**基于 CMMI 的失物招领系统开发过程评估与改进计划**

**一、CMMI 成熟度模型简述**

CMMI（Capability Maturity Model Integration，能力成熟度模型集成）是国际上广泛采用的软件过程改进框架。它将组织的软件开发能力划分为五个层次：

1. **Level 1：初始级（Initial）**  
   无固定过程，开发行为随意，严重依赖个人能力，风险不可控。
2. **Level 2：可管理级（Managed）**
3. 项目具备基本的管理能力，计划、进度、需求、质量均可追踪，但仅限于项目级别，尚未标准化。
4. **Level 3：已定义级（Defined）**  
   有组织级标准流程，每个项目在统一框架下执行。
5. **Level 4：量化管理级（Quantitatively Managed）**  
   建立量化目标和度量体系，对过程进行统计控制。
6. **Level 5：优化级（Optimizing）**  
   通过持续改进和创新手段优化整个开发流程。

**二、“失物招领系统”开发过程成熟度评估（Level 1）**

该系统采用前后端分离架构：Spring Boot + MyBatis + Vue + Element Plus，具备用户注册登录、物品发布、留言板、搜索、数据可视化等功能。虽然功能较为完备，但从 CMMI 的角度分析，该项目仍处于 **Level 1（初始级）**，原因如下：

**1. 无正式流程控制**

项目开发周期主要依赖个人安排，缺乏规范化的流程指导。

无任务分配记录或开发日程表，阶段性目标未明确设定。

**2. 文档与规范缺失**

没有统一的开发手册、接口文档或数据字典，代码可读性依赖命名直觉。

数据库字段与前端字段映射靠记忆或手动维护，易出错。

**3. 测试和质量保障不足**

项目无系统测试用例，仅通过运行调试进行测试。

无单元测试、接口测试、性能测试或异常测试覆盖。

**4. 版本管理松散**

使用 Git 但未做分支策略管理，所有提交集中在 main/master 分支。

无版本标记，无法回退至特定状态或区分迭代进度。

**5. 项目过程高度依赖开发者个人经验和主动性**

无团队协作记录，需求常常边做边改，未经过清晰梳理与评审。

缺乏缺陷跟踪机制，Bug 修复靠即时发现即时解决。

**三、改进目标：从 Level 1 到 Level 2（可管理级）**

为了将项目过程从“初始级”提升至“可管理级”，我们需要在以下几个方面做出改进：

**四、过程改进计划（目标：从 Level 1 提升至 Level 2）**

为了将“失物招领系统”的开发过程从 CMMI Level 1（初始级）提升到 Level 2（可管理级），我从五个方面制定了详细的改进计划，包括项目计划制定、文档体系建设、版本控制优化、基础测试建立和缺陷追踪机制。

**（1）建立项目计划和任务拆解机制**

改进的第一步，是在开发之初制定明确的开发计划。包括划分功能模块、设定开发周期、确定各阶段的目标（如第 1 周完成用户注册模块，第 2 周完成物品发布模块等）。在实际执行过程中，我计划使用飞书日历或 GitHub Project（看板工具）来进行任务跟踪。每项功能会被拆分为若干小任务，以 Issue 形式记录，分配优先级并依照时间推进。这有助于提升计划性，防止项目节奏失控。

**（2）完善基础开发文档体系**

为保障项目的可维护性和后续复用性，必须建立一套完整的开发文档体系。首先，针对数据库结构，我会用 Markdown 或 Excel 详细记录每一张表的字段含义、约束、关系等内容。其次，前后端接口将统一整理为接口说明文档，内容包括接口路径、请求方式、参数格式、响应示例等，可以使用 Postman 的导出格式或 Markdown 编写，也可以使用 YAPI 等工具管理。最后，补充一份部署说明文档，记录运行环境要求、启动命令、常见错误与解决方法，以便部署与迁移。

**（3）引入规范的版本管理策略**

当前项目虽然使用 Git 进行版本管理，但缺乏明确的分支策略与提交规范。未来我将采用 “main + dev + feature” 分支策略，main 用于保存稳定代码，dev 作为日常开发分支，而每一个功能模块创建独立 feature/xxx 分支。在功能开发完成后合并回 dev，并通过 Pull Request 进行简单的代码审查。此外，每次阶段性功能完成后，我会为主分支打上版本标签（如 v1.0、v1.1），便于问题回溯和功能对照。

**（4）引入基础测试机制，保障软件质量**

为了提升系统的稳定性和可维护性，我会开始补充基础的测试流程。后端方面，针对核心服务逻辑，如用户注册、物品发布、留言等功能，使用 JUnit5 + Mockito 编写单元测试；接口层使用 Postman 编写接口测试集合，测试用户登录、搜索物品等功能的输入输出；前端交互部分则至少手动执行完整流程测试，并记录测试用例和期望结果。此外，将探索在提交或发布代码前自动执行测试脚本，降低人工测试成本。

**（5）建立缺陷追踪与项目复盘机制**

每当项目中出现 Bug，不应只是在代码中快速修复了事，而是应建立缺陷记录机制。我计划使用 GitHub Issues 记录每一次 Bug，包括触发条件、影响范围、修复时间和处理人。这样不仅有助于后期回顾，也方便对缺陷进行统计和分析。项目开发结束后，还会撰写总结文档，反思项目中存在的问题，提出优化建议，并沉淀可复用的经验，如模板代码、错误处理方案等。

**五、未来展望与总结**

虽然目前“失物招领系统”的开发过程处于 CMMI Level 1，但通过本次分析，我已明确提升路径。首先建立最基本的项目管理、文档编写和测试流程，在多个项目中复用这套方法，逐步沉淀经验，并向 Level 2 稳步迈进。

随着技能的提升与项目复杂度的增加，未来将尝试接入 CI/CD、引入接口自动化测试、组织级代码风格统一等措施，以实现从“项目可控”到“过程优化”的转变，为更高级别的成熟度（Level 3 及以上）打下坚实基础。