Solution Design Document Template

Project ABC for Client XYZ

Version: 1.00

Updated: 01/01/2000

Author: John Smith

*IMPORTANT!*

**Note 1:** While a lot of effort has been put into this document to make it as generic as possible, you will inevitably find yourself adding additional sections or removing irrelevant ones, and that is perfectly fine.

**Note 2:** Make sure to read [the article](https://softwaredominos.com/home/software-design-development-articles/write-solution-design-document/) on documenting a solution design before attempting to use this document. Reading and understanding the article contents will allow you to achieve the intended purpose from documenting a good solution design.

**Note 3:** Make sure to use your company’s template if you have one.

# Overview or Introduction

* It can be easily read and understood by anyone.
* Allows the reader to determine whether this document is helpful/accessible for them.

# Summary of Existing Functionality

* Provides some context and background while linking new requirements to existing ones through external references.

# Requirement Details

* A breakdown of the new requirements with comments and discussions.
* It helps determine which requirements are already met, excluded, or require further clarification.

# Assumptions and Prerequisites

* This is an integral part of the design process. Its purpose is to ensure that everybody is clear on the foundations of the design.
* It will also allow an early assessment of whether these assumptions are valid, acceptable, or require a review.

# [High-Level Design](https://softwaredominos.com/home/software-design-development-articles/high-level-solution-design-documents-what-is-it-and-when-do-you-need-one/)

* This section aims to provide process flow updates and information pathway changes using diagrams, tables, and other visualizations.
* Typically, also addresses integration issues with other systems.

# Low-Level Design

* This can be achieved via flowcharts, pseudo-code, and flowchart diagrams.
* The aim is to pinpoint expected source code files, data model tables, and functions to be modified.
* It will also give testers an idea of the scope of change and help drive the overall [testing effort](https://softwaredominos.com/home/software-design-development-articles/software-testing-and-quality-assurance-a-modern-analysis-of-its-internal-dynamics-and-impact-on-delivery/).
* This section equally covers exception handling and negative scenarios.

# Impact Analysis

* Discusses the impact on existing versions of the product already in production. It also discusses any possible degradation or interruption of service after the upgrade.
* It can take the form of a [bidirectional traceability matrix](http://www.westfallteam.com/Papers/Bidirectional_Requirements_Traceability.pdf). The latter is extremely useful in linking business requirements to pieces of code and specific test cases and user documentation.

# Out-of-scope

* Essential for a mutual understanding of what will not be covered in the current project.
* It helps avoid misunderstandings and potential "free work".

# Risks and Mitigation

Risks can arise in the form of:

* External dependencies on the availability of third-party components in the form of software modules, specifications, feedback, resource availability, etc.
* Regression issues are introduced due to work on [legacy, poorly documented, or badly designed systems](https://softwaredominos.com/home/software-design-development-articles/writing-quality-clean-code/).
* Incomplete requirements require decisions to be deferred until after the project has started.
* Any missing or unavailable prerequisites for coders, testers, or operations such as tools, test environments, etc.

# Appendices

* Additional information that could be too detailed or not strictly relevant to the topics discussed
* It can be presented for completeness and clarity.