

Software Requirement Specification Document - LogIQ

1. Introduction

1.1 Purpose

The Service Management System is a robust, scalable application designed to assist onsite service engineers from companies like Maytag and Whirlpool in troubleshooting and maintaining household appliances such as refrigerators, air conditioners, washing machines, and cooktops. The system is composed of three core components: the Customer App, the Service Engineer App, and the Admin App. Each component is meticulously crafted to address the specific needs of its user base, ensuring a seamless and efficient service process.

1.2 Customer Problem

Customers who own appliances often face challenges when it comes to registering products, reporting issues, and managing service requests. Traditional methods are time-consuming, lack transparency, and often result in frustration due to delays or miscommunication. There is a clear need for a system that simplifies registration, streamlines service requests, and provides customers with real-time updates and a history of their interactions with service engineers.

For service engineers, managing service requests, accessing appliance history, and ordering spare parts can be cumbersome and inefficient, especially when they are on-site and require immediate solutions. The lack of a centralized system that integrates all these functionalities creates bottlenecks and reduces the overall efficiency of service operations.

For administrators, managing the entire service ecosystem—ranging from tracking engineer performance to managing inventory—is a complex task. Current systems are often fragmented, leading to inefficiencies and a lack of visibility into key operational metrics.

1.3 Project Scope

This system aims to provide a seamless experience for customers, service engineers, and organizational administrators by offering features such as appliance registration, service ticket management, engineer matching, real-time troubleshooting assistance, inventory management, and admin controls. The application will be developed and released in three phases, with testing conducted after each phase to ensure performance and stability before moving on to the next development phase.

2. Overall Description

2.1 Product Perspective

The Service Management System is a standalone application that can be integrated with existing customer applications if required. The system consists of three interconnected modules—Customer App, Service Engineer App, and Admin App. Each module is designed to handle specific tasks, ensuring a streamlined workflow for all users.

2.2 Product Feature Snapshot

- **Customer App:** Appliance registration, service ticket management, service history tracking, OTP verification for engineers, feedback submission, and multilingual support.
- **Service Engineer App:** Ticket assignment and management, real-time troubleshooting using a chatbot, inventory ordering, training modules, and multilingual support.
- **Admin App:** Dashboard with analytics, ticket management, inventory management, training content updates, branding management, and region-specific access.

2.3 User Personas and Characteristics

- **Customers:** Users who own appliances and need service or troubleshooting.
- **Service Engineers:** Technicians responsible for on-site repairs and maintenance.
- **Admins:** Company personnel responsible for managing the service engineers, tickets, and application settings - This includes the ticket support staff and the business analysts.

2.4 Design and Implementation Constraints

- The application must be responsive and compatible with multiple devices, including desktops, tablets, and mobile phones.
- Multilingual support must be provided for all user interfaces.
- The application must be secure, implementing 2FA for critical actions such as ticket closure and inventory orders.
- The system should be scalable to handle a large number of users and service requests.

2.5 Assumptions and Dependencies

- The system will depend on reliable internet connectivity for real-time features like ticket assignment, inventory management, and chatbot assistance.
- The availability and accuracy of the LLM models for RAG will directly impact the effectiveness of the chatbot and troubleshooting features.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Customer App

- Appliance Registration:
 - Users must be able to register their appliances by entering model numbers, purchase details, and serial numbers.
 - Upon registration, the warranty should be automatically logged.
- Service Ticket Management:
 - Users can create service tickets by selecting a registered appliance and describing the issue.
 - The app should generate an OTP for the engineer to verify their identity during the service visit.
 - After the service, the user must provide an OTP to the engineer to close the ticket.
 - Users can view service history and generate reports post-service.
- Feedback Mechanism:
 - After a ticket is resolved, users should be prompted to provide feedback on the service quality.

3.1.2 Service Engineer App

- Engineer Matching:
 - Automatically assign tickets to engineers based on their skills, proximity, and availability.
- Ticket Management:
 - Engineers can view, accept, or reject assigned tickets from the dashboard.
 - Options for "Commute to Location" with route optimization and "Resolve Now" for starting the repair process.
- Inventory Management:
 - Engineers can order spare parts, with costs automatically added to the customer's invoice.
 - The app should consider part warranties when adding items to the inventory.
- Chatbot Assistance:
 - The RAG-enabled chatbot should provide troubleshooting tips, assembly/disassembly instructions, and component testing procedures.
- Training and Collaboration:
 - Engineers can access training modules, service manuals, and collaboration tools.
 - An achievement system rewards engineers based on ticket resolution and skill growth.

3.1.3 Admin App

- Dashboard:
 - Admins should see visualizations of ticket trends, appliance failure rates, and engineer performance.
- Ticket Management:
 - Manually assign tickets that were rejected by engineers or not automatically assigned.
- Inventory Management:
 - Admins can manage spare parts inventory, restock items, and update training content.
- Brand Management:
 - Admins can update the logo, color scheme, and other branding elements for both the Customer and Service Engineer Apps.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

- The application should load within 2 seconds under normal conditions.
- The system must handle up to 10,000 concurrent users without significant performance degradation.

3.2.2 Security Requirements

- Implement 2FA for critical actions like ticket closure and inventory orders.
- All data transmissions must be encrypted using industry-standard protocols (e.g., SSL/TLS).

3.2.3 Usability Requirements

- The UI should be intuitive and accessible to users of varying technical proficiency.
- Multilingual support must be available in at least three languages at launch, with plans for additional languages in future updates.

3.2.4 Scalability Requirements

- The application must be scalable to accommodate an expanding user base and increasing data load as more appliances and tickets are registered.

4. Phased Development Plan

4.1 Phase 1: Core Features Implementation

- Develop the Customer App with appliance registration, basic ticket management, and feedback features.
- Develop the Service Engineer App with ticket management, basic dashboard functionality, and chatbot integration.
- Develop the Admin App with basic dashboard analytics and ticket management.
- **Testing:** Conduct a one-month testing period focusing on core functionalities, user experience, and bug fixes.

4.2 Phase 2: Advanced Features and Optimization


- Expand the Customer App to include service history tracking, OTP verification, and multilingual support.
- Enhance the Service Engineer App with inventory management, commute optimization, and training modules.
- Update the Admin App with inventory management and training content updates.
- **Testing:** Conduct a one-month testing period focusing on performance, scalability, and security.

4.3 Phase 3: Final Features and Release Preparation

- Finalize the Customer App with report generation and comprehensive multilingual support.
- Complete the Service Engineer App with the achievement system, enhanced chatbot features, and collaboration tools.
- Finalize the Admin App with branding management and region-specific access control.
- **Testing:** Conduct a one-month testing period focusing on final feature integration, end-to-end testing, and user feedback collection.

5. References

To view the latest version of this document, follow this URL:

 SRS Document - LogIQ

https://docs.google.com/document/d/1JcGhfIrCzKo6ZQkVpCiE-lwAwD6YVYkAJO_zde3-wrA/edit?usp=sharing