majmuni D lufi

Hospital Management System **Requirements Specification**

Version 1.0

April 21, 2022

Table of Contents

1. Executive Summary

1.1 Project Overview

This is a Project for the Software Engineering course at Epoka University, created by a group of 6 students of CEN. The project is a web site/application designed to help Hospitals, here in Albania, manage their operation in a more effective and convenient way.

1.2 Purpose and Scope of this Specification

The Main objective of the Web application is to allow Hospitals here in Albania a system to interact better and more conveniently with patients as well as having an easier time managing different aspects of running an effective Hospital. One of the main aspects of this solution is a way for patients to have an easier time looking for doctors and booking and appointment with specialists in regards to their needs. As for the hospital side, it will be a website that will allow an easier time to the hospital staff to manage their interaction with patients as well as the documentation needed for them (i.e. “Librez shendetsore”, analysis results, x-rays, MRI, ect).

In scope:

- This project is responsible for creating a platform where patients can contact and book appointments, as well as being able to see which doctors are available in the Hospital at which times.
- On the Hospital side, the web application will be able to create appointments for the Doctors. Will be able to archive and manage the necessary documents to run the hospital.
- There will be web security for the website, as well as 2FA for the login giving the log in process an extra layer of security.
- Maintenance of the web service will be provided by us.
- A mobile app will be developed to offer the booking services more easily.

Out of scope:

- The Web Application (and hence our service) will not be responsible for any monetary transaction between the Hospital and its patients.
- Changes will be made to accommodate for legislative requirements and changes along the way.

2. Product/Service Description

A website that serves at the same time the possible/current patients of the hospital, as well as the hospital staff/management to manage their appointments, their schedule as well as their documents.

2.1 Product Context

For now a good way to demonstrate the context of our project would be the diagrams made available on the projects repository on GitHub :
https://github.com/EloiSherifi/SoftwareEngineeringProject/tree/main/Diagrams

2.2 User Characteristics

User :

- Register
- Log-in
- Request appointment
- View own profile
- View available Doctors
- Use filters to better find the appropriate Doctor
- Delete their own profile
- Review their booked times
- Look up when a certain doctor is free.
- Look at their appointment history
- Look and print their X-Rays/MRI/Receipts

Doctor/Medical staff:

- Log-in
- Check his assigned patients / his history of patients
- Check their timetable
- Check their requested appointments and accept/deny them
- Upload Receipts into their patients profile
- Look at medical documents and the medical history of their patients

Non-medical staff:

- Log-in
- Look at their schedule
- In the case of Reception, request an appointment for an in-place patient.
- Fill up necessary documents for their respective work

Admin:

- Log-in
- Create/Delete/Change internal accounts (Medical and non-medical users)
- Archiving former staff members account and their respective documents

Guest:

- View the front page
- Look up the Frequently asked questions page
- Get the Hospital Contacts.

2.3 Assumptions

Our assumptions are that the staff members will adapt the use of this software and integrate it into their daily work. It’s assumed that the technical background of the users of this software is of a degree that renders the use of the website intuitive. It’s assumed that the scheduled appointments will be respected and the doctors as well as the patients will be on time.

2.4 Constraints

- Deadline : End of course/semester (spring semester of 2022)
- Technologies used : HTML, CSS, JavaScript, Bootstrap, PHP, MySql.
- The timetable will be updated by the internal staff of the Hospital and will be respected.

2.5 Dependencies

List dependencies that affect the requirements. Examples:

- This new product will require a daily download of data from X,
- Module X needs to be completed before this module can be built.

3. 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).

3.1 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**
01	The system should associate a supervisor indicator with each job class.	Business Process = “Maintenance	3	7/13/04	Bob Dylan, Mick Jagger
02	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
03	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
04	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		
05	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		

3.2 *Non-Functional Requirements*

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

3.2.1 Product Requirements

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

3.2.1.1 **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).

3.2.1.2 **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/)

3.2.1.3 **Efficiency**

3.2.1.3.1 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”.

3.2.1.3.2 Space Requirements

3.2.1.4 **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

3.2.1.5 **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.

3.2.2 Organizational Requirements

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

3.2.2.1 **Environmental Requirements**

3.2.2.2 **Operational Requirements**

3.2.2.3 **Development Requirements**

3.2.3 External Requirements

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

3.2.3.1 **Regulatory Requirements**

3.2.3.2 **Ethical Requirements**

3.2.3.3 **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

3.2.3.3.1 Accounting Requirements

3.2.3.3.2 Security Requirements

3.3 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.

4. 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.

Appendix A. Definitions, Acronyms, and Abbreviations

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

Appendix B. References

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

Appendix C. 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

Business Requirement	Area	Deliverables	Status
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_0	1	BUA	BUA-TC-027	BU Assignment Rules Maint TestCase: Modify	ReadyForReview

BizReqID	Pri	Major Area	DevTstItems DelivID	Deliv Name	Status
9				Rule - Success	
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											

Appendix D. 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.