Software Engineering

Software Requirements Specification

(SRS) Document

**{Product Name}**

**Author**:

Nataliia Chausova,

<https://www.linkedin.com/in/nataliiachausova/>

<https://www.thebagirl.com/>

# Revision History

{In the revision history section you would post date of revision, what has changed, and who changed the document (authors of the changes).}

| Version # | Date | Summary of the Changes | Author(s) | Reviewer(s) |
| --- | --- | --- | --- | --- |
| <nn.nn.nn> | <mm/dd/yyyy> | <Describe Changes Made> | <Author Name> | <Reviewer Name> |

# Approval History

{This is your sign-off history, documenting who approved each version of the document.}

| Version # | Date | Approver(s) |
| --- | --- | --- |
| {nn.nn.nn} | {mm/dd/yyyy} | {Approver Name} |

# Introduction

{This section provides a comprehensive overview of the software project, including its purpose, scope, and objectives. The introductory section discusses the business objectives and how the software project aligns with those objectives. It sets the context for the technical requirements.}

## Purpose

{Explain the fundamental reason for the software project, its goals, and how it will address a specific need or problem.}

*Example: "The software project aims to develop a comprehensive customer relationship management system to improve customer service, streamline sales processes, and enhance data analysis."*

## Scope

{Define the boundaries of the project, what is included, and what is not included. This helps in managing expectations and preventing scope creep.}

*Example: "The project will cover all customer-related interactions and data management within the company, including but not limited to lead management, sales order processing, and customer support."*

## Objectives

{List the specific goals and outcomes that the software project aims to achieve. These objectives should be measurable and aligned with the purpose.}

*Example: "1. Increase customer satisfaction by 20% within the first year. 2. Reduce response time to customer inquiries by 30%. 3. Improve sales conversion rates by 15%."*

# Assumptions

{The SRS document may have a dedicated section or subsection where assumptions are explicitly listed. This section is used to make stakeholders aware of any conditions or expectations that are taken for granted during the software development process.}

*Example:*

1. *It is assumed that key stakeholders will be available for project meetings and reviews on a bi-weekly basis throughout the development process.*
2. *The project assumes that the legal department will provide timely approvals for necessary contracts and agreements as they arise during the project.*
3. *The project assumes that the marketing team will have promotional materials ready for the software's launch according to the agreed-upon schedule.*
4. *It is assumed that user training sessions will be conducted within the first week after software deployment to ensure a smooth transition.*
5. *The project assumes that the required hardware and infrastructure for user testing will be provided as specified in the project plan.*
6. *The project assumes that the third-party payment gateway service will maintain 99.9% uptime during the software's operation.*

# Constraints

{Business constraints, such as budget limitations or regulatory requirements, may be documented in the SRS to ensure that the software project stays within the defined constraints.}

*Example:*

1. *The project is constrained by a fixed budget of $500,000, and no additional funds are available for expansion or scope changes.*
2. *There is a strict regulatory requirement that the software must comply with GDPR, and failure to do so could result in significant legal penalties.*
3. *The project is subject to a predefined launch date due to a marketing campaign that has already been scheduled, and any delays may result in missed opportunities.*
4. *The project is constrained by a limited pool of available skilled resources, which may affect the speed of development and testing.*
5. *The software must run on the company's existing hardware infrastructure, and there is no budget allocated for hardware upgrades or replacements.*

# Functional Requirements

{Here, the document outlines the specific features and functionalities the software must deliver.

A functional requirement is a specific and detailed description of the **behavior and functionality** that a software system, product, or component is expected to exhibit. Functional requirements define what the software should do and how it should respond to various inputs or stimuli. These requirements typically focus on the system's features, capabilities, and interactions with users and other software systems.

Some business requirements may translate into functional requirements in the SRS. For example, if a business requires something, this can be translated into specific functional requirements.The elaboration on each requirement depends on the time, methodology, resources.}

# Non-Functional Requirements:

{These encompass constraints, performance expectations, and qualities like security, scalability, and usability.}

*Examples:*

***Performance Requirements****:*

*"The system shall respond to user interactions within 1 second for 95% of all requests."*

*"The software must support up to 1,000 concurrent users during peak usage periods."*

*"The database queries should execute in less than 300 milliseconds for 95% of requests."*

*"The page loading time should be less than 4 seconds for 95% of times for 10 Mbit and more internet connection."*

***Security Requirements:***

*"User authentication and authorization must follow industry-standard security protocols, such as OAuth 2.0."*

*"Sensitive customer data must be encrypted using AES-256 encryption algorithm both at rest and in transit."*

***Reliability Requirements:***

*"The software must achieve a system uptime of 98% during regular business hours."*

*"Backup and disaster recovery procedures should be in place, and data backups must be performed daily and stored offsite."*

*"The system should be able to recover from unexpected failures within 15 minutes”*

***Scalability Requirements:***

*"The system should support an increase in data storage by at least 50% annually till 2025 year."*

*"The software architecture must be designed to seamlessly scale horizontally by adding new servers to accommodate increased load."*

***Compatibility Requirements:***

*"The application shall be compatible with the latest versions of the following web browsers: Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge. 2 latest major versions for each browser at the moment of the release."*

*"The software must be compatible with mobile devices running Android 7.0 and later, as well as iOS 13 and later, that are available (not beta) by the moment of the release."*

***Maintainability Requirements:***

*"Documentation must be provided for all components and APIs, including code comments and user guides."*

## Process Models

{Explains the processes in the system, usually flowcharts that cover main scenarios.}

## Data Models

{Explains how data will be structured and manipulated within the software.}

# System Architecture

{Describes the overall structure and components of the system, including hardware, software, and data flow. Usually assisted by charts}

*Example:*

*"The system architecture will consist of a web server running on a Linux-based OS, a backend API built with Node.js, and a database using MySQL. Data flow will follow a RESTful architecture."*

# Dependencies

{Lists any external systems, libraries, or components the software relies on.}

*Example:*

*"The software depends on a third-party payment gateway service for processing transactions and a geolocation API for location-based features."*

# Testing and Validation

- {Outlines the approach to testing and validation, including acceptance criteria.}

- Example:

- "Acceptance Criteria: All functional requirements will be tested through unit testing, integration testing, and user acceptance testing. Acceptance criteria will be met when all test cases pass without critical defects."

# Glossary

{A list of terms and definitions used in the document to ensure consistent understanding, shall include both technical, industry-specific terms and project-specific terms.}

*Example:*

*API (Application Programming Interface): A set of rules and protocols that allows different software applications to communicate with each other.*

*UX (User Experience): The overall experience that a user has while interacting with a product, particularly the interface and usability aspects.*

*CRUD (Create, Read, Update, Delete): Acronym for the four basic database operations - Create (adding new data), Read (retrieving data), Update (modifying data), and Delete (removing data).*

*OAuth 2.0: An industry-standard authorization framework that enables secure third-party access to user data without exposing user credentials.*

*Intrusion Detection and Prevention System (IDPS): A security system designed to detect and prevent unauthorized access, misuse, or anomalies in a network or system.*

*Horizontal Scaling: Increasing a system's capacity by adding more servers or nodes in a distributed environment to handle increased load.*