软件工程化说明文档

软件过程说明

项目名称: Yas

项目目标

Yas 是一个用于识别和导出《原神》圣遗物及《崩坏：星穹铁道》遗器信息的工具。它通过模拟人工操作进行识别，并采用 SVTR 字符识别模型，实现高效、准确的字符识别。支持 Windows, Linux 和 macOS 操作系统。

一、开发流程

1. **需求分析**
   - 确定项目需求，详细记录每个功能模块的具体要求。
   - 确定项目的技术栈和工具链。
2. **设计阶段**
   - 设计项目的整体架构，包括模块划分、数据流和控制流。
   - 详细设计每个模块的功能和接口。
3. **实现阶段**
   - 根据设计文档，开始代码编写。
   - 每个功能在完成后，提交到相应的功能分支。
4. **测试阶段**
   - 编写单元测试和集成测试。
   - 使用 CI 自动化测试工具进行持续集成测试。
5. **部署阶段**
   - 通过 CI/CD 工具进行自动化部署。
   - 在各操作系统环境下进行兼容性测试。
6. **维护阶段**
   - 监控系统运行情况，收集用户反馈。
   - 定期更新和优化系统。

二、角色和职责

1. **项目经理**
   - 负责项目整体规划和进度控制。
   - 协调各部门之间的沟通和合作。
2. **开发人员**
   - 负责具体功能模块的开发和实现。
   - 编写单元测试和集成测试。
3. **测试人员**
   - 负责系统的全面测试，包括功能测试、性能测试和兼容性测试。
   - 提交 bug 报告和改进建议。

4. **运维人员**
    - 负责系统的部署和维护。
    - 监控系统运行情况，及时处理突发问题。

三、标准和规范

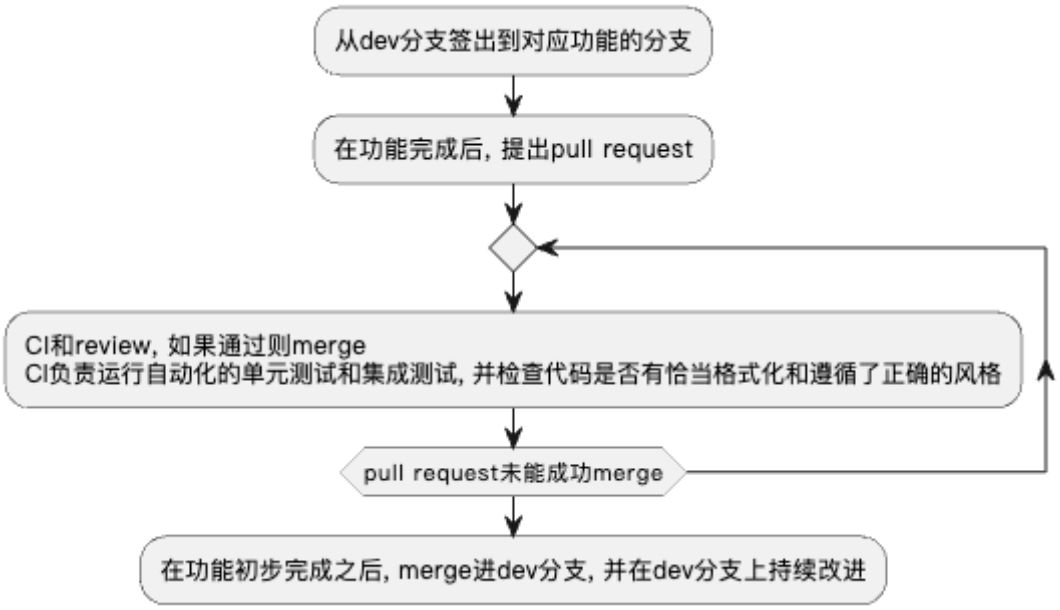
1. **编码规范**
    - 遵循 Rust 编码规范，代码需格式化整齐，注释清晰。
    - 所有代码变更需通过代码审查，确保代码质量。
2. **文档规范**
    - 所有功能模块需附带详细的设计文档和使用说明。
    - 每次代码提交需更新相关文档，保证文档与代码一致。
3. **测试规范**
    - 每个功能模块需编写单元测试，覆盖率需达到 90%以上。
    - 所有测试需通过 CI 工具自动执行，保证代码的稳定性和可靠性。

四、工具和技术

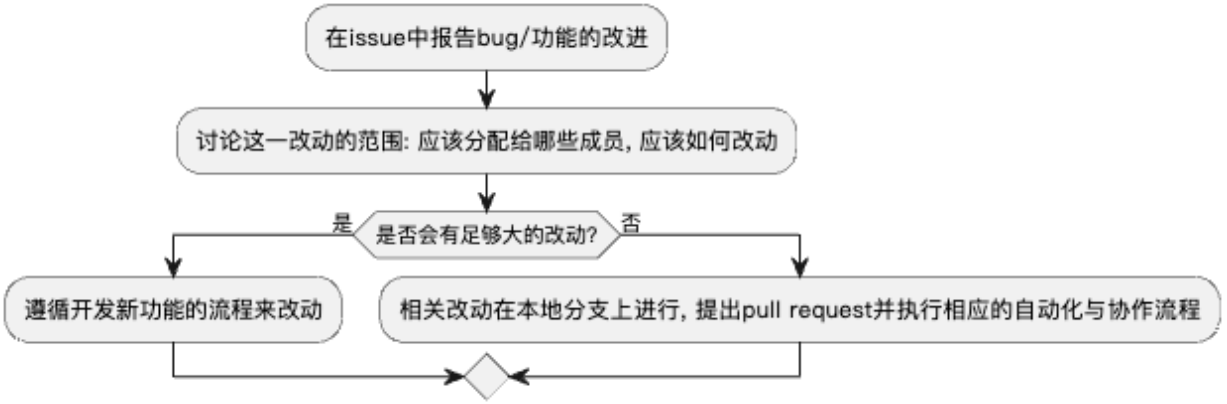
1. **版本控制**
    - 使用 Git 进行版本控制，所有代码提交需遵循 Git 工作流程。
    - 采用 monorepo 风格，将所有模块都放在同一个 Git 仓库中。
2. **CI/CD**
    - 使用 GitHub Actions 进行持续集成和持续部署。
    - 编写自动化构建和测试脚本，确保每次提交都能顺利构建和通过测试。
3. **包管理**
    - 使用 Rust 的 cargo 工具进行包管理，追踪和维护项目的所有依赖。
4. **测试工具**
    - 使用 Rust 的内置测试工具进行单元测试和集成测试。
    - 通过 `cargo test` 命令运行所有测试。

五、工作流程

1. **新功能开发流程**



2. **现有功能改进和bug修复流程**



六、沟通和协作

1. **沟通工具**
   - 使用微信进行实时沟通和讨论。
   - 定期召开线上会议，汇报工作进展，讨论项目问题。
2. **协同平台**
   - 使用 GitHub 进行版本管理和代码托管。
   - 使用 GitHub Issues 和 Projects 进行任务管理和进度跟踪。
3. **文档管理**
   - 使用腾讯文档记录项目的详细设计和进展情况。
   - 所有文档需实时更新，确保信息的及时性和准确性。

七、质量保证

1. **代码审查**
   - 所有代码变更需通过 Pull Request 提交，并由团队成员进行代码审查。
   - 确保代码质量和一致性。
2. **自动化测试**
   - 所有功能模块需编写单元测试和集成测试，确保代码的稳定性和可靠性。

- 使用 CI 工具自动执行测试，保证每次提交都能顺利通过。

3. **定期更新**
   - 根据用户反馈和项目需求，定期更新和优化系统。
   - 确保系统的稳定性和性能。

八、发布流程

1. **预发布**
   - 在 dev 分支上的提交在成功构建时打包生成对应的可执行文件，生成 pre-release。
2. **正式发布**
   - 在 main 分支上的提交在成功构建时生成一个正式的 release 并进行分发。