Software Requirements Specification

for

Authentication System

Version 2.1 approved

Prepared by

Tamara Nicole Rodríguez Luna

Jeffrey Daniel Leiva Cascante

Technological Institute of Costa Rica

May 24, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

Tamara Rodríguez Jeffrey Leiva	10-5-2023	Document creation.	1.0
Tamara Rodríguez Jeffrey Leiva	16-5-2023	Changes to sections 1.1 - 1.7.	1.1
Tamara Rodríguez Jeffrey Leiva	18-3-2023	Section aggregation 1.6 - Definitions, Acronyms or Abbreviations, Creation of section 2 and 3 of the document.	2.0
Tamara Rodríguez Jeffrey Leiva	24-5-2023	Changes on section 2 and 3 of the document.	2.1

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 Definitions, Acronyms or Abbreviations

Authenticator System: It is a software that limits and allows the registration of unauthorized users.

Bootstrap is a free and open source CSS framework aimed at mobile and responsive front-end web development.
CSS: A style sheet language used to describe the presentation of a document written in a markup language such as HTML.
Desktop: refers to the window where the system functionalities are found.
Express: It is a Javascript library, it is designed to create web applications and APIs.
Front-End: is the practice of converting data to a graphical interface, through the use of HTML, CSS, and JavaScript, so that users can view and interact with that data.
HTML: HyperText Markup Language.
IDE: Integrated Development Environment.
JavaScript: It is a programming or scripting language that allows you to implement complex functions in web pages.
Main: It is where the program begins its execution. It is the function that controls the flow of the program.
MongoDB: is a non-relational database management system.
Node.js – Asynchronous event-driven Javascript runtime that can build scalable networked applications.
Non-relational: is one that does not use the tabular layout of rows and columns.
Blind: Function that prevents you from seeing what was written in a text field.
Responsive: It is a design and development philosophy whose objective is to adapt the appearance of web pages to the device that is being used to visit them.
Salt – Comprises random bits that are used as one of the inputs in a key-derivative function.
Self-service: It is a philosophy that establishes that the user attends to himself, does not need any employee to carry out an action.
Unauthorized user: person that is not registered on the system and has no access to the system.
User: person that is registered on the system and has access to the system, The user of the system can be any person from the university, be it principal, financial principal, students, teachers, administrative and providers.

1.7 References

The page where the MongoDB non-relational database is located is:
https://www.mongodb.com/

1.8 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

The authenticator system is built on a website, it could be used on any system that needs to authenticate its users. For it to work you must need a browser and a connection to the MongoDB database.

The system allows you to manage user access to system functionalities such as managing your account and changing your password, as well as allowing unauthorized users to create a new account and access the system with this created account.

2.2 Product Functions

The system is made up of four main use cases: access, registration, account management and password change. They are described in the following class diagram:

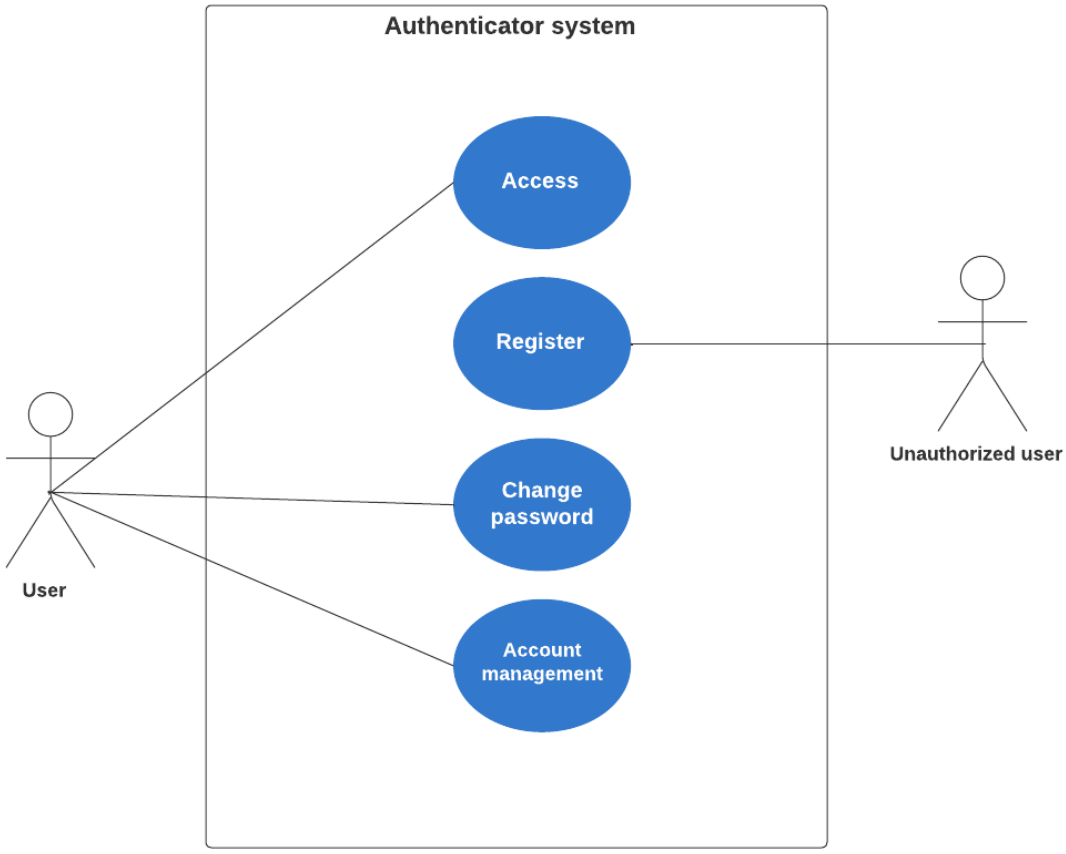

Figure 2.2.1: Use case diagram

List of use cases:

Actor	*Function*
User	*Access*
Unauthorized user	*Register*
User	*Account management*
User	*Change password*

Access Use Case

Use Case ID:	SRS-000		
Use Case Name:	Access		
Created By:	Jeffrey Leiva	Last Updated By:	Tamara Rodríguez
Date Created:	March 4, 2023	Date Last Updated:	May 24, 2023

Actors:	User
Description:	Functionality that allows access to the system by a duly authorized actor by identifying their credentials.
Trigger:	The user clicks on the login button.
Preconditions:	1. The user must have a profile and credentials previously registered in the system, that is, an email and password associated with a previously registered account. 2. The data must be contained in the system's MongoDB database so that the data given by the user can be verified.
Postconditions:	1. The user has access to the system and its functionalities. 2. You are not allowed access if you are an unauthorized user or your credentials are invalid.

Normal Flow:	1. The user enters the login window. 2. The user enters his credentials, that is, the email and password in the corresponding boxes. 3. The user clicks on the "Next" button. 4. The system verifies that the data entered coincides in the database and that they are correct. 5. The user logs in successfully. 6. The system shows the desktop to the user.
Alternative Flows:	After the first step the user can do- the following alternative: 2. The user closes the window. 3. The user is redirected to his browser if he has other windows open or to the desktop of the computer.
Exceptions:	In step 4, the system rejects the user's data because they are incorrect or did not match, so it would follow the following flow steps: 4. The system verifies that the data entered does not match in the database and that they aren't correct. 5. The system displays an error message. 6. The user returns to point 2 of the normal flow.
Includes:	1. Verify the existence of the user in the system database 2. Verify login credentials against the database.
Priority:	1
Frequency of Use:	Daily.
Business Rules:	The system must be under Microsoft-Like.
Special Requirements:	1. The access must be self-service. 2. The system must be developed with Microsoft-Like.

Assumptions:	1. Access to the system was developed to be authenticated by means of an email and password. 2. The MongoDB database is used for data persistence.
Notes and Issues:	Doesn't Apply.

Register Use Case

Use Case ID:	SRS-001		
Use Case Name:	Register		
Created By:	Jeffrey Leiva	Last Updated By:	Nicole Rodríguez
Date Created:	March 4, 2023	Date Last Updated:	May 24, 2023

Actors:	User.
Description:	System functionality that allows an unauthorized user to register.
Trigger:	Enter the Login window, then click the register button.
Preconditions:	1. The database must be available to store the new information. 2. The unauthorized user must have a valid email for registration.

Postconditions:	1. The unauthorized user happens to have a valid account and be a normal user of the system.
Normal Flow:	1. The system presents the login window. 2. The unauthorized user presses the register button. 3. The system presents the form requesting user data. 4. The unauthorized user registers data in the form, fills in the spaces and presses the confirm button. 5. The system verifies information. 6. The system displays a message that the registration was successful. 7. The user is returned to log in.
Alternative Flows:	After the first step the user can do- the following alternative: 2. The user closes the window. 3. The user is redirected to his browser if he has other windows open or to the desktop of the computer. After the third step of the system the following could happen: 4. The user closes the window. 5. The user is redirected to his browser if he has other windows open or to the desktop of the computer. After the second step of the system the following could happen: 3. The user presses the return button. And the user is back at step 1 of the normal flow.

Exceptions:	After step 5 the following exception may occur: 6. The system displays an error message because the email is already associated with another account. 7. The unauthorized user remains in the registration window and returns to step 4 of the normal flow. After step 5 the following exception may occur: 6. The system shows an error message because the mail space is empty. 7. The unauthorized user remains in the registration window and returns to step 4 of the normal flow. After step 5 the following exception may occur: 6. The system displays an error message because the password space is empty. 7. The unauthorized user remains in the registration window and returns to step 4 of the normal flow. After step 5 the following exception may occur: 6. The system displays an error message because the name space is empty. 7. The unauthorized user remains in the registration window and returns to step 4 of the normal flow. After step 5 the following exception may occur: 6. The system displays an error message because the address space is empty. 7. The unauthorized user remains in the registration window and returns to step 4 of the normal flow.

Includes:	1. The database must be available to store the new information. 2. Valid email verification.
Priority:	2
Frequency of Use:	Daily
Business Rules:	The system must be under Microsoft-Like.
Special Requirements:	1. The access must be self-service. 2. The system must be developed with Microsoft-Like.
Assumptions:	1. The registration to the system was developed to be authenticated by name, email, address and password. 2. The MongoDB database is used for data persistence.
Notes and Issues:	Doesn't Apply.

Change password Use Case

Use Case ID:	SRS-002		
Use Case Name:	Change password		
Created By:	Jeffrey Leiva	Last Updated By:	Nicole Rodríguez
Date Created:	March 4, 2023	Date Last Updated:	May 24, 2023

Actors:	User
Description:	Functionality that allows the duly authorized user to change their password.
Trigger:	Have successfully logged in, have the Landing Place window displayed and have clicked on the option to Change Password.
Preconditions:	1. The user must exist in the database. 2. The user must have successfully accessed the system. 3. 3. The user must have clicked on the change password functionality.
Postconditions:	1. The password is successfully updated in the database.
Normal Flow:	1. The user makes successful access to the system. 2. The system displays the desktop window. (Landing Place) 3. The user clicks on the change password button. 4. The user displays the password change form. 5. The user fills in the fields for the current password, the new password, and the confirmation of the new password. 6. The user clicks on the confirm button. 7. The system displays a successful change message. 8. The user is returned to the desktop. (Landing Place)

Alternative Flows:	After the first step the user can do- the following alternative: 2. The user closes the window. 3. The user is redirected to his browser if he has other windows open or to the desktop of the computer. After the fourth step the user can do- the following alternative: 2. The user closes the window. 3. The user is redirected to his browser if he has other windows open or to the desktop of the computer. After the fifth step the user can do- the following alternative: 2. The user closes the window. 3. The user is redirected to his browser if he has other windows open or to the desktop of the computer. 4. The password change does not happen because the confirm button is not pressed..
Exceptions:	After step 5 can occur: 6. The system shows an error because the current password is not correct. User proceeds to step 5. After step 5 can occur: 6. The system shows an error because the new password and the password confirmation do not match. User proceeds to step 5.
Includes:	1. Check the existence of the user in the database 2. Update of the password in the database.
Priority:	1
Frequency of Use:	Daily.
Business Rules:	The system must be under Microsoft-Like.
Special Requirements:	1. The access must be self-service. 2. The system must be developed with Microsoft-Like.

Assumptions:	1. In the password change we assume that the format of the password change would be the spaces of the current password, the new password and confirm the same. 2. The MongoDB database is used for data persistence.
Notes and Issues:	The password uses an encryption system for system security.

Account Management Use Case

Use Case ID:	SRS-003		
Use Case Name:	Account Management		
Created By:	Jeffrey Leiva	Last Updated By:	Nicole Rodríguez
Date Created:	March 4, 2023	Date Last Updated:	May 24, 2023

Actors:	User
Description:	Functionality that allows the duly authorized user to manage their account and account data.
Trigger:	Have successfully logged in, have the Landing Place window displayed and have clicked on the option to Manage Account.

Preconditions:	1. The user must exist in the database. 2. The user must have successfully accessed the system. 3. The user must have clicked on the change password functionality.
Postconditions:	Changes made to the account are saved.
Normal Flow:	1. The user makes successful access to the system. 2. The system displays the desktop window. (Landing Place) 3. The user clicks on the manage account button. 4. The user displays the managed account change form. 5. The user fills in the information field that he wishes to update. 6. The system displays a successful change message. 7. The user is returned to the desktop. (Landing Place)
Alternative Flows:	After the first step the user can do- the following alternative: 2. The user closes the window. 3. The user is redirected to his browser if he has other windows open or to the desktop of the computer. After the fourth step the user can do- the following alternative: 2. The user closes the window. 3. The user is redirected to his browser if he has other windows open or to the desktop of the computer. After the fifth step the user can do- the following alternative: 2. The user closes the window. 3. The user is redirected to his browser if he has other windows open or to the desktop of the computer. 4. The change of any of your data does not occur because the confirmation button is not pressed.
Exceptions:	After step 5 may occur: 6. The system displays an error because the data entered is in an invalid format. The user continues with step 5.

Includes:	1. Check the existence of the user in the database 2. Updating of data in the database.
Priority:	1
Frequency of Use:	Daily.
Business Rules:	The system must be under Microsoft-Like.
Special Requirements:	1. The access must be self-service. 2. The system must be developed with Microsoft-Like.
Assumptions:	1. It is assumed that all account data can be changed at any time. 2. The MongoDB database is used for data persistence.
Notes and Issues:	Doesn't Apply.

2.3 User Classes and Characteristics

In the system there are authorized users and unauthorized users, those who are not authorized can register in the system and at the end of the registration they become users of the system and as such; they can be authorized. Authorized users can access the system and its functionalities such as account management and password change.

The user base of the authorization system is made up of the different members that make up the college environment. These include; to name a few: students, professors, administrative personnel, financial chief officers, and providers. All of the previous mentioned will potentially make use of the system and its functionality. Accordingly; we must make a description of the characteristics of the defined users; the intention is to cover a broad set of them and this will be done in the next paragraphs.

The first characteristic that is worth mentioning is that all of the users mentioned above have an extensive record of using the Windows operating system; and as such; the vast majority of them have become accustomed to the applications that are available in it. It is crucial to understand this because the system must be developed taking into account the guidelines given by Microsoft to build, test, and deploy the software. Foremost; the ease of usability, the look and feel of the system should be alike to that of Microsoft's applications; these will be the point of discussion in the following characteristics in a detailed manner.

A second important characteristic of the users is that most of them have a certain degree of knowledge on the security and disposal of their data because the applications and systems they have used comply with high standards of security and they remove the additional difficulty this imposes off of the user's shoulders. This characteristic matters to us because it englobes a wide range of sections that we must take into account when developing the system; some of them include the following: the manner in which the user's data is handled, how sensitive or personal information is treated, as well as taking into account guidelines to guarantee the security of their account, including passwords and the corresponding encryption.

In general, all of these users or the majority of them have some knowledge in how to use these types of systems; and they are accustomed to them being self-service and end-user oriented. This means they want to be able to manipulate the system on their own without the need of help from the IT department or others. The users want to be able to do what they consider to be simple tasks like changing their password or account information by themselves. This is due to their previous background and experience in similar systems where most of the available functionality is self-serviced.

In conclusion It is of the utmost importance to take consideration of the previously defined user characteristics so that the product can comply and meet all the standards of quality and usability expected by all of the users of the system.

2.4 Operating Environment

The system is intended for it to be used in a web environment due to it being implemented via JavaScript and Nodejs. The system is designed so that it can be deployed in the browsers more widely used by the Windows operating system as it is the system that most users are accustomed to. Google Chrome and Edge fall in this category as they are two of the most well known browsers by users on this operating system. The software has been tested in both of these browsers, ensuring that it runs in a smooth manner for the end user.

2.5 Design and Implementation Constraints

The main constraints of the system are:

1. *The system must be used on a computer due.*
2. *It must be responsive, so that it adapts to the resolution of any browser and screen.*
3. *The database engine used is MongoDB.*
4. *The system is web type.*
5. *The system must use the self-service philosophy.*
6. *The way to identify a user corresponds to his mail.*
7. *The password must be encrypted.*

2.6 User Documentation

The user is given a requirements specification.

2.7 Assumptions and Dependencies

The main system dependencies are:

1. *The Internet host service LocalHost.*
2. *The style classes provided by Bootstrap.*
3. *Web use with HTML, CSS and JavaScript.*
4. *Using Node.js.*
5. *The library: express.*
6. *MongoDB database engine.*

3. External Interface Requirements

3.1 User Interfaces

The application consists of 3 layers, one for the graphical interface, another for the application and the last one for the persistent data. On the other hand, in the business logic, the type of user who is going to use the interface windows was identified as a normal user or an unauthorized user.

In the interface we can see multiple windows such as Login which consists of two login and registration buttons, in addition to two text fields for the email name and password, the unauthorized user registration window has 4 spaces: name, email email, address and password, in addition to a confirmation button. The desktop window (Landing Place) consists of the window that the user observes to use the system functionalities, in this project are the password change and administration from the account.
the existing user window consisting of an accept button and a text describing the completed action, the data does not match window is to inform the user that the email does not match the entered password, the password not allowed window informs him that the password entered does not comply with the established security standards and has an accept button, the email window does not exist informs the user that he is trying to enter with an email that is not registered in the database and has a button accept, the email exists window informs the user that he is trying to register with an email that is already registered in the database and has an accept button, finally the empty boxes error window tells the user that he still needs to complete a box to register or log in.
The change password functionality is based on three spaces to enter data, the passwords are always masked, the current password, the new password and the confirmation of the new password are entered, as well as a confirm button so that the changes are saved successfully. The account administration functionality consists of entering the data that you want to change and that the previous ones are replaced, the changes must be saved by clicking on the confirm button.
In the application there are functions such as registerUser that receives as parameters all the information that was requested from the unauthorized user, calls the verifyPassword function to confirm that the password complies with the established standards and matches its confirmation, upon receiving the true as return encrypts the password and asks the database to register the user with the data it received, access the system that only receives an email and the password, uses

the verifyUser function to confirm that the user exists and the password matches, after confirming it allows access to the system; In addition, it has all the functions to display the different windows of the interface.
The database saves all the information that was requested in the registration of each user and responds to requests to register a new user and search for a user to display the data as needed.

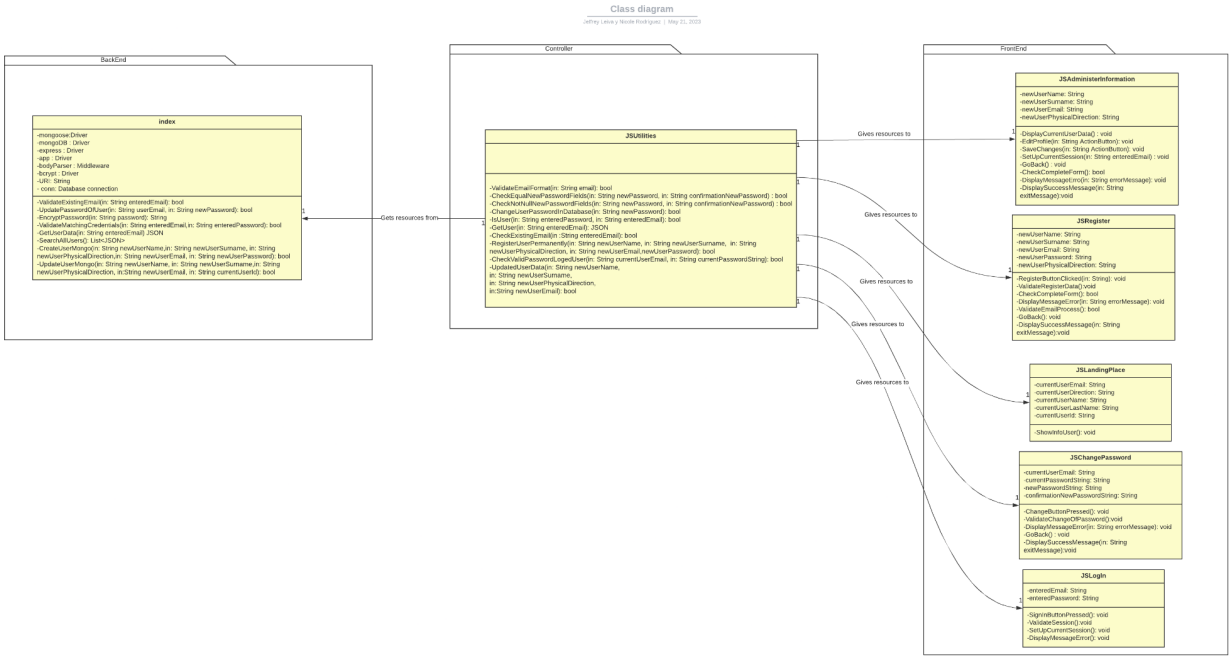

Figure 3.1: Class diagram

3.2 Hardware Interfaces

The system is intended for it to be used in a web environment due to it being implemented via JavaScript and Nodejs. The system is designed so that it can be deployed in the browsers more widely used by the Windows operating system as it is the system that most users are accustomed to. Google Chrome and Edge fall in this category as they are two of the most well known browsers by users on this operating system. The software has been tested in both of these browsers, ensuring that it runs in a smooth manner for the end user.

3.3 Software Interfaces

3.3.1 Access:

- *3.3.1.1 Priority and description:*
  The diagram in the next section represents the sequence of execution when logging in. First the unauthorized user enters the system through the browser, the function displays the Login window and the database in this case MongoDB needs to be active. In addition, the Main application creates a new Login Window object to display on the screen to the user. Then the unauthorized user fills in the required fields, which in this case are the username and password, and then through the Login button activate the function to access the system

and allow the JSUtilities program to read the data and verify if they match a document in the MongoDB database. If no problem occurs, the database returns true and the JSUtilities screen of the application is displayed on the screen through the browser. Finally, Main frees the memory corresponding to the Login window that was active for the session and it ends its life cycle.

- *3.3.1.2 Stimulus and Response Sequences:*

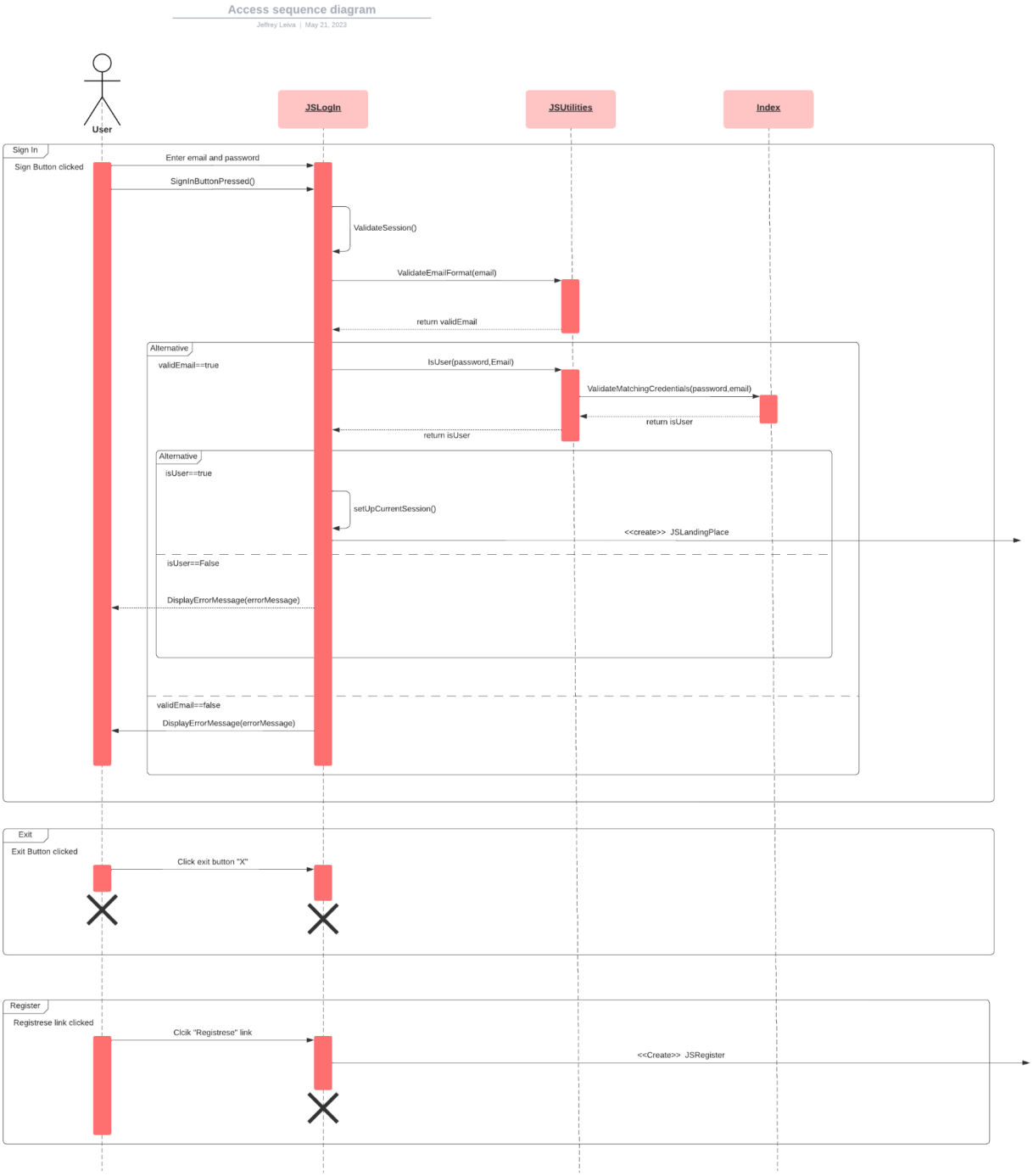

Figure 3.3.1.2.1: Access sequence diagram

- *3.3.1.3 Functional requirements:*

1. *REQ-1: The system must display the Login Window and receive information.*

2. *REQ-2: The system must create a new Login Window object in order to display it.*
3. *REQ-3: The system must process the request of the unauthorized user, validate and consult the user database. Go to figure 3.1.*
4. *REQ-4: The System must verify if the information is in the database and answer if the user exists or not.*
5. *REQ-5: The system should display the main screen once logged into the system.*
6. *REQ-6: The system must erase the memory belonging to the Login Window object once its use has finished.*
7. *REQ-7: The system must ensure that the database of data works correctly.*

3.3.2 Register:

- *3.3.2.1 Priority and description:*
  The diagram in the next section represents the execution sequence when registering a new user. First, when the unauthorized user enters the system in the browser, the function of displaying the Login window is activated, and the Main application creates a new Login Window object to display on the screen to the user. Then through the registration button the user activates the display function of the user form, the JSUtilities application creates a new Registration Window object to be displayed on the screen to the unauthorized user. JSUtilities frees the memory corresponding to the Login window that was active for the session and it ends its life cycle. Then the user fills in the required fields, and then, by means of the Accept button, activate the function of registering a new user to the system and allow the JSUtilities program to read the data and by means of the Verify existing user function if the data of the email is already registered or has an invalid format, in addition to verifying that passwords meet the requirements through the verify valid password function. If no problem occurs, the user will be registered in the database and a message that the registration was successful will be displayed on the user's screen. Finally, JSUtilities frees the memory corresponding to the Login window that was active for the session and it ends its life cycle.
- *3.3.2.2 Stimulus and Response Sequences:*

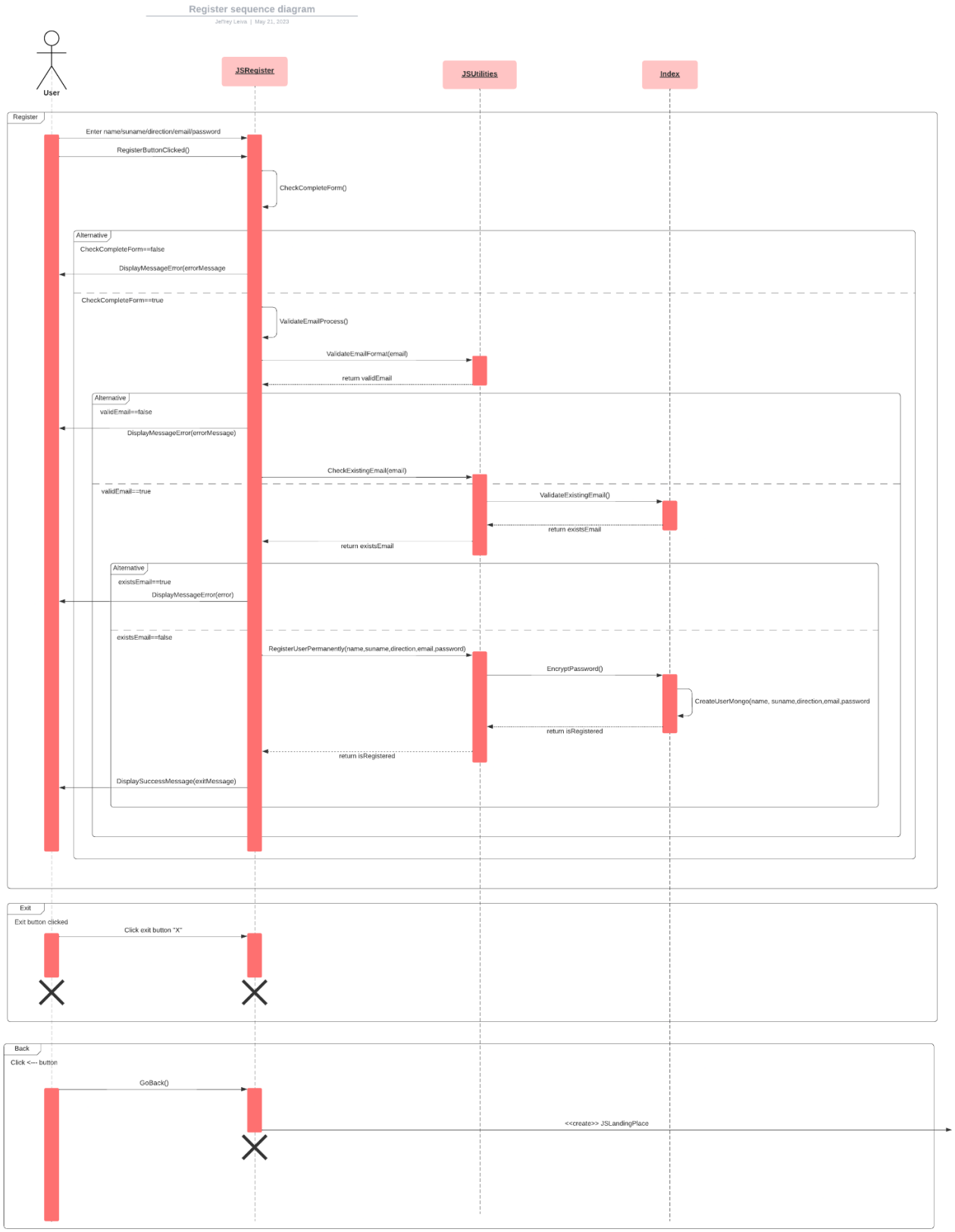

Figure 3.3.2.2.1: Register sequence diagram

- *3.3.2.3 Functional requirements:*

1. *REQ-1: The System must display the Registration Window and receive information.*
2. *REQ-2: The system must create a new Login Window object in order to display it.*
3. *REQ-3: The system must create a new User Template Window object in order to display it.*
4. *REQ-4: The System must process the user's request, validate and consult and add to the user database. Go to figure 3.1*
5. *REQ-5: The System verifies if the information is in the database and responds if the user exists or not.*
6. *REQ-6: The system should display the Registration Complete screen once the user has registered*
7. *REQ-7: The system must erase the memory belonging to the Login Window object once its use has finished.*
8. *REQ-8: The system must erase the memory belonging to the User Template Window object once its use has finished.*
9. *REQ-9: The system must ensure that the database of data works correctly.*

3.3.3 Change Password:

- *3.3.3.1 Priority and description:*
  El diagrama de la siguiente sección representa la secuencia de ejecución cuando el usuario desea cambiar su contraseña. Primeramente, cuando el usuario autorizado ya debe estar ingresado en el sistema, se activa la función de mostrar la ventana de cambio de contraseña y la aplicación principal crea un nuevo objeto de ventana de cambio de contraseña para mostrar en la pantalla al usuario. Luego, a través del botón de cambio de contraseña, el usuario activa la función del uso del formulario de cambio de contraseña, la aplicación JSUtilities crea un nuevo objeto de ventana de Landing Place para mostrar en la pantalla al usuario autorizado. JSUtilities libera la memoria correspondiente a la ventana del escritorio que estaba activa para la sesión y finaliza su ciclo de vida. Luego el usuario llena los campos requeridos, para luego mediante el botón de confirmar y que el programa JSUtilities lea los datos y mediante la función Verificar si los datos tienen un formato no válido, además de verificar que las contraseñas cumplen con los requisitos a través de la función verificar contraseña válida. Si no ocurre ningún problema, el usuario será registrado en la base de datos y se mostrará en la pantalla del usuario un mensaje de que el cambio de contraseña fue exitoso. Finalmente, JSUtilities libera la memoria correspondiente a la ventana de Cambio de contraseña que estaba activa para la sesión y finaliza su ciclo de vida.
- *3.3.3.2 Stimulus and Response Sequences:*

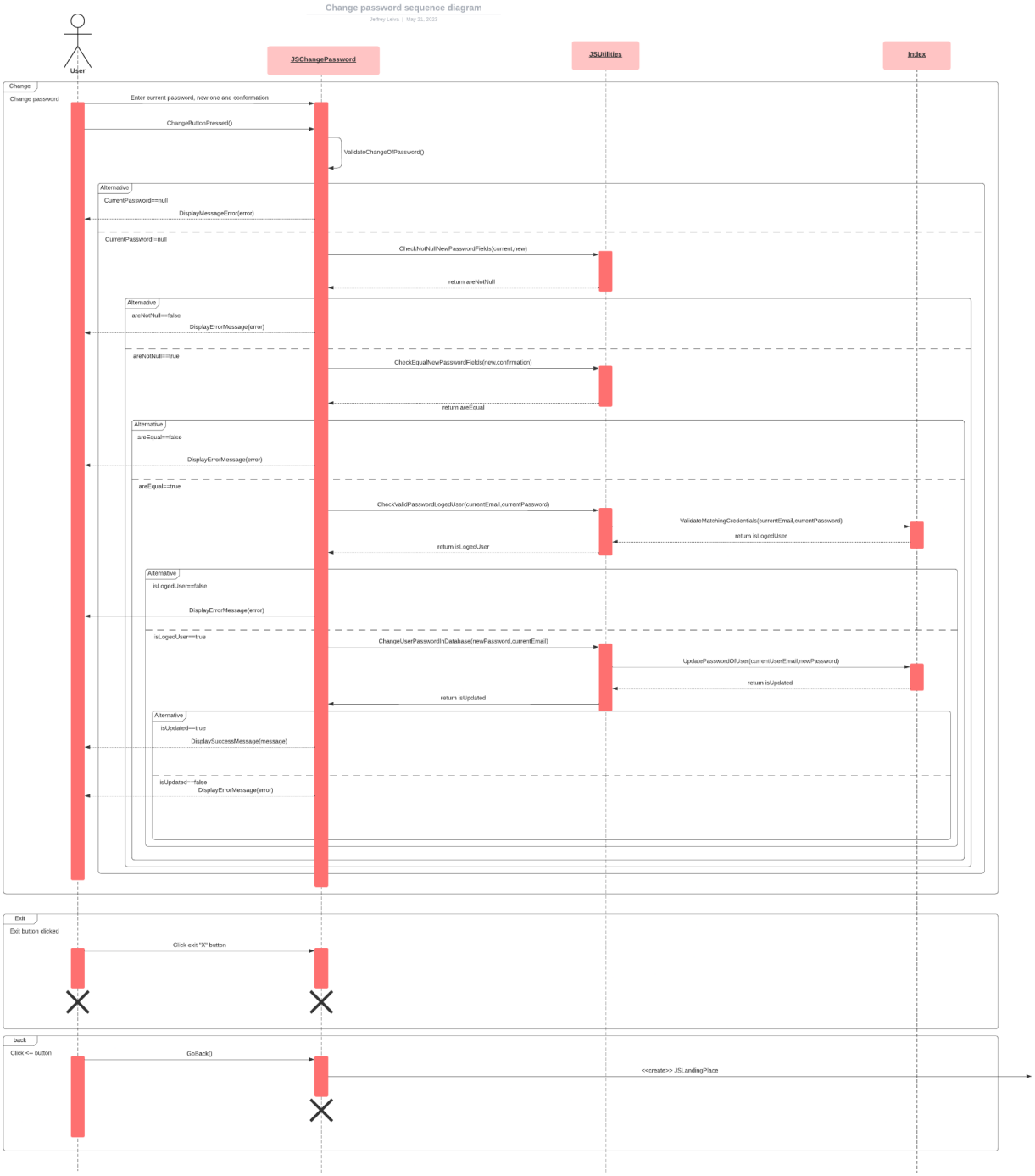

Figure 3.3.3.2.1: Change Password sequence diagram

- *3.3.3.3 Functional requirements:*

1. *REQ-1: The System must show the Password Change Window and receive information.*
2. *REQ-2: The system must create a new Password Change window object to display it.*
3. *REQ-3: The system must create a new password change form window object to display it.*
4. *REQ-4: The System must process the user's request, validate and consult and add to the user database. Go to figure 3.1*
5. *REQ-5: The System verifies if the information is in the database and the account data is displayed.*

6. *REQ-6: The system should display a message that the change was successful.*
7. *REQ-7: The system must erase the memory belonging to the Password Change Window object once its use is finished.*
8. *REQ-8: The system must clear the memory belonging to the Window object after its use has finished.*
9. *REQ-9: The system must ensure that the database works correctly.*
10. *REQ-10: The system must verify that the passwords match and that they comply with the established standards.*

3.3.4 Manage Account:

- *3.3.4.1 Priority and description:*
  The diagram in the next section represents the execution sequence when the user wants to manage the account. First, when the authorized user should already be logged into the system, the Landing Place window display function is activated and the main application creates a new account management window object to display on the user's screen. Then, through the account management button, the user activates the function of using the account management form, the JSUtilities application creates a new Landing Place window object to display on the screen to the authorized user. JSUtilities frees the memory corresponding to the desktop window that was active for the session and ends its life cycle. Then the user fills in the required fields, then using the confirm button and having the JSUtilities program read the data and using the Verify if the data has a valid format function. If no problem occurs, the user will be registered in the database and a message that the data change was successful will be displayed on the user's screen. Finally, JSUtilities frees the memory corresponding to the account administration window that was active for the session and ends its life cycle.
- *3.3.4.2 Stimulus and Response Sequences:*

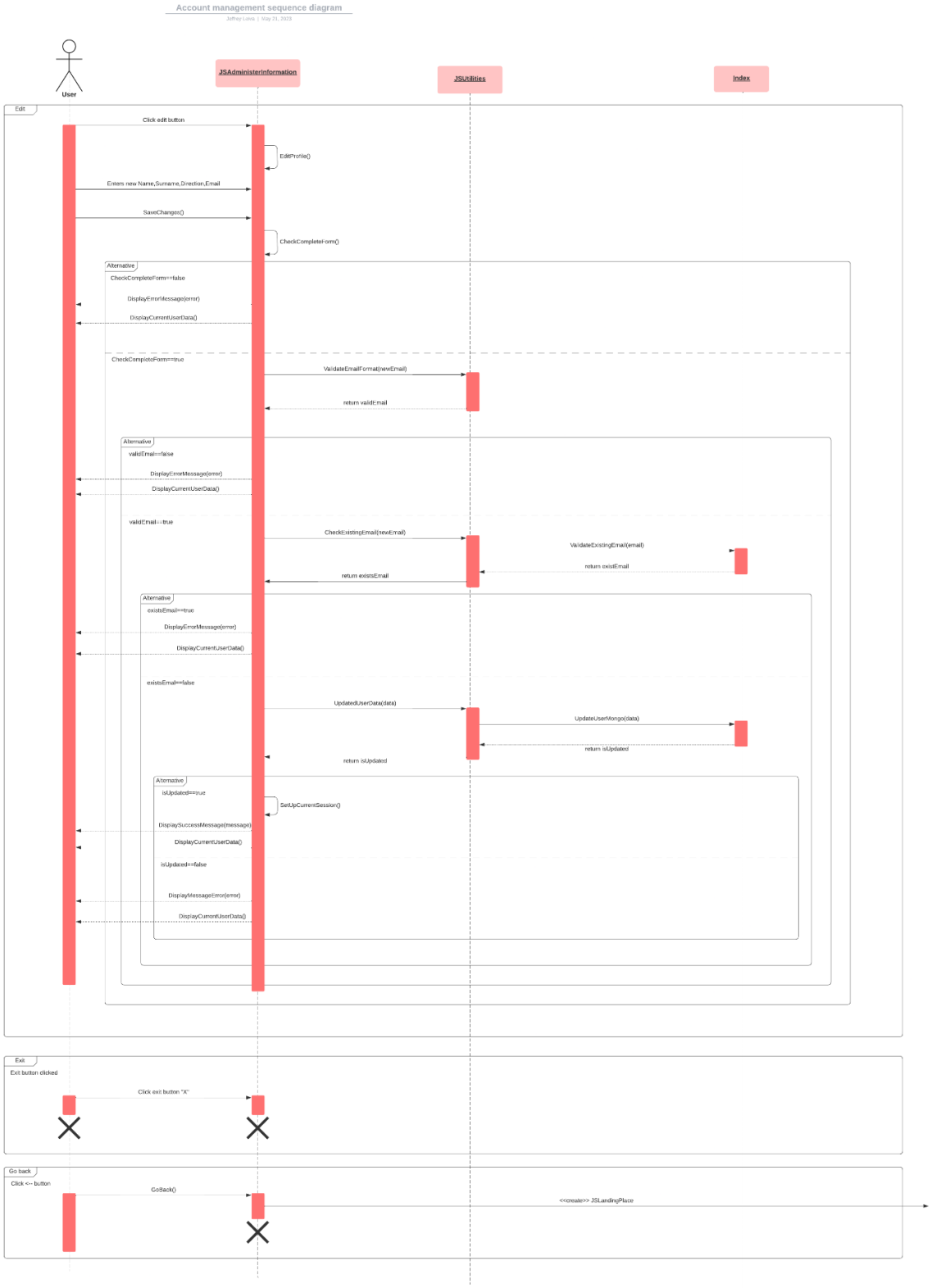

Figure 3.3.4.2.1: Change Password sequence diagram

- *3.3.4.3 Functional requirements:*

1. *REQ-1: The System must show the Account Management Window and receive information.*
2. *REQ-2: The system must create a new account management window object to display it.*
3. *REQ-3: The system must create a new account management form window object to display it.*
4. *REQ-4: The System must process the user's request, validate and consult and add to the user database. Go to figure 3.1*
5. *REQ-5: The System verifies if the information is in the database and the account data is displayed.*
6. *REQ-6: The system should display a message that the change was successful.*
7. *REQ-7: The system must erase the memory belonging to the Account Management Window object once its use is finished.*
8. *REQ-8: The system must clear the memory belonging to the Window object after its use has finished.*
9. *REQ-9: The system must ensure that the database works correctly.*

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.>