Software Requirements Specification

for

Authentication System

Version 1.0 approved

Prepared by

Tamara Nicole Rodríguez Luna

Jeffrey Daniel Leiva Cascante

Technological Institute of Costa Rica

May 10, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

Tamara Rodríguez Jeffrey Leiva	10-5-2023	Document creation.	1
Tamara Rodríguez Jeffrey Leiva	16-5-2023	changes to sections 1.1 - 1.7.	2

1. Introduction

1.1 Purpose

The purpose of the project exists due to a need for the members of a university to be able to authenticate themselves through an email and password. Authentication is extremely vital in this world of technology due to security and data privacy issues. This system allows authentication by means of a login that would be the access, but it is not limited only to function as a login, it also has the options of registration, account management and system password change. It is very important that the system reflects usability, security and good software development practices.

1.2 Document Conventions

Doesn't Apply.

1.3 Intended Audience and Reading Suggestions

This document provides explanations for the following audiences: system users (Principal, financial principal, students, teachers, administrative and providers), technical domain users (software developers) and all possible stakeholders of the system.
This document is divided into six numbered sections, they are: introduction, overall description, external interface requirements, system features, non-functional requirements, and other requirements. The scope of this document if you are an user who is unfamiliar with requirements engineering or technology in general, it is recommended that you focus on sections 1 and 2 of the documents because they do not require prior technical knowledge for their correct understanding. If you want to know more about the interface or non-functional requirements of the system, you can read sections 4 and 5 of the system. Section 3 is for system developers because of its terminology and that it focuses on understanding the system architecture.
If you are a developer, section 1 and 2 encompasses the idea of what the system should do and how it behaves, as well as understanding the points of view of the different users of the system. In section 3 you can find the architecture and functionality of the system with all its focus on classes, objects, attributes and other important to understand the internal structure of the system.
Section 4 is about the interface of the system and how each button, window and functionality should look and understand, section 5 is about the non-functional requirements such as appearance, security and so on. Finally, section 6 deals with the system requirements that include the UML diagrams that allow us to communicate with the non-relational database as well as with the use cases used and implemented in the system.

1.4 Product Scope

The product consists of an authentication system for a college with four principal functions that make up the entirety of the scope that the product covers. Those functionalities include: registration of users into the system, self-administration of the details of a user account, change of the password linked to a user account and the log-in to the system functionality. Together, these functionalities give form and define a scope for the product.

In general, the overarching goal of the system is to authenticate users through an email and a password and give them access to the different options that the system offers, these include the administration of the account's data as well as the possibility to change the password. However, the system is designed so that new functionalities can be added in the future if the users of the system require them.

In addition; if they do not have an account, there is the functionality of self-registration in the system as stipulated in the requirements given for the architecture of the system. Furthermore; the system also complies with the specified requirements for self-administration of the user account as well as including the functionality of the password change requested by the users. Finally, the system also complies with all the non-functional requirements specified; these include the feeling and look of the system as well as naming and code conventions that give the system structure cohesion.

1.5 User Story

This section is concerned with the detailed description of the four user stories that describe the use cases and the functionality that the system described in this specification needs to implement and deploy. These user stories are linked to the role of a user.

The first user story explains the requirements as per the user point of view for the use case of the registration in the authentication system, the expected behavior of the system at that stage, any alternatives that could apply to the design as well as the exceptions to the particular use case.

As a user of the authentication system, I need to be able to register into the system by creating an account with my personal information, it could be my email address, my physical address, name, surname and of course, a password. Furthermore, the password field should not be plainly visible; it must be hidden, even when I write the password I want. I would like the fields of the registration to be labeled as to which one corresponds to which information.
Likewise, there should be a button to confirm my registration; when I do click it, I would like to see a message that confirms that I am registered in the system. If for some reason I forgot to enter information into a field or I violated some restriction that doesn't allow the registration I want to see a message that points it out to me so I can fix my mistake. Additionally; there should be a button to go back to the login page, so that I can go when I'm done registering or I want to go back without doing so. If I click the exit button of the window it should close the system without any inconvenience. Finally, the register window of the system should be elegant and welcoming.

The second user story explains the requirements as per the user point of view for the use case of the log-in into the authentication system, the expected behavior of the system at that stage, any alternatives that could apply to the design as well as the exceptions to the particular use case.

As a user of the authentication system, I need to be able to log-in to my account on the system by providing my credentials; these are my email address and chosen password. When I go to the log-in window of the system I want to see the corresponding fields for my credentials, these should be labeled as to which one is for the password and which one is for the email address. Moreover, the password should not be plainly visible, even when I'm writing it.
There should also be a button to confirm the log-in; when I do click it, I want to see the homepage of the system if my credentials were right. If for some reason, I did not enter the corresponding information into a field or I violated some restriction I want to see a message that explains what I did wrong so I can correct it. Also, if the credentials do not match with my account, I should not be able to log-in. Additionally, there should also be a link to the register window so that I can register if

I don't have an account. Furthermore, when I click the exit button the system must close and shut down. Finally, the log-in window must have an elegant and welcoming feeling.

The third user story explains the requirements as per the user point of view for the use case of the administration of the user account in the authorization system, the expected behavior of the system for the particular requirement, the alternatives that could apply to the design as well as the exceptions for the presented use case.

As a user of the authentication system, I want to be able to change or update the information of my account in the system. For this, in the homepage I want to see an option that lets me administer my information. When I click that option I want to see a window that displays my current information as well as a button with the option to edit it. When I click the edit button I want the fields displaying my information to be able to receive changes. So, when I'm done making the changes I want to be able to save them by clicking a button. If for some reason I left fields blank or I violated some restriction I want to see a message with the error so that I can fix it to proceed with the changes and the displayed information should return to the one that was previously shown. If the changes were successful, I would like to see a message pointing it out and the updated information in the window. There also should be a button to go back to the homepage and when I click it I want to see the homepage with the options of administering my account and the change password option. Furthermore, when I click the exit button the system must close and shut down. Additionally, the window must have the same feeling as the others of the system.

The fourth user story explains the requirements as per the user point of view for the use case of changing the password of the account in the authorization system, the expected behavior of the system for the particular requirement, the alternatives that could apply to the design as well as the exceptions for the presented use case.

As a user of the authorization system, I want to be able to change the password of my account in the system. In the homepage of the system I want to see an option for the change of password and when I click it, the change password window should be shown. In the window for the change of password there should be at least three fields, one that is for the current password, for security reasons and two more fields for the new password and its confirmation.
There should also be a button to confirm the change when I'm done providing the information. When I click the said button and everything is correct I would like to see a message that confirms the change. If for some reason I left blank fields or the information I typed is wrong, I want to see a message that tells me what went wrong so I can correct it and try again. In addition to this, there should be a button to go back to the homepage and when I click it I want to see the homepage with the options of administering my account and the change password option. Furthermore, when I click the exit button the system must close and shut down. Finally, the window must have the same feeling of elegance and welcoming as the others of the system.

The previously specified user stories cover the requirements of the authorization system, both functional and non-functional as per the user point of view. They will serve as a guideline for the next sections of the system specification as well as the way the system should be architectured, programmed, polished and tested to achieve the desired results

1.6 References

Doesn't Apply.

1.7 Overview

In the next section of the document you can find the general description of it, where the use cases of the users and characteristics of the project such as its design, dependencies and others. The idea of the next section is to understand the product, the prospects, functionality and other important aspects.

2. Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

2.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security

considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3. External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be

implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:
REQ-2:

4.2 System Feature 2 (and so on)

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>