前端）、Spring Boot（后端）、PyTorch（AI模型）。

3. 配置管理

- 提交材料规范：源代码、模型、演示视频等分类上传至网盘（信息摘要表）。

- 版本控制：设计文档标注版本号（V1.0）。

4. 质量保证

- 测试覆盖功能/性能/算法：如跌倒检测（YOLOv8）、菜品识别（MobileNetV2）通过测试（测试报告）。

未达到Level 3及以上：

1. 缺乏组织级标准流程

- 设计文档未提及编码规范、UI设计标准或代码审查机制。

- 技术实现描述笼统（如“使用ECharts实现可视化”未说明具体规范）。

2. 量化管理不足

- 测试报告仅标注“通过”✅，未提供缺陷密度、测试覆盖率等量化指标（如算法准确率91.4%未与需求目标对比）。

- 性能需求模糊（如“实时性”未定义具体延迟阈值）。

3. 风险管理缺失

- 未识别技术风险（如AI模型依赖DeepSeek API的稳定性）、数据安全风险（健康隐私保护）。

4. 持续改进机制空白

- 项目总结仅描述成果，未分析过程短板（如未提及蓝牙同步数据时的具体问题解决过程）。

---

三、过程改进计划

基于当前成熟度（Level 2），目标提升至Level 3（已定义级），分阶段改进：

阶段1：建立组织级标准流程（3-6个月）

| 改进领域 | 具体措施 | 产出物 |

|--------------------|----------------------------------------------------------------------------|------------------------------|

| 需求工程 | 制定需求模板（含用户场景、验收标准） | 《需求规格说明书模板》 |

| 技术规范 | 定义前端（Vue组件规范）、后端（RESTful API设计）、数据库（索引/分区标准） | 《技术开发规范手册》 |

| 配置管理 | 从网盘迁移至GitLab，建立分支策略（main/dev/feature） | 《Git操作指南》+ 代码仓库示例 |

| 评审机制 | 引入设计/代码评审会（每周1次），使用Checklist审查关键模块 | 《评审记录表》+ 缺陷跟踪日志 |

阶段2：推行量化管理（6-12个月）

| 改进领域 | 具体措施 | 产出物 |

|--------------------|----------------------------------------------------------------------------|------------------------------|

| 度量指标 | 定义核心指标：<br> - 代码缺陷率（每千行BUG数）<br> - 测试覆盖率（≥80%）<br> - 构建失败率 | 《度量指标定义表》 |

| 过程监控 | 使用Jira+Confluence跟踪任务进度，仪表盘展示需求完成率、BUG趋势 | 项目实时看板 |

| 算法质量管控 | 要求算法测试报告包含混淆矩阵、ROC曲线，设定准确率阈值（如菜品识别≥90%） | 《算法验收标准》 |

阶段3：持续优化（12个月+）

| 改进领域 | 具体措施 | 产出物 |

|--------------------|----------------------------------------------------------------------------|------------------------------|

| 自动化 | 部署CI/CD流水线（Jenkins），自动化单元测试/API测试 | 构建流水线配置脚本 |

| 根因分析 | 针对重大缺陷召开回溯会议（如跌倒检测误报），输出预防措施 | 《故障分析报告》模板 |

| 技术债管理 | 每季度评估技术债（如老旧库升级），分配20%工时偿还 | 《技术债清单》+ 优化路线图 |

---

四、改进案例：针对当前项目的实践建议

1. 风险管理

- 风险库建立：列出“AI服务依赖第三方API”（DeepSeek）、“蓝牙连接稳定性”等风险，制定备用方案（如本地缓存模型）。

- 安全合规：参考《个人信息保护法》，在设计中加入数据匿名化处理（如健康报告去标识化）。

2. 量化测试改进

- 原测试报告：仅写“性能测试通过” → 改进后：

```markdown

性能测试结果

- 并发能力：支持1000用户同时在线（目标：≥800）

- 数据同步延迟：< 2s（手环→App端）

- API响应P99：≤300ms

```

3. 持续改进文化

- 在项目总结（第六章）中增加“过程改进建议”栏目，例如：

> “蓝牙数据同步初期失败率40%，通过引入重试机制+压缩协议降至5%。后续建议：所有硬件交互模块需预设超时重试策略。”

---

五、总结

您的项目已具备可重复级（Level 2）的核心能力（需求管理、任务分工、基础测试），但需系统化构建组织级标准（Level 3）并逐步引入量化管理（Level 4）。改进核心在于：

1. 标准化：将个人经验沉淀为团队规范（设计/编码/测试）；

2. 数据驱动：用量化指标替代主观判断（如缺陷率、延迟）；

3. 制度化改进：定期回溯问题，将优化措施纳入流程。

通过上述计划，团队可从“项目可控”迈向“过程可预测”，最终实现持续技术创新（Level 5）。