Software Requirement Specification (SRS) :

Organization : SmartServe Solutions
Project Title : Web-Based Online Service Management System
Version : 1.0
Date : 16 December 2025

---

1. Introduction and Purpose

- **Technical Translation:** The SRS converts business goals (BRS) and user needs (URS) into clear and simple instructions that developers can easily understand and follow.
- **Development Roadmap:** It acts as a step-by-step guide for the whole project, from planning and designing to testing and final deployment.
- **Stakeholder Alignment:** The document makes sure that developers, testers, and architects all have the same understanding of how the system should work.
- **Architectural Foundation:** It explains what the system will do and gives developers a clear technical structure to build the system correctly.
- **Scope Control:** The SRS clearly defines project limits, so any new feature or change must be checked against it before being added.
- **Verifiable Requirements:** Each requirement is written clearly so it can be tested and confirmed that the system works as expected.
- **Strategic Goal Support:** The SRS helps achieve digital improvement and better efficiency in daily operations.
- **Role-Specific Utility:** It is mainly used by technical people like software developers, architects, and project managers during development.
- **Prioritization Guide:** The document helps the team focus on the most important features first.
- **Project Success Baseline:** It serves as the main reference point to check whether the completed system meets all requirements and is successful.

2. System Overview

- **Project Success Baseline:** It serves as the main reference point to check whether the completed system meets all requirements and is successful.

- **Multi-Tier Architecture:** The system is built in layers, where the user interface (front end) and data processing/storage (back end) are kept separate to improve safety and speed.
- **Cloud-Based Core:** The system runs on a cloud server, so data is always up to date and can be accessed from anywhere at any time.

— **Three Access Points:** The system offers three different interfaces: a Customer Portal, an Admin Dashboard for office staff, and a Mobile App for technicians.
— **Role-Based Security:** The system controls access based on user roles, ensuring each user only sees and uses what they are allowed to.
— **Request Lifecycle Management:** A module manages each service request from creation to completion and final payment.
— **Smart Dispatch Engine:** The system automatically assigns the most suitable technician to each job to save time and reduce travel costs.
— **Automated Notification Engine:** The system sends automatic SMS and email updates to customers and staff without manual effort.
— **Seamless Third-Party Links:** The system connects smoothly with external services like payment gateways, messaging services, and map tools.
— **Near Real-Time Updates:** Any change in service status is updated quickly, so users always see the latest information.
— **Single Source of Truth:** All service data is stored in one central place, removing the need for manual records or multiple files.

3. Functional Requirements (Numbered)

3.1: User Authentication and Authorization

— The system shall allow Customers to register by creating an account using their email address and a password.
— The system shall provide a secure login option for all users (Customer, Admin, Technician) using an email/username and password.
— The system shall allow users to reset or recover their password through a verification link sent to their email.
— The system shall use Role-Based Access Control (RBAC) and define user roles such as Customer, Technician, Dispatcher, Service Manager, and Administrator.
— The system shall restrict access based on roles, ensuring Customers can view only their own data, Technicians can view only their assigned tasks, and Admin users have controlled access based on their permissions.

3.2 : Customer Service Request Management

— The system shall provide an online form that allows Customers to submit a new service request by entering the service type, problem details, preferred date and time, service address, and uploading related files or images.
— The system shall automatically create a unique tracking number for every new request and record the date and time when the request is submitted.
— The system shall display a **“My Requests”** page where Customers can view all their submitted requests along with the current status, request date, and tracking number.

— The system shall allow Customers to open a detailed view of each request, showing submitted details, message history, assigned technician information, and a clear status progress indicator.

3.3 : Administrative Dispatch and Scheduling

— The system shall provide an Admin Dashboard that shows all service requests in real time, with options to filter by status, date, or assigned technician.
— The system shall provide a calendar or schedule view so Dispatchers can easily see technician assignments for each day or week.
— The system shall allow Dispatchers to manually assign a service request to a Technician by selecting from a list or using drag-and-drop.
— The system shall suggest the best technician for a job based on skills, location, and current workload to help Dispatchers make faster decisions.

3.4 : Technician Field Operations

— The system shall provide a mobile-friendly web app or a mobile application for Technicians to use in the field.
— The Technician app shall show a list of assigned jobs, arranged by priority and scheduled time.
— The system shall allow Technicians to quickly update their status (Available, On the Way, At Site, Break) with a single tap.
— For each job, the Technician shall be able to update job status, add notes, upload photos, and record used parts.
— The system shall allow Technicians to capture the customer's digital signature after job completion for confirmation.

3.5 : Real-Time Notification System

— The system shall automatically send email or SMS notifications to Customers for important updates such as request confirmation, technician assignment, technician arrival, and job completion.
— The system shall automatically notify Technicians when a new job is assigned, reassigned, or when a customer sends a message.
— The system shall provide an in-app notification section where users can view all past notifications.
— The system shall allow Administrators to edit and manage notification message templates.

3.6 : Billing and Payment Integration

— When a job is marked as completed, the system shall automatically create a PDF invoice including company details, services, taxes, and total amount.

— The system shall support online payments by integrating with a secure payment gateway such as Stripe.
— The Customer portal shall show all invoices with a **“Pay Now”** option that opens a secure payment page.
— The system shall track payment status such as Pending, Paid, Failed, or Refunded and update the invoice accordingly.
— The Admin Dashboard shall provide a financial overview showing payments, unpaid invoices, and total revenue.

3.7 : Reporting and Analytics

— The system shall generate standard reports such as service request volume, technician performance, customer satisfaction, and revenue.
— The system shall allow reports to be downloaded in PDF and CSV formats.
— The Admin Dashboard shall display key performance indicators like open requests, average completion time, and daily revenue.

3.8 : Data Management and System Configuration

— The system shall provide an admin panel to manage service categories, technician details, skills, pricing rules, and inventory items.
— The system shall keep a complete activity log of all important data changes such as add, update, and delete actions.
— The system shall automatically back up the database every day and store it in a secure external location.

4. Non-Functional Requirements

Performance

- **Response Time:**The system should load most web pages (about 95%) within **2 seconds** under normal use. Important actions like submitting a service request or updating job status should complete within **1 second**.
- **Throughput:** The system should be able to handle **at least 50 users at the same time** and process **100 new service requests per hour** without slowing down.
- **Scalability:**The system should be designed in a way that it can grow easily and handle **three times more users and data** over the next **3 years**.

Reliability & Availability

- **Availability:** The system should be available and working **99.5% of the time** during business hours (7:00 AM to 8:00 PM, Monday to Saturday).
- **System Stability:**Important parts of the system should run for a long time without failure, ideally **at least 720 hours** between major issues.
- **Recovery:**If something goes wrong, the system should be able to restore data from backups within **4 hours**, and data loss should not be more than **24 hours old**.

Security

- **Data Protection:**Sensitive data like passwords and payment details should be kept safe using strong encryption when being sent and when stored.
- **Authentication:**User passwords should be stored securely using strong password protection methods (such as hashing).
- **Authorization:**Users should only be able to access what their role allows, and no user should gain higher access by mistake.
- **Vulnerability Protection:**The system should be built using standard security rules to protect against common attacks like SQL Injection and Cross-Site Scripting (XSS).

Usability

- The system should be easy to use and score **at least 75** in user experience testing.
- The Customer and Admin websites should work smoothly on **all screen sizes**, from mobile phones to large desktop screens.
- The Technician app should be simple enough that a technician can update a job status in **2 taps or fewer**, with very little training.

Maintainability & Supportability

- The system code should be **well-organized**, easy to understand, and include comments for complex parts.
- The system should provide admin tools to **monitor performance**, check errors, and manage active user sessions easily.

5. System Constraints

- **Technological Constraints:**The system must be built using commonly used technology stacks like **LAMP** (Linux, Apache, MySQL, PHP) or **MEAN** (MongoDB, Express.js, Angular, Node.js), based on the team's skills.
- **Integration Constraints:**Online payments must follow **payment security rules**. SMS and email services will depend on external providers such as messaging and email APIs.
- **Mobile Platform Constraints:**The Technician app must work on **iOS 13 or higher** and **Android 9 or higher** devices.
- **Budget Constraints:**The cost of development and third-party services must stay **within the approved project budget**.
- **Time Constraints:**The first working version of the system (MVP) must be ready and live within **6 months**.
- **Legal Constraints:**The system must follow **data privacy and consumer protection laws**, including safe handling of user data.

6. Assumptions and Dependencies

Assumptions

- It is assumed that Customers and Technicians have **internet access** and a suitable device such as a phone or computer.
- It is assumed that the company will **fully use this system** and gradually stop manual or paper-based processes.
- It is assumed that proper **training** will be given to Admins and Technicians before the system goes live.
- It is assumed that **pricing rules and service types** will remain mostly stable and not change every day.
- It is assumed that **existing customer data** will be moved into the new system if required.

Dependencies

- The project depends on successful setup and access to **payment, SMS, and email service providers**.
- Live tracking features depend on Technicians **keeping location services enabled** on their mobile devices.
- System speed and uptime depend on the **cloud hosting provider** chosen for deployment.
- The development schedule depends on **timely feedback and support** from company experts for clarification and approvals.
- Offline features in the Technician app depend on the **mobile framework's ability** to store data locally and sync it later.