# 质量计划

# 引言

本质量管理计划的编制目的是为云南省人力资源市场数据采集系统项目的升级改造提供质量保证和控制，确保项目按时交付、满足用户需求，并提高系统的稳定性、性能和可靠性。通过本计划，将明确质量目标和要求，建立质量管理组织架构，确定质量确保对象，并制定相应的质量控制措施。

# 质量目标和要求

## 质量目标

为了确保项目质量，我们制定了以下质量目标：

**可用度：**

软件运行后在任一随机时刻需要执行规定任务或完成规定功能时软件处于可使用状态的概率应不大于98%。

**初期故障率：**

软件交付后三个月内每100小时故障率不大于5%。

**偶然故障率：**

软件交付后三个月内每1000小时故障率不大于2%。

**平均失效时间：**

在失效前正常工作的平均统计时间应大于365天。

**平均失效间隔时间：**

在相继两次失效之间正常工作的平均统计时间大于200天。

**缺陷密度：**

开发阶段为50－60缺陷，交付后为15－18缺陷。

**平均失效恢复时间：**

软件失效后恢复正常工作所需的平均统计时间。

## 完成目标所需方案

为了实现以上质量目标，我们将采取以下方案：

在测试部分，将给予足够的压力测试，以验证系统的可用性和稳定性。

售后部分将跟踪软件质量问题，及时解决问题，确保客户满意度。

项目经理将全程管控开发流程，合理分配时间，避免时间不足导致的软件质量问题。

# 质量管理组织架构

项目经理负责项目规划、执行、监控和结束的各个阶段。配置管理员负责配置项的标识和控制、配置项变更管理、配置库管理和支持项目管理。开发人员负责软件编码、测试和维护，以及技术研究和团队协作。具体见人力管理计划。

# 质量确保对象

## 文档对象

### 文档质量保证对象

文档质量保证对象包含：

《需求规格说明书》

《质量保证计划》

《配置管理计划》

《成本计划》

《人力资源计划》

《沟通计划》

《软件设计文档》

《风险计划》

### 文档质量衡量标准

**准确性：**

文档内容应准确无误，避免错误和误导。

**完整性：**

文档内容应包含所需的全部信息，不应漏掉任何重要细节。  
**适应性：**

文档应当适应不同读者的需求和背景。考虑到读者的知识水平和背景差异，提供相应的解释、说明或附加材料，以确保文档对各类读者都具有适应性。

**更新性：**

文档应当及时更新以反映最新的信息和变化。随着时间的推移和项目的演进，确保文档的内容保持最新和准确，通过定期审查和更新来保持文档的更新性。

**一致性：**

文档应当保持一致性，即在整个文档中使用一致的术语、表达方式和格式。避免使用相互矛盾或混淆的术语，确保文档中的信息和表述保持一致，以提高读者的理解和阅读体验。

**可追溯性：**

文档应当具备可追溯性，即读者能够追溯文档中所引用的信息的来源和依据。为了确保文档的可信度和透明度，提供引用和参考的信息来源，以便读者能够进一步深入了解和核实所述内容。

**易更新性：**

文档应当易于进行修改和更新。为了满足不断变化的需求和反馈，确保文档具备易于编辑和修改的特性，使得更新文档变得简便和高效。

**清晰度：**

文档应该使用简单明了的语言，避免使用模棱两可或歧义的表述。

**可读性：**

文档应该易于阅读和理解，包括字体大小、排版、图表等方面的设计。

**可理解性：**

文档应该易于理解，包括专有术语的解释、概念的阐述等。

**可靠性：**

文档应该可靠，内容应该基于实际事实和数据，避免虚假或不准确的信息。

**可用性：**

文档应该易于使用，包括易于查找、易于导航等方面的设计。

## 代码对象

### 代码质量保证对象

代码质量保证对象包括项目中的所有软件，包括.html文件、.css文件、.js文件、.json文件、.xml文件、.dll文件等。这些代码应具备可靠性、可读性、可维护性、性能效率和安全性等标准。

### 代码质量衡量标准

**可靠性：**

代码质量应确保代码的执行稳定，没有明显的逻辑错误和异常情况。通过进行彻底的测试和调试，包括单元测试、集成测试和系统测试等，以验证代码的可靠性和稳定性。

**可读性：**

代码应具备良好的可读性，即代码结构清晰，命名规范，注释清晰，易于理解和维护。使用有意义的变量和函数命名，提供清晰的注释和文档，遵循一致的代码风格和格式，以使代码易于阅读和理解。

**可维护性：**

代码应具备良好的可维护性，即代码应该模块化，低耦合高内聚，易于修改和扩展。通过良好的设计原则和设计模式，将代码拆分为模块，使得每个模块都有清晰的职责和接口，并且修改一个模块不会影响其他模块的功能。

**性能效率：**

代码应具备高性能效率，即代码执行效率高，响应时间短，资源利用合理。通过优化算法和数据结构的选择，避免冗余计算和资源浪费，以提高代码的执行效率和性能。

**安全性：**

代码应具备必要的安全防护措施，防范潜在的安全风险和漏洞。使用安全编码的最佳实践，避免常见的安全漏洞，如SQL注入、跨站脚本攻击等。实施访问控制和数据验证等安全机制，以保护代码和系统的安全性。

**可靠性：**

代码质量应确保代码的执行稳定，没有明显的逻辑错误和异常情况。通过进行彻底的测试和调试，包括单元测试、集成测试和系统测试等，以验证代码的可靠性和稳定性。

**可读性：**

代码应具备良好的可读性，即代码结构清晰，命名规范，注释清晰，易于理解和维护。使用有意义的变量和函数命名，提供清晰的注释和文档，遵循一致的代码风格和格式，以使代码易于阅读和理解。

**可维护性：**

代码应具备良好的可维护性，即代码应该模块化，低耦合高内聚，易于修改和扩展。通过良好的设计原则和设计模式，将代码拆分为模块，使得每个模块都有清晰的职责和接口，并且修改一个模块不会影响其他模块的功能。

**性能效率：**

代码应具备高性能效率，即代码执行效率高，响应时间短，资源利用合理。通过优化算法和数据结构的选择，避免冗余计算和资源浪费，以提高代码的执行效率和性能。

**安全性：**

代码应具备必要的安全防护措施，防范潜在的安全风险和漏洞。使用安全编码的最佳实践，避免常见的安全漏洞，如SQL注入、跨站脚本攻击等。实施访问控制和数据验证等安全机制，以保护代码和系统的安全性。

# 质量计划

## 需求分析阶段的质量控制措施

在需求分析阶段，计划采取如下措施来确保质量控制：

**需求验证：**

与客户和利益相关者确认需求，确保需求的准确性和完整性。

**需求文档审查：**

对需求文档进行严格的审查和验证，确保文档清晰、一致、可追踪。

**需求变更管理：**

建立变更控制流程，对需求变更进行评估和控制，避免对开发过程造成不必要的干扰。

## 设计阶段的质量控制措施

在设计阶段，计划采取如下措施来确保质量控制：

**设计审查：**

对系统设计进行技术审查，确保设计符合项目要求和最佳实践。

设计文档质量控制：

对设计文档进行审查，确保文档准确、完整、易读，与需求一致。

**设计复审：**

组织设计团队进行复审，识别潜在的问题和改进点，并进行必要的调整和优化。

## 编码和测试阶段的质量控制措施

在编码和测试阶段，计划采取如下措施来确保质量控制：

**编码规范和标准：**

制定统一的编码规范和标准，确保代码的可读性、可维护性和一致性。

**代码审查：**

进行代码审查，发现和修复潜在的缺陷和错误，提高代码质量。

**自动化测试：**

建立自动化测试框架，执行单元测试、集成测试和系统测试，确保代码质量和功能的正确性。

## 上线和运维阶段的质量控制措施

在上线和运维阶段，计划采取如下措施来确保质量控制：

**上线前测试：**

进行系统集成测试和用户验收测试，确保系统的稳定性和完整性。

**系统监控和日志管理：**

建立系统监控和日志管理机制，及时发现和解决问题，保障系统的正常运行。

**定期维护和更新：**

制定维护计划，定期进行系统维护和更新，修复漏洞和改进功能，确保系统持续稳定运行。

# 质量检查和审核

质量检查和审核将在不同阶段进行，审计内容包括需求分析、设计、编码、测试和项目验收等。审计报告将通过内部会议和讨论、缺陷跟踪工具、文件和报告、项目管理工具等方式提交和记录。质量保证人员将与项目经理、高层管理者和其他相关人员合作，确保发现的问题得到纠正和改进。