Software Requirements Specifications

Version 1.0

April 7, 2023

Aryan Sagavekar

Shravani Chidrewar

Table Of Contents

* Table of contents
* List of figures
* Introduction
  + Purpose
  + Scope of Project
  + Glossary
  + References
  + Overview of Document
* Overall Description
  + System Environment
  + Functional Requirements specification
  + User Characteristics
  + Non-Functional Requirements
* Requirements Specification
  + External Interface Requirements
  + Functional Requirements
  + Detailed Non-Functional Requirements

**INTRODUCTION:**

**Purpose**

The purpose of this document is to define the requirements and specifications for the development of a Local Services Management System. It is a system that will be used by customers as well as service providers according to their daily needs.

**Scope**

The scope of the project includes the following functionalities defined in short[Note that there will be more functionalities than the once mentioned below]:

* Customer Management: The software should allow customers to add, modify or delete their accounts as per their needs.
* Service Management: It should allow local service providers to manage their service offering such as service type, category, pricing and service availability.
* Service Scheduling: It should allow customers to schedule when they want their service after negotiating with the service providers.
* Payment Management: It should be able to manage transactions, invoices, billings and payment processing.

**Glossary**

The following definitions, acronyms and abbreviations are used in the document:

* LSMS: Local Services Management System.
* UI: User interface.
* API: Application Programming Interface.

**References**

The following documents are referenced in this SRS:

* Web Publishing system by Paul Adams, Bobbie Baker, Charles Charlie: For formatting of this document.
* Industry Standards for software development technologies.

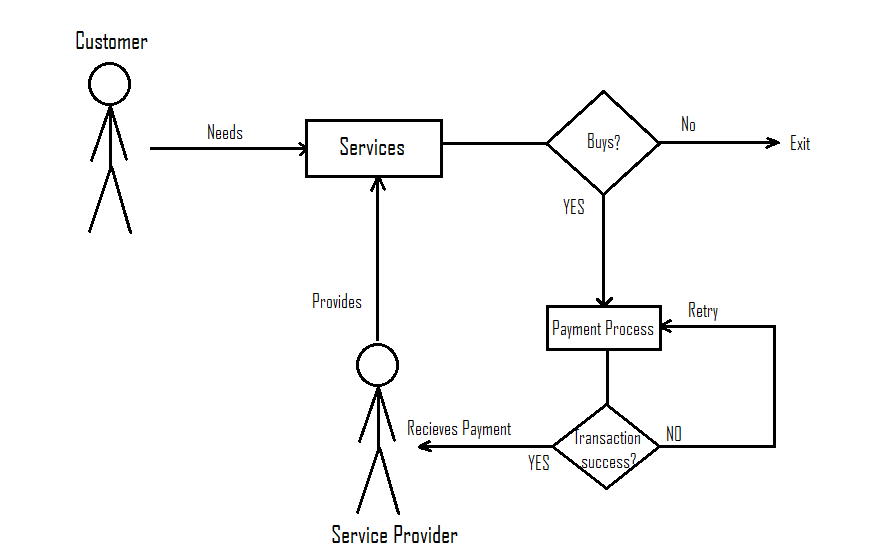
**Overview**

This SRS document is organized into four sections: Introduction, General Description, Specific requirements and Appendices. The General Description section provides high-level overview of how LSMS is going to be built and used. Specific requirements section outlines the function and non-functional requirements for the system detail. Lastly, the appendices section includes supplementary materials that are relevant to the SRS document, such as software development and analysis models.

**GENERAL DESCRIPTION:**

**System Environment**

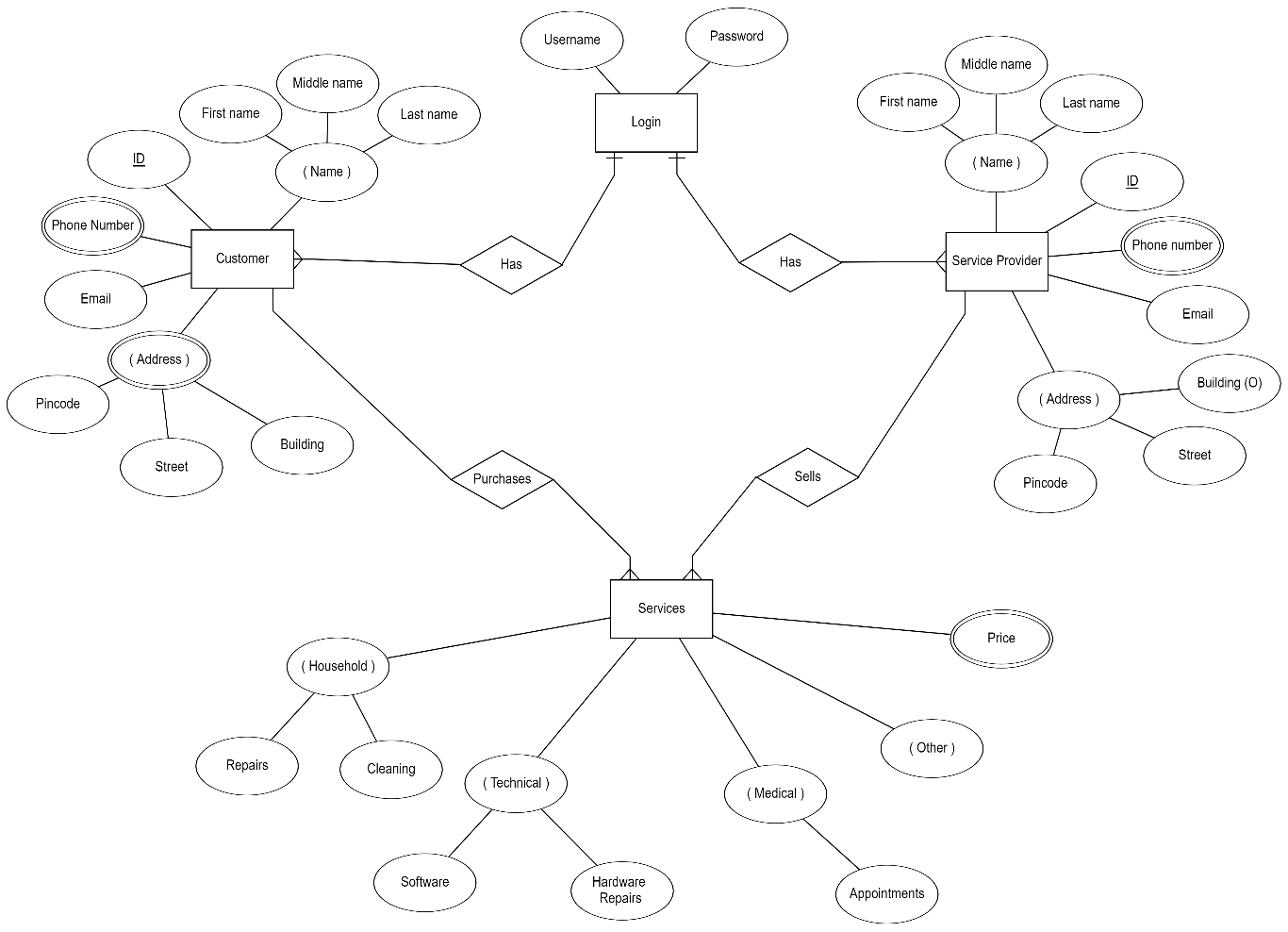
The following diagram represents a very basic structure of the project.



**System Environment.**

The Local services management system has mainly two active users, customers/naïve users and service providers. Customers can buy any service that is available on the software and service providers can manipulate services accordingly and upload them here.

**Product Functions**

The major functions of Local services management system are shown in the ER diagram below:

**User Class and Characteristics**

Following types of users will use LSMS software:

* Customers: People who are in need of services can login their account. They can search for available services in a user-friendly search engine. They can further modify or delete their account as per their needs. The following are functionalities of a customer:
  + Search for services in a search engine.
  + Check availability of services.
  + Schedule services according to needs.
  + Change/remove their accounts accordingly.
  + Pay for a service in advance or after the work is done.
* Service Providers: People who are willing to provide services through this software. They can update, add or remove their services as per their needs.
  + Add/remove/update services in the software.
  + Negotiate with customers to schedule services.
  + Set prices for different services accordingly.
  + Change/remove their accounts accordingly.
* Admin: Management team, who will have access to all functionalities and whose job is to prevent malicious activities using different tools.
  + Warn accounts if found in malicious practices.
  + Remove any account if warning exceed count.
  + Maintain the software so that its usable for all kinds of users.

**Design and Implementation Constraints**

LSMS must adhere to the following constraints:

* Compatibility: The software should be compatible with all kinds of devices and industry standard web browsers too.
* Security: It should be designed with security in mind, and it must comply with industry standard security protocols to ensure privacy and safety of user data.
* Usability: The software should be intuitive and user friendly to use, with clear choice and concise instructions provided to guide users through the system.
* Performance: The software should be built keeping in mind that it should be able to handle large amount of data and traffic without degrading the performance by any means.
* Implementation of database using a centralised database management system.

**Assumptions and Dependencies**

The development of LSMS software is based upon the following assumptions:

* Request for available services exists nearest to the destination of the customer.
* The project will be developed in a reasonable timeframe, with a regular progressive app update [Newer versions with additional features].
* It will be developed using industry-standard web development technologies, such as c++, java, SQL database etc.

**SPECIFIC REQUIREMENTS:**

**Functional Requirements**

This section provides a detailed view on all the important functional requirements.

* Registration- First step is to register. There will be two modes of registration
  + Service Providers: These have to provide details about themselves such as name, address, valid documents, phone number etc.
  + Regular users/Customers: They have to provide details about their address, phone number, payment information etc.
* Login and sign up-
  + Input details mentioned in registration.
  + After processing the information, if there is no error a unique id will be allocated to the user. They can set their username and password to complete the sign-up procedure.
  + They can now login through their username and password to use the features of the app.
* Managing services by service providers-
  + Service providers can upload services according to them and provide details such as, time/availability and price.
  + Add category to their services, which can be automated later.
* Search-
  + A user-friendly search engine where users can search for services by typing a specific service or by using a filter system. List of services related to the keyword mentioned are shown.
* Booking-
  + Customers can book services as per the availability given by the service providers. They can cancel the service in a short period if they booked it by mistake.
  + Confirmation using OTP or failure message in case wrong information was displayed or service is not available.
* Payments management-
  + The users detailed will be stored securely so the user is comfortable to make transactions.
  + There will be history option for users to check their transactions and keep track of them.

**Non-Functional Requirements**

The non-functional requirements are mentioned in detail below:

* Usability: The user interface of the software will be user friendly to use and step by step instructions will be provided to the user about how the software works.
* Performance: It should be able to handle large amounts of data and traffic without degrading the performance.
* Security: It should be designed with security in mind, and it must comply with industry standard security protocols to ensure privacy and safety of user data.
* Scalability: The project should be built keeping scalability in mind, that is, it should be able to grow/advance and adapt according to feedbacks and other requirements.

**Logical Database Requirements**

The software should be able to keep a check on the history of user’s past services so that the next time user opens the app it could suggest the user with related services based on the past services.

**Software Languages Used**

The languages used in this project will be Java [supporting the UI/working as backend], XML (Extensible markup language )[For frontend part], C++ native/C make [help speeding up the software building process by using already built libraries].

**Development Tools**

The software building procedure will make use of different tools to complete the project. Some examples of tools are android studio, APIs, Java development kit and different c++ libraries. It will also need a json parserer further when map will be added to the project.

**Software System Attributes**

Software attributes such as performance, reliability, portability etc are important things when building a project and can have a significant effect on the architecture of the project. Some of the attributes are mentioned below:

* Reliability: To solve issues related to the software so that failures in the system could be minimized and users can get a good experience.
* Security: to ensure that all data related to user is stored safely and encrypted so that there is no possibility of scams.
* Maintenance: The maintenance for this kind of software lies only in problem solving as the database is handled pretty much automatically. Making sure the app gets newer versions time to time is necessary.
* Portability: Although the basic build of the software will be related to one operating system. The further upgrades should increase its portability, that is, make it work on different operating systems.

**APPENDICES:**

**Software development model**

There are several software models, but for this project we have chosen incremental model. The incremental model involves building a base project and implementing it and then adding additional features starting again from the resource gathering process in the next iteration. The following diagram gives an idea about how the incremental model works:

+---------------------+

| Requirements |

+---------------------+

|

|

+---------+-----------+

| Design |

+---------+-----------+

|

|

+----------------+----------------+

| Implementation |

+----------------+----------------+

|

|

+----------+----------+

| Testing |

+----------+----------+

|

|

+--------+---------+

| Deployment |

+------------------+

(in a iteration)

**TESTING PHASE**

**Introduction**

Software testing involves the execution of a software component to evaluate one or more properties of interest. Testing is the process of exercising software with the intent of finding errors and ultimately solving them.

Software testing can provide new task apart from the primary tasks of building the app and makes the developing team clear of the risks its implementation can have.

It also provides stakeholders with information about the quality of the software product or service under test. Testing is a set of activities that should be planned in advance and conducted systematically.

**Types of Testing**

There are many types of testing, each for a particular part of the project. They are as follows:

* Unit Testing: these tests are used to test the individual components of the project, written by programmers themselves, to know that each part is functioning properly.
* Integration Testing: these tests are used to test the interactions between different components of the software. This ensures that the software as whole is working properly.
* System Testing: Used to test the software as a whole, including its interactions with external hardware components. They ensure that the software meets the specified requirements and is functioning well in a real time environment.
* Regression Testing: One of the important test cases in this project because it uses the incremental model and upgrades are significant. This test case ensures that there are no unintended effects on the software when it is modified or updated, that is, software continues to run properly even after updates.
* Performance Testing: used to test the performance of the software, such as its response time, scalability and resource usage. They are used to ensure that the software meets the performance requirements specified in the SRS.