<Project Name>

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Testability 4

2.5 Usability 4

3. Design Constraints 4

# Introduction

[The introduction of the **Supplementary Specification** provides an overview of the entire document.

The **Supplementary Specification** captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:

Legal and regulatory requirements, including application standards.

Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.

Other requirements such as operating systems and environments, compatibility requirements, and design constraints.]

The Supplementary Specification is a document that provides a comprehensive overview of the system requirements that are not captured in the use cases of the use-case model. It includes legal and regulatory requirements, quality attributes of the system, and other essential requirements such as compatibility requirements, operating systems, and design constraints. The purpose of this document is to capture all the necessary system requirements to ensure the successful development and deployment of the system. The Supplementary Specification defines the quality attributes of the system, including usability, reliability, performance, and supportability requirements, and identifies any regulatory or compliance standards that must be met. By capturing these requirements, the development team can create a roadmap for the design, implementation, and testing of the system.

In this document we will be able to see the requirements that will be needed for the project to work.

# Non-functional Requirements

*[Define system quality attributes in terms of scenarios according to the following template:*

* *Quality attribute definition*
* *Source of stimulus: the entity (human or another system) that generated the stimulus or event*
* *Stimulus: a condition that determines a reaction of the system*
* *Environment: the current condition of the system when the stimulus arrives*
* *Artifact: is a component that reacts to the stimulus. It may be the whole system or some pieces of it*
* *Response: the activity determined by the arrival of the stimulus*
* *Response measure: the quantifiable indication of the response*
* *Tactics*

*]*

## Availability

Quality Attribute Definition: The ability of the system to remain accessible and operational to users, even in the event of system failures or errors.

Source of Stimulus: System users and other systems.

Stimulus: System failures or errors, including hardware failures, software crashes, and network interruptions.

Environment: Normal operation and during system failures or errors.

Artifact: The entire system or subsets of the system, including servers, databases, and networks.

Response: The system must be able to recover from system failures or errors and maintain access for users.

Response Measure: The system must be available 99.9% of the time.

Tactics: Use of redundant hardware and software components, load balancing, and failover mechanisms.

## Performance

Quality Attribute Definition: The ability of the system to process user requests and return results in a timely and efficient manner.

Source of Stimulus: System users and other systems.

Stimulus: User requests for data, system updates, and other operations.

Environment: Normal operation and peak usage times.

Artifact: The entire system or subsets of the system, including servers, databases, and networks.

Response: The system must process user requests and return results within an acceptable timeframe.

Response Measure: The system must respond to user requests within 2 seconds.

Tactics: Caching, indexing, optimizing queries, and load balancing.

## Security

Quality Attribute Definition: The ability of the system to protect user data and prevent unauthorized access or use.

Source of Stimulus: Hackers and unauthorized users.

Stimulus: Attempts to gain access to the system, steal user data, or cause damage to the system.

Environment: Normal operation and during security breaches or attacks.

Artifact: The entire system or subsets of the system, including servers, databases, and networks.

Response: The system must prevent unauthorized access, detect and respond to security breaches, and protect user data.

Response Measure: The system must meet industry-standard security practices and comply with relevant regulations.

Tactics: Encryption, secure authentication, access control, monitoring and logging.

## Testability

Quality Attribute Definition: The ability of the system to be easily tested and validated for correctness.

Source of Stimulus: Quality assurance teams and system developers.

Stimulus: Changes to the system, new features, and bug fixes.

Environment: Testing environments, development environments.

Artifact: The entire system or subsets of the system, including servers, databases, and networks.

Response: The system must be designed in a way that facilitates testing and validation of new features and changes.

Response Measure: Test coverage of at least 90%.

Tactics: Unit testing, integration testing, automated testing.

## Usability

Quality Attribute Definition: The ability of the system to be easy to use and understand for users.

Source of Stimulus: System users.

Stimulus: Attempts to use the system for various tasks and operations.

Environment: Normal usage of the system.

Artifact: The entire system or subsets of the system, including user interfaces and documentation.

Response: The system must be designed in a way that makes it easy for users to understand and use.

Response Measure: The system must receive a usability score of at least 75% in user testing.

Tactics: User-centered design, user testing, and feedback mechanisms.

# Design Constraints

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

The system must be built using the Angular framework.

The backend must be built using Node.js and MongoDB.

The system must adhere to the RESTful API design.

The system must use JWT for user authentication.

The system must be hosted on Amazon Web Services.

The design must be responsive and optimized for mobile devices.

The system must use industry-standard encryption for data security.

The system must comply with relevant data privacy laws and regulations.

The system must be designed to handle a high volume of concurrent users.