**[**GYMIFY**] Requirements Specification**

**Version** 0**.**2

April 7, 2023

Use this Requirements Specification template to document the requirements for your product or service, including priority and approval. Tailor the specification to suit your project, organizing the applicable sections in a way that works best, and use the checklist to record the decisions about what is applicable and what isn't.

The format of the requirements depends on what works best for your project.

This document contains instructions and examples which are for the benefit of the person writing the document and should be removed before the document is finalized.

To regenerate the TOC, select all (CTL-A) and press F9.

**Table of Contents**

[**[GYMIFY] REQUIREMENTS SPECIFICATION 1**](#_19c6y18)

[**VERSION 0.2 1**](#_3tbugp1)

[**APRIL 7, 2023 1**](#_28h4qwu)

[**1.**](#_nmf14n) **EXECUTIVE SUMMARY 3**

[1.1](#_37m2jsg) Project Overview 3

[1.2](#_1mrcu09) Purpose and Scope of this Specification 3

[**2.**](#_46r0co2) **PRODUCT/SERVICE DESCRIPTION 3**

[2.1](#_2lwamvv) Product Context 3

[2.2](#_111kx3o) User Characteristics 3

[2.3](#_3l18frh) Assumptions 3

[2.4](#_206ipza) Constraints 3

[2.5](#_4k668n3) Dependencies 4

[**3.**](#_2zbgiuw) **REQUIREMENTS 4**

[3.1](#_1egqt2p) Functional Requirements 5

[3.2](#_3ygebqi) Non-Functional Requirements 5

[*3.2.1*](#_2dlolyb) *Product Requirements 5*

[**3.2.1.1**](#_44sinio) **User Interface Requirements** 6

[**3.2.1.2**](#_2jxsxqh) **Usability** 6

[**3.2.1.3**](#_z337ya) **Efficiency** 6

[3.2.1.3.1](#_3j2qqm3) Performance Requirements 6

[3.2.1.3.2](#_1y810tw) Space Requirements 6

[**3.2.1.4**](#_4i7ojhp) **Dependability** 6

[**3.2.1.5**](#_2xcytpi) **Security** 7

[*3.2.2*](#_sqyw64) *Organizational Requirements 7*

[**3.2.2.1**](#_3whwml4) **Environmental Requirements** 7

[**3.2.2.2**](#_2bn6wsx) **Operational Requirements** 7

[**3.2.2.3**](#_qsh70q) **Development Requirements** 7

[*3.2.3*](#_3cqmetx) *External Requirements 7*

[**3.2.3.1**](#_1pxezwc) **Regulatory Requirements** 7

[**3.2.3.2**](#_49x2ik5) **Ethical Requirements** 7

[**3.2.3.3**](#_2p2csry) **Legislative Requirements** 7

[3.2.3.3.1](#_147n2zr) Accounting Requirements 7

[3.2.3.3.2](#_3o7alnk) Security Requirements 7

[3.3](#_1rvwp1q) Domain Requirements 7

[**4.**](#_4bvk7pj) **USER SCENARIOS/USE CASES 7**

# Executive Summary

## ***Project Overview***

The purpose of this project is to create an application for a gym chain in order for them to add usability to their user experience. The intended audience is a big enough chain of gyms and the people that respectively go there.

## ***Purpose and Scope of this Specification***

The specification document is there to define the functionality and requirements of the project, its scope and the people it will serve. It is a physical record of the use of our application. and it is a description of the way our app is intended to be used. Also it is there to make sure that our application upholds the laws and ethical norms it falls into.

The document is created in a way to guide us through creating an application without missing any of the requirements while also staying within the scope, and not breaking any guidelines. The document is created in such a way that even someone that hasn’t been part of the document's creation can contribute to its execution and/or understand how the application will come to be. Its intended audience is everyone interested in going to a gym and it is there to provide convenience in doing so.

(appendix 1.0)(this will be a hyperlink that takes us to the guidelines)

**In scope**

This document addresses requirements related to phase 2 of Project A:

* modification of Classification Processing to meet legislative mandate ABC.
* modification of Labor Relations Processing to meet legislative mandate ABC.

**Out of Scope**

The following items in phase 3 of Project A are out of scope:

* modification of Classification Processing to meet legislative mandate XYZ.
* modification of Labor Relations Processing to meet legislative mandate XYZ.

(Phase 3 will be considered in the development of the requirements for Phase 2, but the Phase 3 requirements will be documented separately.)

# Product/Service Description

Our app will provide convenience as a way of service. The main factors affecting our app are the user base, technology, and resources. We are limited by the implementation process and lack of manpower.

## ***Product Context***

Our product relates to similar applications in other areas, but by solely focusing on the gyms we are unrelated to any current products to our knowledge.

It is independent and self contained without any dependencies on other software.

## ***User Characteristics***

* Member
* Coach/Trainer/Gym Expert
* Manager

## ***Assumptions***

* Browser
* Storage

We will assume that the user will have storage and a browser installed on their computer.

## ***Constraints***

* usability between old and new versions of software
* Server Resources ( depending on how much processing power we have at hand)
* User Base ( slow connection due to too many users)
* Hardware restrictions based on old computers
* Storage ( very little but it may be a constraint)
* Php ( will prove to be difficult to implement certain features)

## ***Dependencies***

List dependencies that affect the requirements. Examples:

* Our software will depend on current web programming languages, and its features
* It will depend on the browser and device / os that will be using it

# Requirements

* Describe all system requirements in enough detail for designers to design a system satisfying the requirements and testers to verify that the system satisfies requirements.
* Organize these requirements in a way that works best for your project. See Appendix DAppendix D, Organizing the Requirements for different ways to organize these requirements.
* Describe every input into the system, every output from the system, and every function performed by the system in response to an input or in support of an output. (Specify what functions are to be performed on what data to produce what results at what location for whom.)
* Each requirement should be numbered (or uniquely identifiable) and prioritized.

See the sample requirements in Functional Requirements, and System Interface/Integration, as well as these example priority definitions:

**Priority Definitions**

The following definitions are intended as a guideline to prioritize requirements.

* Priority 1 – The requirement is a “must have” as outlined by policy/law
* Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits
* Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS **must be** either A or I" or "It **would be nice** if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

* A good requirement is:
* Correct
* Unambiguous (all statements have exactly one interpretation)
* Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)
* Consistent
* Ranked for importance and/or stability
* Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms and specify measurable quantities)
* Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)
* Does not specify any particular design
* Traceable (cross-reference with source documents and spawned documents).

## ***Functional Requirements***

In the example below, the requirement numbering has a scheme - BR\_LR\_0## (BR for Business Requirement, LR for Labor Relations). For small projects simply BR-## would suffice. Keep in mind that if no prefix is used, the traceability matrix may be difficult to create (e.g., no differentiation between '02' as a business requirement vs. a test case)

The following table is an example format for requirements. Choose whatever format works best for your project.

For Example:

| **Req#** | **Requirement** | **Comments** | **Priority** | **Date Rvwd** | **SME Reviewed / Approved** |
| --- | --- | --- | --- | --- | --- |
| BR\_LR\_05 | The system should associate a supervisor indicator with each job class. | Business Process = “Maintenance | 3 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_08 | The system should handle any number of fees (existing and new) associated with unions. | Business Process = “Changing Dues in the System”  An example of a new fee is an initiation fee. | 2 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_10 | The system should capture and maintain job class status (i.e., active or inactive) | Business Process = “Maintenance”  Some job classes are old and are no longer used. However, they still need to be maintained for legal, contract and historical purposes. | 2 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_16 | The system should assign the Supervisor Code based on the value in the Job Class table and additional criteria as specified by the clients. | April 2005 – New requirement. It is one of three new requirements from BR\_LR\_03. | 2 |  |  |
| BR\_LR\_18 | The system should provide the Labor Relations office with the ability to override the system-derived Bargaining Unit code and the Union Code for to-be-determined employee types, including hourly appointments. | April 2005 – New requirement. It is one of three new requirements from BR\_LR\_04.  5/11/2005 – Priority changed from 2 to 3. | ~~2~~  3 |  |  |

## ***Non-Functional Requirements***

**In here try to use the Structure given at slide 13 in Requirements Engineering Lecture Slides, with main categories of:**

### **Product Requirements**

* + Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.

#### **User Interface Requirements**

In addition to functions required, describe the characteristics of each interface between the product and its users (e.g., required screen formats/organization, report layouts, menu structures, error and other messages, or function keys).

#### **Usability**

Include any specific usability requirements, for example,

Learnability

* The user documentation and help should be complete
* The help should be context sensitive and explain how to achieve common tasks
* The system should be easy to learn

(See <http://www.usabilitynet.org/>)

#### **Efficiency**

##### ***Performance Requirements***

Specify static and dynamic numerical requirements placed on the system or on human interaction with the system:

* Static numerical requirements may include the number of terminals to be supported, the number of simultaneous users to be supported, and the amount and type of information to be handled.
* Dynamic numerical requirements may include the number of transactions and tasks and the amount of data to be processed within certain time period for both normal and peak workload conditions.

All of these requirements should be stated in measurable form. For example, "95% of the transactions shall be processed in less than 1 second" rather than “an operator shall not have to wait for the transaction to complete”.

##### ***Space Requirements***

#### **Dependability**

**Availability**

Include specific and measurable requirements for:

* Hours of operation
* Level of availability required
* Coverage for geographic areas
* Impact of downtime on users and business operations
* Impact of scheduled and unscheduled maintenance on uptime and maintenance communications procedures
* reliability (e.g., acceptable mean time between failures (MTBF), or the maximum permitted number of failures per hour).

**Reliability**

**Monitoring**

Include any requirements for product or service health monitoring, failure conditions, error detection, logging, and correction.

**Maintenance**

Specify attributes of the system that relate to ease of maintenance. These requirements may relate to modularity, complexity, or interface design. Requirements should not be placed here simply because they are thought to be good design practices.

**Integrity**

#### **Security**

Specify the factors that will protect the system from malicious or accidental access, modification, disclosure, destruction, or misuse. For example:

* encryption
* activity logging, historical data sets
* restrictions on intermodule communications
* data integrity checks

Specify the Authorization and Authentication factors. Consider using standard tools such as PubCookie.

### **Organizational Requirements**

Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc

#### **Environmental Requirements**

#### **Operational Requirements**

#### **Development Requirements**

### **External Requirements**

* + Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

#### **Regulatory Requirements**

#### **Ethical Requirements**

#### **Legislative Requirements**

Specify the requirements derived from existing standards, policies, regulations, or laws (e.g., report format, data naming, accounting procedures, audit tracing). For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values

##### **Accounting Requirements**

##### **Security Requirements**

## ***Domain Requirements***

Everything related to the domain that might be needed in the project shall be mentioned here. Sometimes the domain Requirements might be thought of as part of either functional or non-functional requirements.

Please provide all necessary non-functional requirements, similar to the requirements explained in the lesson slides or in the textbook.

# User Scenarios/Use Cases

Provide a summary of the major functions that the product will perform. Organize the functions to be understandable to the customer or a first time reader. Include use cases and business scenarios, or provide a link to a separate document (or documents). A business scenario:

* Describes a significant business need
* Identifies, documents, and ranks the problem that is driving the scenario
* Describes the business and technical environment that will resolve the problem
* States the desired objectives
* Shows the “Actors” and where they fit in the business model
* Is specific, and measurable, and uses clear metrics for success

APPENDIX

The appendixes are not always considered part of the actual Requirements Specification and are not always necessary. They may include

* Sample input/output formats, descriptions of cost analysis studies, or results of user surveys;
* Supporting or background information that can help the readers of the Requirements Specification;
* A description of the problems to be solved by the system;
* Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements.

When appendixes are included, the Requirements Specification should explicitly state whether or not the appendixes are to be considered part of the requirements.

1. **Definitions, Acronyms, and Abbreviations**

Define all terms, acronyms, and abbreviations used in this document.

1. **References**

List all the documents and other materials referenced in this document.

1. **Requirements Traceability Matrix**

The following trace matrix examples show one possible use of naming standards for deliverables (FunctionalArea-DocType-NN). The number has no other meaning than to keep the documents unique. For example, the Bargaining Unit Assignment Process Flow would be BUA-PF-01.

For example (1):

| **Business Requirement** | **Area** | **Deliverables** | **Status** |
| --- | --- | --- | --- |
| BR\_LR\_01  The system should validate the relationship between Bargaining Unit/Location and Job Class.---Comments: Business Process = "Assigning a Bargaining Unit to an Appointment" (Priority 1) | BUA | BUA-CD-01  Assign BU Conceptual Design | Accepted |
| BUA-PF-01  Derive Bargaining Unit-Process Flow Diagram | Accepted |
| BUA-PF-01  Derive Bargaining Unit-Process Flow Diagram | Accepted |
| BR\_LR\_09  The system should provide the capability for the Labor Relations Office to maintain the job class/union relationship.---Comments: Business Process = "Maintenance" (Priority 1) | BUA | BUA-CD-01  Assign BU Conceptual Design | Accepted |
| BUA-PF-02  BU Assignment Rules Maint Process Flow Diagram | ReadyForReview |

For example (2):

| **BizReqID** | **Pri** | **Major Area** | **DevTstItems DelivID** | **Deliv Name** | **Status** |
| --- | --- | --- | --- | --- | --- |
| BR\_LR\_01 | 1 | BUA | BUA-CD-01 | Assign BU Conceptual Design | Accepted |
| BR\_LR\_01 | 1 | BUA | BUA-DS-02 | Bargaining Unit Assignment DB Modification Description | Accepted |
| BR\_LR\_01 | 1 | BUA | BUA-PF-01 | Derive Bargaining Unit-Process Flow Diagram | Accepted |
| BR\_LR\_01 | 1 | BUA | BUA-UCD-01 | BU Assign LR UseCase Diagram | ReadyForReview |
| BR\_LR\_01 | 1 | BUA | BUA-UCT-001 | BU Assignment by PC UseCase - Add Appointment and Derive UBU | Reviewed |
| BR\_LR\_01 | 1 | BUA | BUA-UCT-002 | BU Assignment by PC UseCase - Add Appointment (UBU Not Found) | Reviewed |
| BR\_LR\_01 | 1 | BUA | BUA-UCT-006 | BU Assignment by PC UseCase - Modify Appointment (Removed UBU) | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-CD-01 | Assign BU Conceptual Design | Accepted |
| BR\_LR\_09 | 1 | BUA | BUA-DS-02 | Bargaining Unit Assignment DB Modification Description | Accepted |
| BR\_LR\_09 | 1 | BUA | BUA-PF-02 | BU Assignment Rules Maint Process Flow Diagram | Accepted |
| BR\_LR\_09 | 1 | BUA | BUA-UCD-03 | BU Assign Rules Maint UseCase Diagram | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UCT-045 | BU Assignment Rules Maint: Successfully Add New Assignment Rule | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UCT-051 | BU Assignment Rules MaintUseCase: Modify Rule | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UCT-053 | BU Assignment Rules MaintUseCase - Review Assignment Rules | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UCT-057 | BU Assignment Rules MaintUseCase: Inactivate Last Rule for a BU | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UI-02 | BU AssignRules Maint UI Mockups | ReadyForReview |
| BR\_LR\_09 | 1 | BUA | BUA-TC-021 | BU Assignment Rules Maint TestCase: Add New Rule (Associated Job Class Does Not Exist) - Success | ReadyForReview |
| BR\_LR\_09 | 1 | BUA | BUA-TC-027 | BU Assignment Rules Maint TestCase: Modify Rule - Success | ReadyForReview |
| BR\_LR\_09 | 1 | BUA | BUA-TC-035 | BU Assignment Rules Maint TestCase: Add New Rule (Associated Job Class Does Not Exist) - Error Condition | ReadyForReview |
| BR\_LR\_09 | 1 | BUA | BUA-TC-049 | BU Assignment Rules Maint TestCase: Modify Rule - Error Condition | ReadyForReview |

For example (3):

| **BizReqID** | **CD01** | **CD02** | **CD03** | **CD04** | **UI01** | **UI02** | **UCT01** | **UCT02** | **UCT03** | **TC01** | **TC02** | **TC03** | **TC04** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BR\_LR\_01 |  |  | X |  | X |  | X |  |  | X |  | X |  |
| BR\_LR\_09 | X |  |  | X |  | X |  |  | X |  | X |  | X |
| BR\_LR\_10 | X |  |  | X |  |  |  |  | X |  | X |  |  |
| BR\_LR\_11 |  | X |  |  |  |  |  |  |  |  |  |  |  |

1. **Organizing the Requirements**

This section is for information only as an aid in preparing the requirements document.

Detailed requirements tend to be extensive. Give careful consideration to your organization scheme. Some examples of organization schemes are described below:

**By System Mode**

Some systems behave quite differently depending on the mode of operation. For example, a control system may have different sets of functions depending on its mode: training, normal, or emergency.

**By User Class**

Some systems provide different sets of functions to different classes of users. For example, an elevator control system presents different capabilities to passengers, maintenance workers, and fire fighters.

**By Objects**

Objects are real-world entities that have a counterpart within the system. For example, in a patient monitoring system, objects include patients, sensors, nurses, rooms, physicians, medicines, etc. Associated with each object is a set of attributes (of that object) and functions (performed by that object). These functions are also called services, methods, or processes. Note that sets of objects may share attributes and services. These are grouped together as classes.

**By Feature**

A feature is an externally desired service by the system that may require a sequence of inputs to affect the desired result. For example, in a telephone system, features include local call, call forwarding, and conference call. Each feature is generally described in a sequence of stimulus-response pairs, and may include validity checks on inputs, exact sequencing of operations, responses to abnormal situations, including error handling and recovery, effects of parameters, relationships of inputs to outputs, including input/output sequences and formulas for input to output.

**By Stimulus**

Some systems can be best organized by describing their functions in terms of stimuli. For example, the functions of an automatic aircraft landing system may be organized into sections for loss of power, wind shear, sudden change in roll, vertical velocity excessive, etc.

**By Response**

Some systems can be best organized by describing all the functions in support of the generation of a response. For example, the functions of a personnel system may be organized into sections corresponding to all functions associated with generating paychecks, all functions associated with generating a current list of employees, etc.

**By Functional Hierarchy**

When none of the above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be used to show the relationships between and among the functions and data.

**Additional Comments**

Whenever a new Requirements Specification is contemplated, more than one of the organizational techniques given above may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification.

There are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; and when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.