ICS
CCS 点击此处添加 CCS 号

团　　体　　标　　准

T/QGCML XXXX—XXXX

设备售后维修服务管理平台技术规范

Technical specifications for equipment after-sales maintenance service management platform

XXXX－XX－XX 发布　　XXXX－XX－XX 实施

全国城市工业品贸易中心联合会　　发 布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由××××提出。

本文件由××××归口。

本文件起草单位：

本文件主要起草人：

本文件为首次发布。

设备售后维修服务管理平台技术规范

1 范围

本文件规定了设备售后维修服务管理平台技术规范的术语和定义、技术要求、功能要求、安全要求、测试方法。

本文件适用于设备售后维修服务管理平台的设计和检验。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

3 术语和定义

下列术语和定义适用于本文件。

3.1

设备售后维修服务管理平台 Equipment after-sales maintenance service management platform

指用于管理设备售后维修服务的信息化平台，包括设备报修、维修派工、维修进度跟踪、维修记录管理、备件管理、客户反馈等功能。

4 技术要求

4.1 系统架构

4.1.1 平台应采用模块化设计，将不同功能划分为独立的模块，每个模块之间通过标准化的接口进行通信，以实现模块的独立开发和升级。

4.1.2 平台应具备良好的可扩展性，能够随着业务的发展和变化，方便地进行功能扩展和性能提升。这要求平台能够支持新的模块和组件的集成，同时不影响现有模块的稳定运行。

4.1.3 对于大型系统或高并发的应用场景，平台应采用分布式部署方案，将不同功能模块部署在不同的服务器上，以实现负载均衡和故障隔离。

4.2 数据库设计

4.2.1 数据库设计应确保数据的完整性和准确性，通过合理的数据表结构和约束条件，避免数据冗余和错误。

4.2.2 数据库应支持数据加密、备份和恢复等安全措施，以保护数据不被非法访问和篡改。

4.2.3 数据库应具备良好的读写性能和查询性能，以满足大量数据的快速处理和查询需求。

4.3 用户界面设计

4.3.1 用户界面应设计得直观、简洁、易于操作，使用户能够快速上手并高效地完成各项任务。

4.3.2 平台应支持多种终端设备的访问，包括 PC 端、移动端等，以满足不同用户的需求和使用场景。

4.3.3 用户界面应采用响应式设计，能够自适应不同屏幕尺寸和分辨率的设备，确保用户在不同设备上都能获得良好的使用体验。

4.4 技术选型

4.4.1 根据项目的实际情况和需求，选择合适的开发语言进行开发。常见的开发语言包括 Java、Python、PHP 等。

4.4.2 选择成熟的开发框架，以提高开发效率和代码质量。例如，对于 Java 项目，可以考虑使用 Spring 框架；对于 Python 项目，可以考虑使用 Django 或 Flask 框架。

4.4.3 根据项目的数据存储需求，选择合适的数据库系统。常见的数据库系统包括 MySQL、Oracle、SQL Server 等。

4.5 接口设计

4.5.1 平台应提供标准化的接口，方便其他系统或平台与之进行集成和数据交换。接口应遵循通用的数据交换标准和协议，如 RESTful API、SOAP 等。

4.5.2 接口设计应充分考虑安全性问题，采用身份验证、授权、加密等安全措施，确保接口不被非法访问和篡改。

4.5.3 平台应提供详细的接口文档，包括接口的功能描述、参数说明、返回值说明等，方便开发人员进行集成和调试。

5 功能要求

5.1 设备报修管理

5.1.1 用户应能够通过平台在线提交设备报修申请，填写设备信息、故障描述、报修人信息等必要信息。

5.1.2 系统应自动或手动对提交的报修申请进行审核，确保信息的真实性和完整性。

5.1.3 用户应能够随时查询报修申请的当前状态，如待审核、已受理、维修中等。

5.1.4 系统应能够自动向用户发送报修进度通知，如受理通知、维修进度更新、维修完成通知等。

5.2 维修派工管理

5.2.1 系统应能够根据维修任务的紧急程度、维修人员的能力、地理位置等因素，自动或手动分配维修任务给合适的维修人员。

5.2.2 系统应支持维修任务的优先级设置和调度，确保重要任务能够得到及时处理。

5.2.3 维修人员应能够查询自己已接收的维修任务列表，包括任务详情、处理进度等。

5.3 维修进度跟踪

5.3.1 系统应实时更新维修任务的进度信息，包括维修开始时间、预计完成时间、实际完成时间等。

5.3.2 用户应能够随时查询维修任务的当前进度和处理状态，以便了解维修进度和预计完成时间。

5.3.3 维修人员应能够向系统反馈维修进度，如已完成某个阶段的维修工作、遇到的困难等。

5.4 维修记录管理

5.4.1 维修人员完成维修任务后，应能够在系统中创建维修记录，包括维修详情、更换备件信息、维修费用等。

5.4.2 用户应能够查询设备的维修历史记录，了解设备的维修情况和维修历史。

5.4.3 系统应支持维修记录的导出功能，方便用户将维修记录以表格、PDF 等格式导出到本地进行保存或打印。

5.5 备件管理

5.5.1 系统应支持备件库存的录入、查询、修改和删除等操作，确保备件信息的准确性和实时性。

5.5.2 当备件库存量低于设定的预警值时，系统应能够自动向管理人员发送预警通知，以便及时补充备件库存。

5.5.3 系统应记录备件的使用情况，包括使用数量、使用时间、使用设备等，以便对备件使用情况进行统计和分析。

5.6 客户反馈管理

5.6.1 系统应支持客户对维修服务进行评价，包括满意度、服务质量、维修效率等方面的评价。

5.6.2 系统应对客户评价进行统计和分析，生成评价报告，以便管理人员了解客户对维修服务的满意度和改进方向。

5.6.3 系统应支持客户在线咨询功能，客户可以通过平台向客服人员咨询维修问题或提出建议。

6 安全要求

6.1 用户安全

6.1.1 平台应要求所有用户进行身份验证，包括管理员、维修人员、用户等。验证过程应使用强密码策略，并定期要求用户更改密码。

6.1.2 对于关键操作或高权限用户，平台应支持多因素认证，如密码、手机验证码、指纹识别等，以提高账户安全性。

6.1.3 平台应实施严格的权限管理策略，确保每个用户只能访问其被授权的数据和功能。权限分配应遵循最小权限原则，即只授予用户完成其工作所需的最小权限。

6.2 数据安全

6.2.1 平台应对敏感数据进行加密存储和传输，包括用户密码、个人信息、维修记录等。加密过程应使用强加密算法，并定期更换密钥。

6.2.2 平台应定期备份数据，以防止数据丢失或损坏。备份数据应存储在安全的位置，并定期进行恢复测试以确保备份的有效性。

6.2.3 平台应实施严格的数据访问控制策略，确保只有被授权的用户才能访问和修改数据。同时，平台应记录所有数据访问和修改操作，以便进行审计和追踪。

6.3 网络安全

6.3.1 平台应部署防火墙和入侵检测系统，以防止非法访问和攻击。防火墙应配置合理的安全策略，以限制对平台的访问。入侵检测系统应实时监测平台的网络流量和异常行为，并及时报警和处理。

6.3.2 平台应定期进行安全漏洞扫描和评估，及时发现并修复潜在的安全漏洞。同时，平台应关注最新的安全漏洞信息，并采取相应的防护措施。

6.3.3 平台应建立安全事件响应机制，制定应急预案和处置流程。在发生安全事件时，平台应迅速响应并采取有效措施，以减少损失和影响。

6.4 软件安全

6.4.1 平台应采用安全的软件开发方法和技术，如输入验证、输出编码、错误处理等，以防止常见的软件安全漏洞。

6.4.2 平台应定期进行代码审计和测试，以确保代码的安全性和可靠性。审计和测试应覆盖平台的所有关键功能和模块。

6.4.3 平台应定期发布软件更新和补丁，以修复已知的安全漏洞和缺陷。同时，平台应提供详细的更

新说明和操作方法，以确保用户能够及时、正确地更新软件。

7 测试方法

7.1 功能测试

7.1.1 根据平台的功能需求，设计覆盖所有功能点的测试用例。确保每个功能点都有相应的测试用例进行验证。

7.1.2 按照测试用例进行逐一测试，验证平台的功能是否按照预期执行。对于发现的问题，及时记录并跟踪解决。

7.1.3 在修复问题后，进行回归测试，确保已修复的问题不再出现，同时验证其他功能是否受到影响。

7.2 性能测试

7.2.1 测试平台在不同负载下的响应时间，确保平台能够在规定的时间内响应用户请求。

7.2.2 模拟多个用户同时访问平台的情况，测试平台的并发处理能力。确保在高并发场景下，平台能够保持稳定运行。

7.2.3 测试平台在运行过程中对系统资源的占用情况，如 CPU、内存、磁盘等。确保平台在运行时不会过度消耗系统资源。

7.3 安全测试

7.3.1 使用专业的漏洞扫描工具对平台进行扫描，发现潜在的安全漏洞。

7.3.2 模拟黑客攻击行为，对平台进行渗透测试，验证平台的安全防护能力。

7.3.3 对平台的代码、配置、日志等进行安全审计，确保平台的安全策略得到有效执行。

7.4 兼容性测试

7.4.1 测试平台在不同浏览器下的表现情况，确保平台在各种浏览器下都能正常运行。

7.4.2 测试平台在不同操作系统下的兼容性，确保平台能够在各种操作系统上正常运行。

7.4.3 测试平台在不同设备上的兼容性，包括 PC、手机、平板等，确保平台能够在各种设备上提供一致的用户体验。

7.5 用户体验测试

7.5.1 测试平台的操作流程是否简单易懂，用户是否能够快速上手并高效完成操作。

7.5.2 测试平台的界面设计是否合理美观，是否符合用户的使用习惯。

7.5.3 测试平台的交互设计是否合理，用户是否能够方便地获取所需信息并进行操作。

7.6 自动化测试

7.6.1 根据测试用例，编写自动化测试脚本，实现测试过程的自动化执行。

7.6.2 将自动化测试脚本集成到持续集成流程中，每次代码提交或变更时都自动执行测试，确保代码质量。

7.6.3 通过自动化监控工具对平台进行实时监控，及时发现并处理异常情况。