**Final Report - KristMed**

* **Project III -**

Student : Efrem Ion Stefan

Facultatea de Automatica, Calculatoare si Electronica,

specializarea Calculatoare Engleza

Anul universitar 2020-2021

Contents

[1. System Architecture and Database Design 3](#_Toc61897698)

[2. UML Diagrams 10](#_Toc61897699)

[3. System Architecture and Database Structure 18](#_Toc61897700)

[4. Implemented Functionalities 20](#_Toc61897701)

[5. Plans to improve the Application 32](#_Toc61897702)

[6. Interface Design and Tools Used 32](#_Toc61897703)

# System Architecture and Database Design

The architecture of a system describes its structure and makes it understandable. There are different factors that influence the development of an architecture such as:

* functional requirements
* performance, reusability, scalability
* experience with existing architecture
* technical aspects

Architecture patterns are used to describe recurring design problems, which have been solved in the past and have helped develop certain design solutions which can be used for applications which have similar purposes (online shops, micro-transactions, etc…). We need to remind ourselves that patterns are just guidelines that need to be adapted for the required application.

For the Web Application “KristMed”, I have used the MVC and repository patterns.

The MVC pattern helped ease the development of the application as my experience with software engineering for web applications was limited. MVC gave me a lot more room to work with in developing and implementing the system requirements that were specified in the previous report. It also facilitates the ability to display large chunks of data on different pages.

I used the repository patterns in order to layer the access of my application and have the services access the repositories and not the data directly, thus having a smaller degree of dependency and facilitating future development of the application.

## Requirements

* + 1. Functional requirements:
       1. ***User Class 1 – The User***
          1. *Functional requirement 1.1*

Title: User registration

Description: In order to use the application, each new user must go through the registration process. In order to create a new account, the user must provide his first and last name, an username, an email, a phone number, his type of user, a photo and a password, etc..

Dependencies: None

* + - * 1. *Functional requirement 1.2*

Title: User Log-in

Description: After the user has successfully registered, the user must go through the log in process. In order to enter in his account, the user must provide username and his password.

Dependencies: FR1

* + - * 1. *Functional requirement 1.3*

Title: User changes password

Description: In case the user wants to change his password, he must go through the change password process. In order to change the password, the user must provide his old password, his new password twice as to confirm he has written it correctly.

Dependencies: FR2

* + - * 1. *Functional requirement 1.4*

Title: User adds contact message

Description: If the user wants to send a contact message with tips or complains for the clinic they will be able to do that from the “Add Contact Message Tab”.

Dependencies: FR2

* + - * 1. *Functional requirement 1.5*

Title: User sees contact messages

Description: If the user wants to see his own sent contact messages he will be able to do so by clicking the “Contact Messages” after logging in

Dependencies: FR2

* + - * 1. *Functional requirement 1.6*

Title: User display all appointments

Description: The user can display all appointments and also filter through them by medics

Dependencies: FR2

* + - * 1. *Functional requirement 1.7*

Title: User can buy a ticket

Description: The client user can buy tickets. In order to buy a ticket, the user must provide the details needed. He can also buy tickets for other persons, as long as he provides the required information.

Dependencies: FR2

* + - * 1. *Functional requirement 1.8*

Title: User display past appointments

Description: The user can display past appointments

Dependencies: FR2

* + - * 1. *Functional requirement 1.9*

Title: User display future appointments

Description: The user can display future appointments

Dependencies: FR2

* + - * 1. *Functional requirement 1.10*

Title: User can delete appointment

Description: The user can delete his future appointments

Dependencies: FR2

* + - * 1. *Functional requirement 1.11*

Title: User can log out.

Description: The user can log out of his account.

Dependencies: FR2

* + - * 1. *Functional requirement 1.12*

Title: User can delete account

Description: The user can delete his account. Dependencies: FR2

* + - * 1. *Functional requirement 1.13*

Title: User can edit account

Description: The user can edit his account. Dependencies: FR2

* + - 1. ***User Class 2 – The Administrator***
         1. *Functional requirement 2.1*

Title: User registration

Description: In order to use the application, each new user must go through the registration process. In order to create a new account, the user must provide his full name, a username, a password, etc… as well as his type of user.

Dependencies: None

* + - * 1. *Functional requirement 2.2*

Title: User Log-in

Description: Given that the user has registered, in order to use the application, each user must go through the log in process. In order to enter in account, the user must provide username and his password.

Dependencies: FR1

* + - * 1. *Functional requirement 2.3*

Title: User changes password

Description: In case the user wants to change his password, he must go through the change password process. In order to change the password, the user must provide his old password and his new password.

Dependencies: FR2

* + - * 1. *Functional requirement 2.4*

Title: User adds medication

Description: If the user wants to add a medication type they will be able to do that from the “Add Medicine” Tab.

Dependencies: FR2

* + - * 1. *Functional requirement 2.5*

Title: User sees medications

Description: If the user wants to see the available medicine he will be able to do so by clicking the “Medicine” after logging in

Dependencies: FR2

* + - * 1. *Functional requirement 2.6*

Title: User adds equipment

Description: If the user wants to add an equipment type they will be able to do that from the “Add Equipment” Tab.

Dependencies: FR2

* + - * 1. *Functional requirement 2.7*

Title: User sees equipment

Description: If the user wants to see the available equipment he will be able to do so by clicking the “Equipment” after logging in

Dependencies: FR2

* + - * 1. *Functional requirement 2.8*

Title: User displays messages

Description: The user can display all the messages sent by clients

Dependencies: FR2

* + - * 1. *Functional requirement 2.9*

Title: User sees appointments

Description: The user can display all appointments

Dependencies: FR2

* + - * 1. *Functional requirement 2.10*

Title: User adds appointments

Description: The user can add appointments

Dependencies: FR2

* + - * 1. *Functional requirement 2.11*

Title: User can log out.

Description: The user can log out of his account.

Dependencies: FR2

* + - * 1. *Functional requirement 2.12*

Title: User can delete account

Description: The user can delete his account. Dependencies: FR2

* + - * 1. *Functional requirement 2.13*

Title: User can edit account

Description: The user can edit his account. Dependencies: FR2

* + - 1. ***User Class 3 – The Medic***
         1. *Functional requirement 3.1*

Title: User registration

Description: In order to use the application, each new user must go through the registration process. In order to create a new account, the user must provide his full name, a username, a password, etc… as well as his type of user.

Dependencies: None

* + - * 1. *Functional requirement 3.2*

Title: User Log-in

Description: Given that the user has registered, in order to use the application, each user must go through the log in process. In order to enter in account, the user must provide username and his password.

Dependencies: FR1

* + - * 1. *Functional requirement 3.3*

Title: User changes password

Description: In case the user wants to change his password, he must go through the change password process. In order to change the password, the user must provide his old password and his new password.

Dependencies: FR2

* + - * 1. *Functional requirement 3.4*

Title: User display past appointments

Description: The user can display past appointments

Dependencies: FR2

* + - * 1. *Functional requirement 3.5*

Title: User display future appointments

Description: The user can display future appointments

Dependencies: FR2

* + - * 1. *Functional requirement 3.6*

Title: User adds treatments

Description: The user can add treatments to past appointments with clients

Dependencies: FR2

* + - * 1. *Functional requirement 3.7*

Title: User displays treatments

Description: The user can display all treatments

Dependencies: FR2

* + - * 1. *Functional requirement 3.8*

Title: User can log out.

Description: The user can log out of his account.

Dependencies: FR2

* + - * 1. *Functional requirement 3.9*

Title: User can delete account

Description: The user can delete his account. Dependencies: FR2

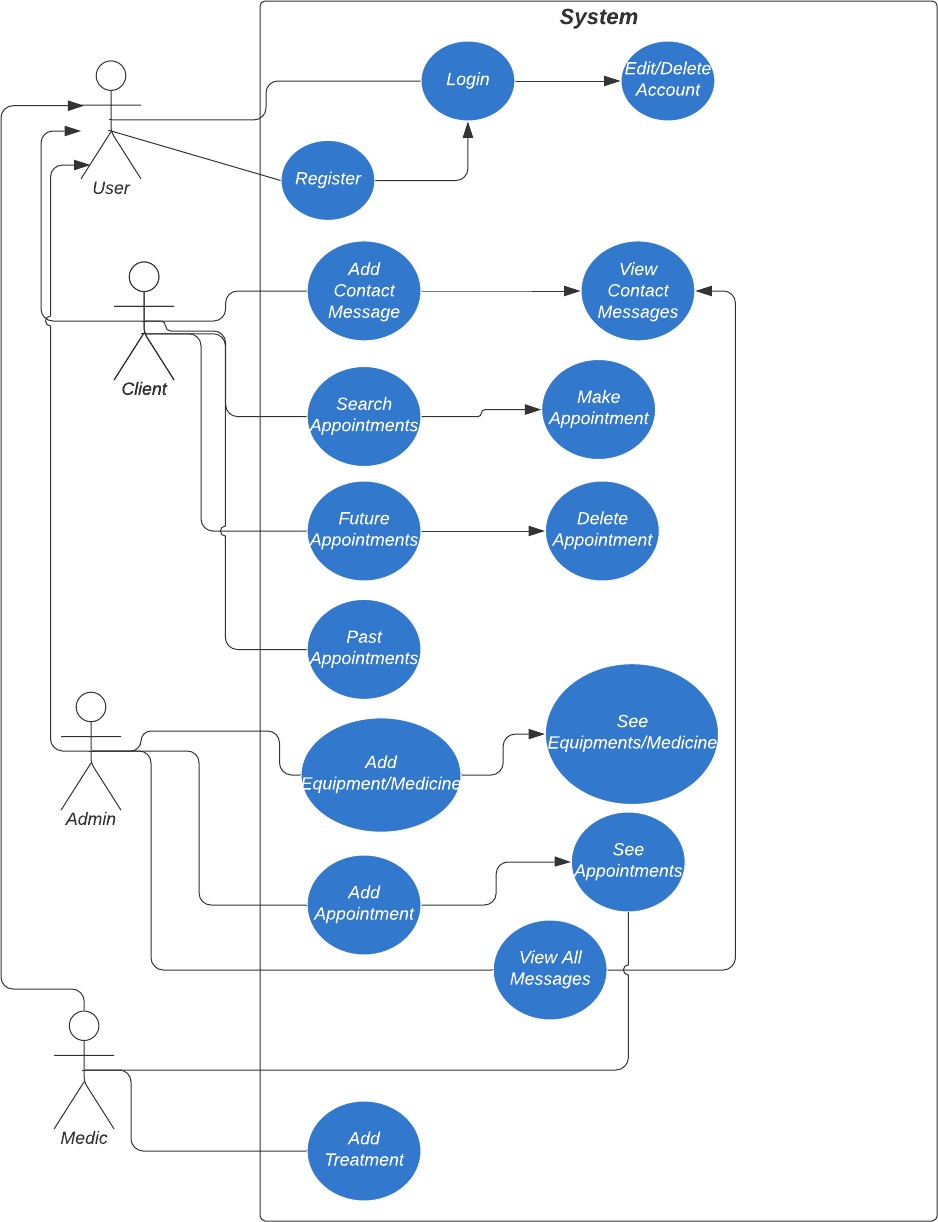
* + - * 1. *Functional requirement 3.10*

Title: User can edit account

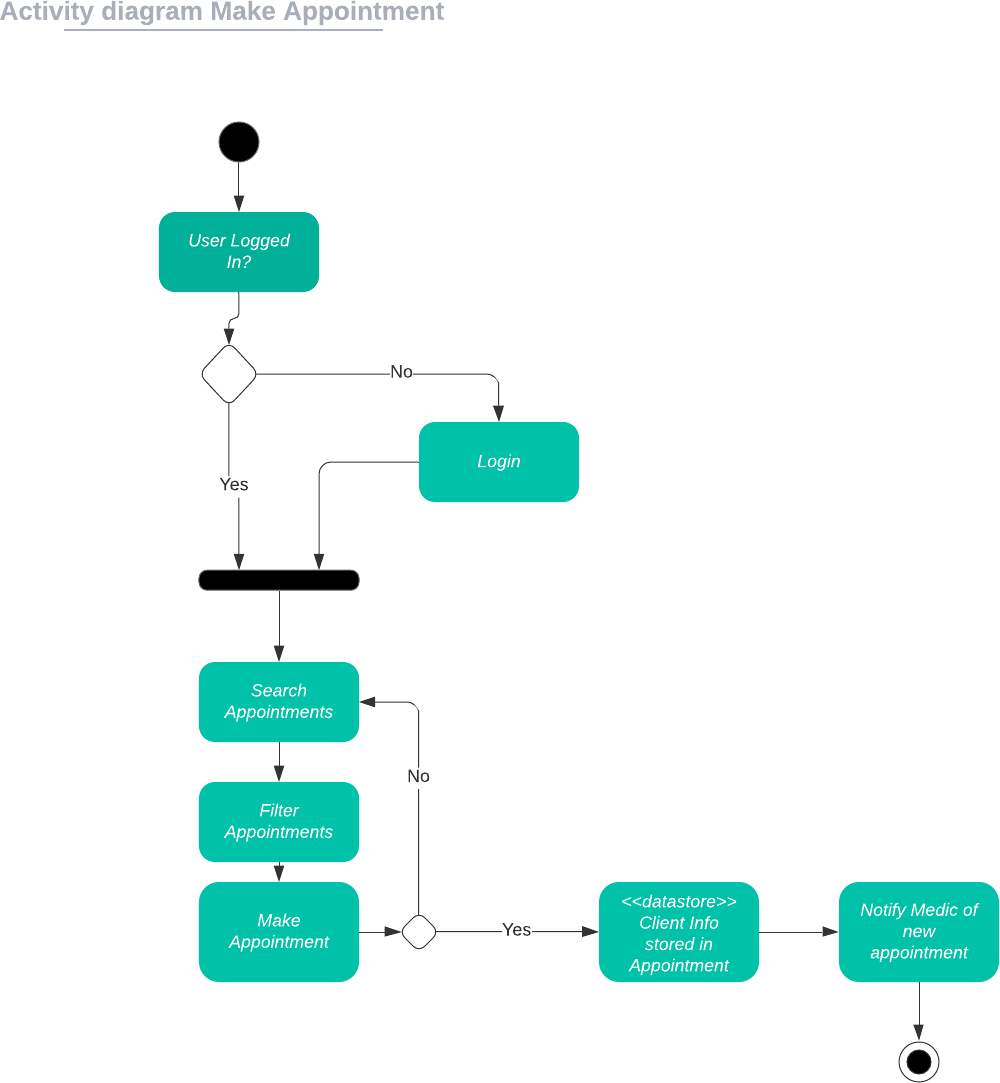
Description: The user can edit his account. Dependencies: FR2

# UML Diagrams

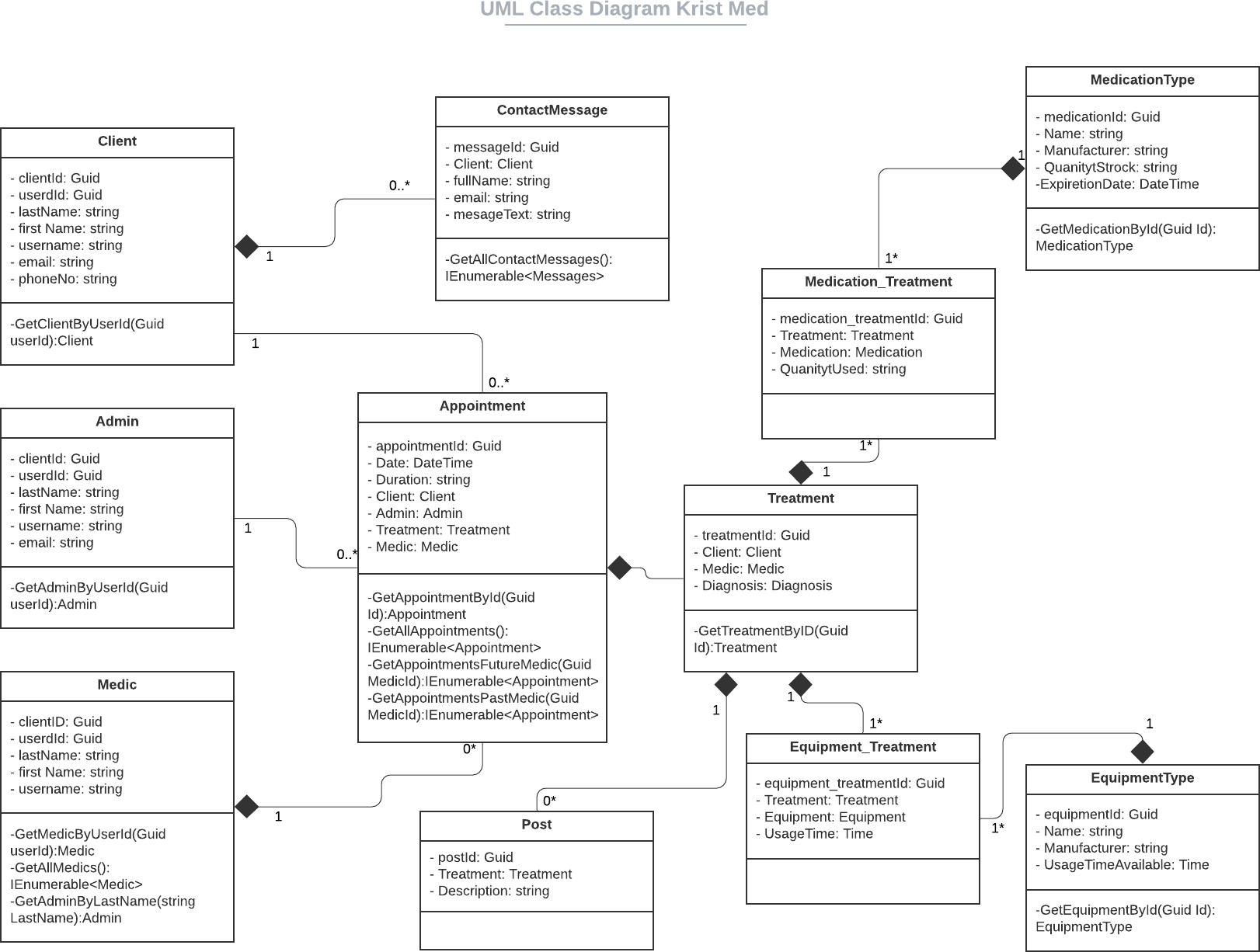
* Use case diagram



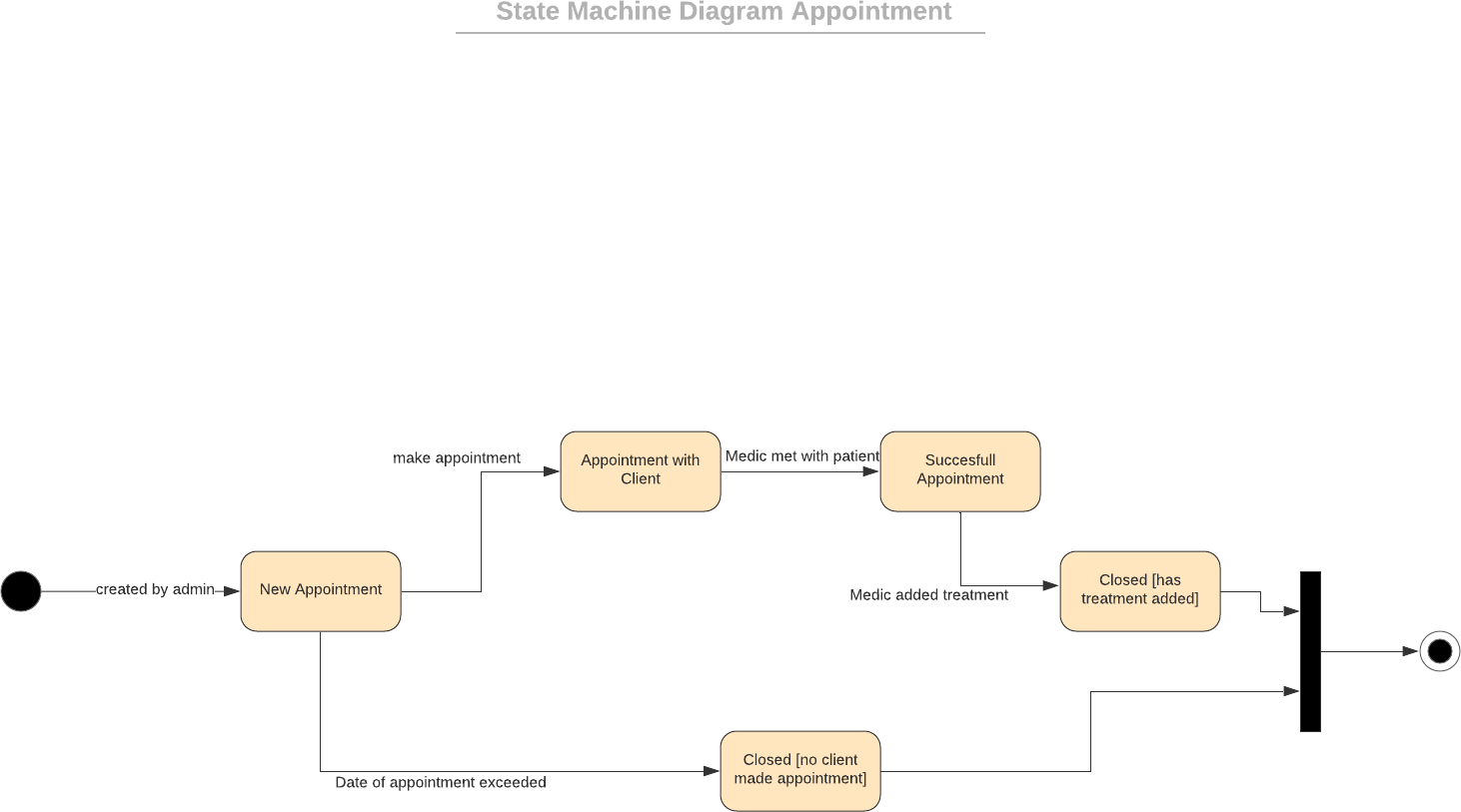
* Activity diagram



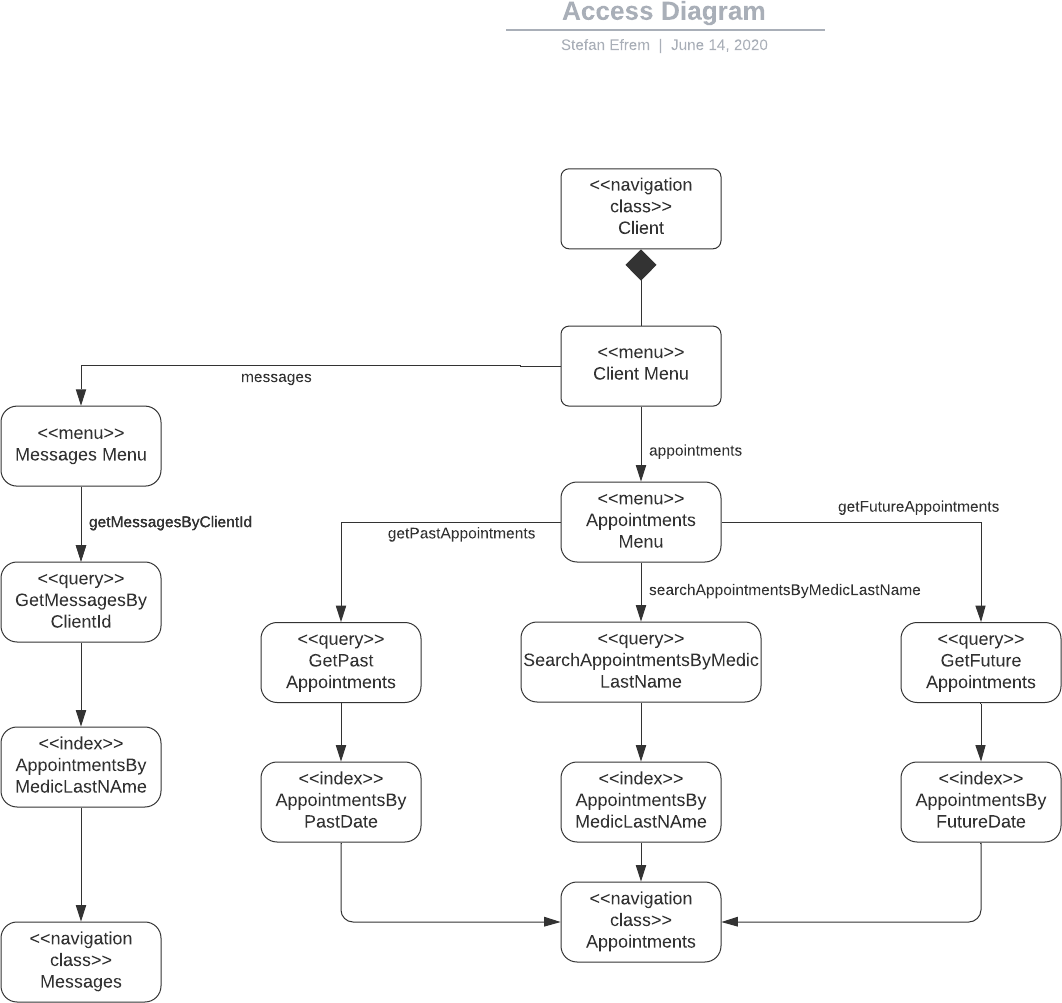
* Class diagram



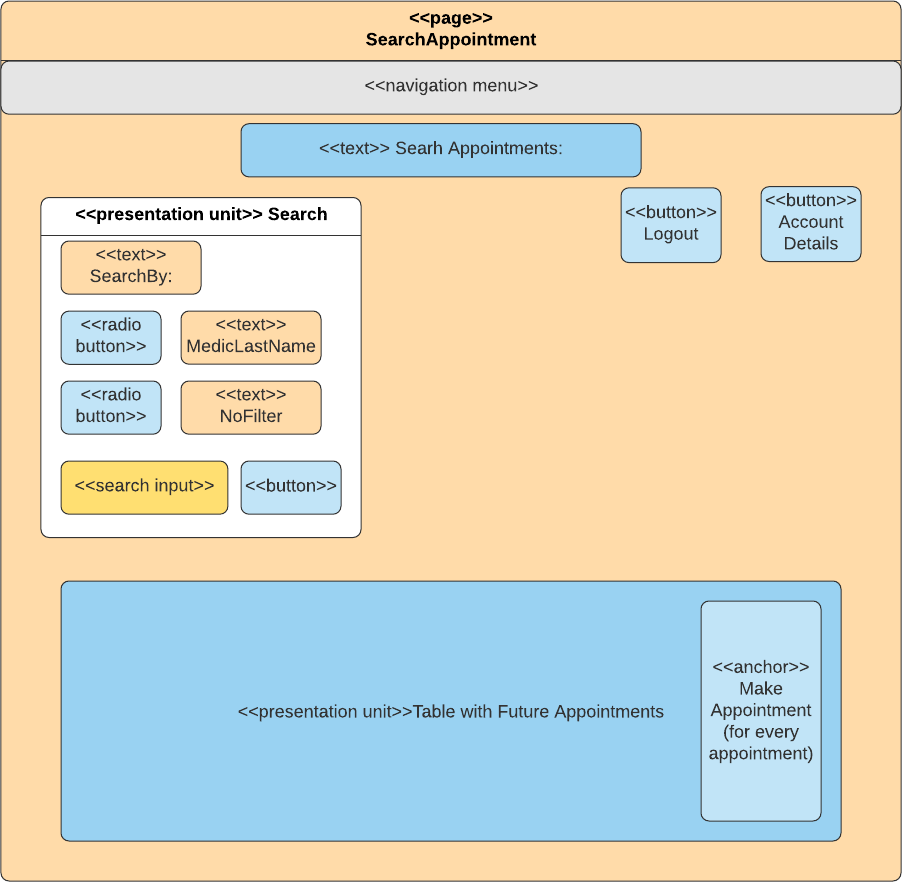
* State machine diagram



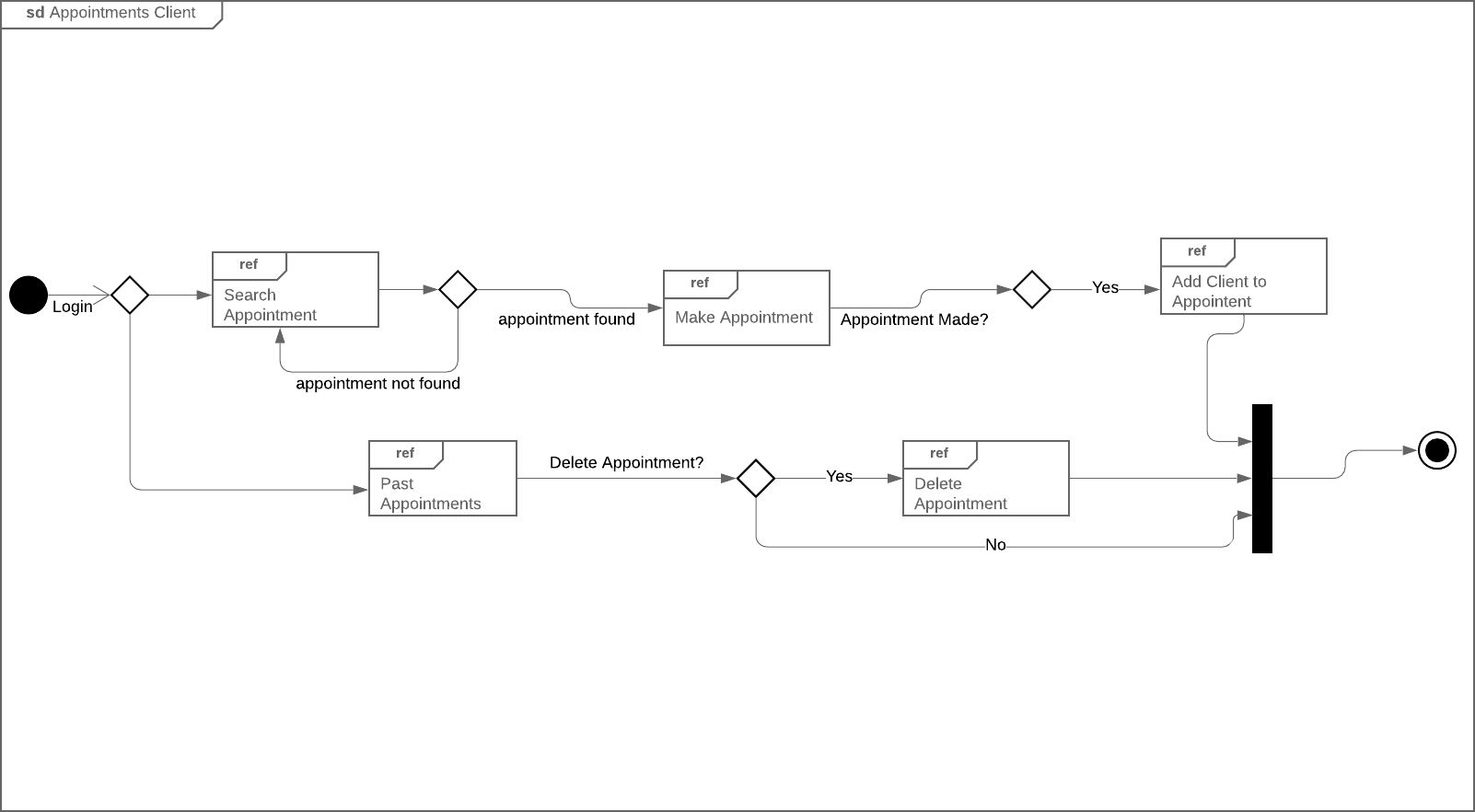
* Access model



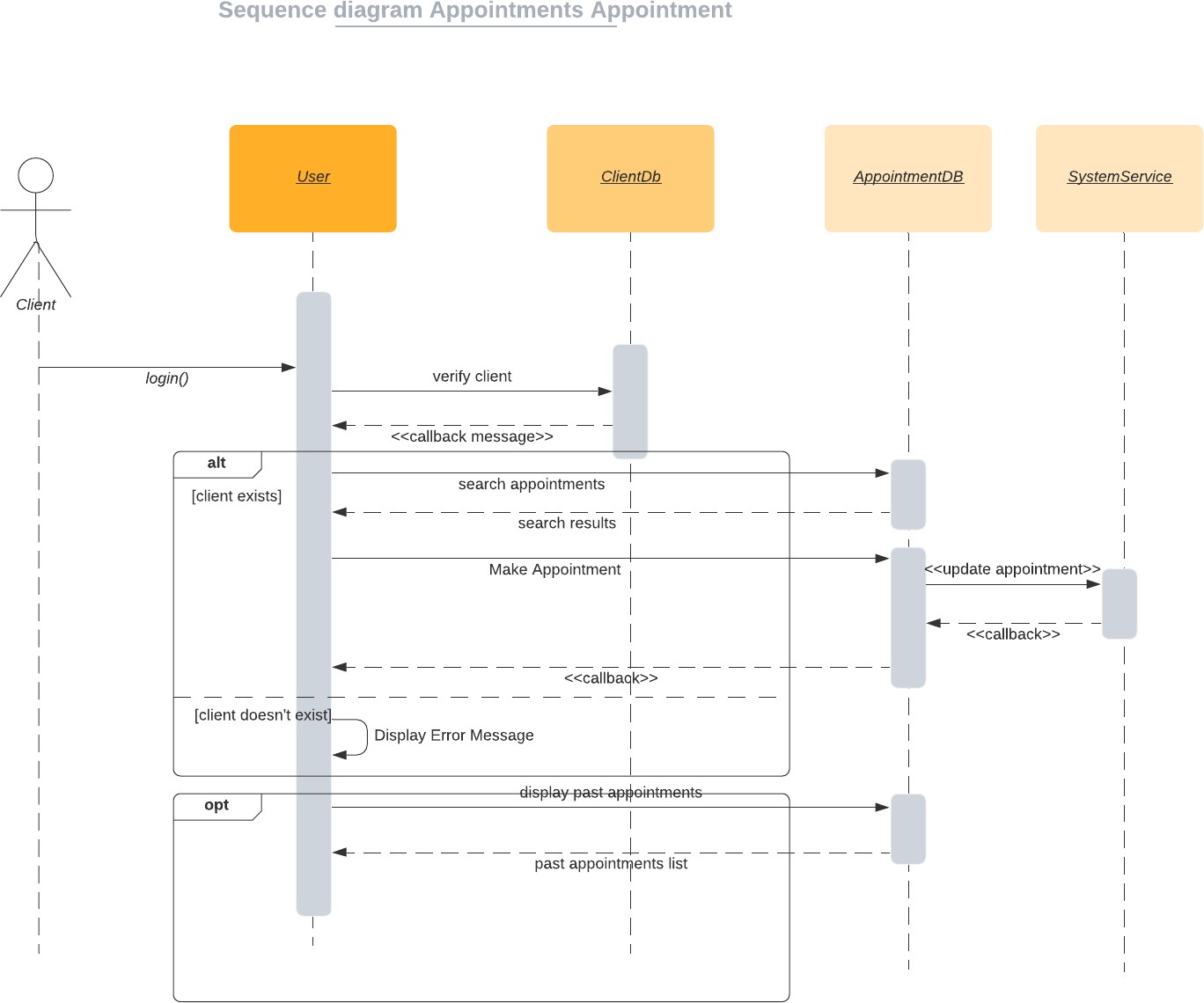
* Presentation page



* Interaction overview diagram



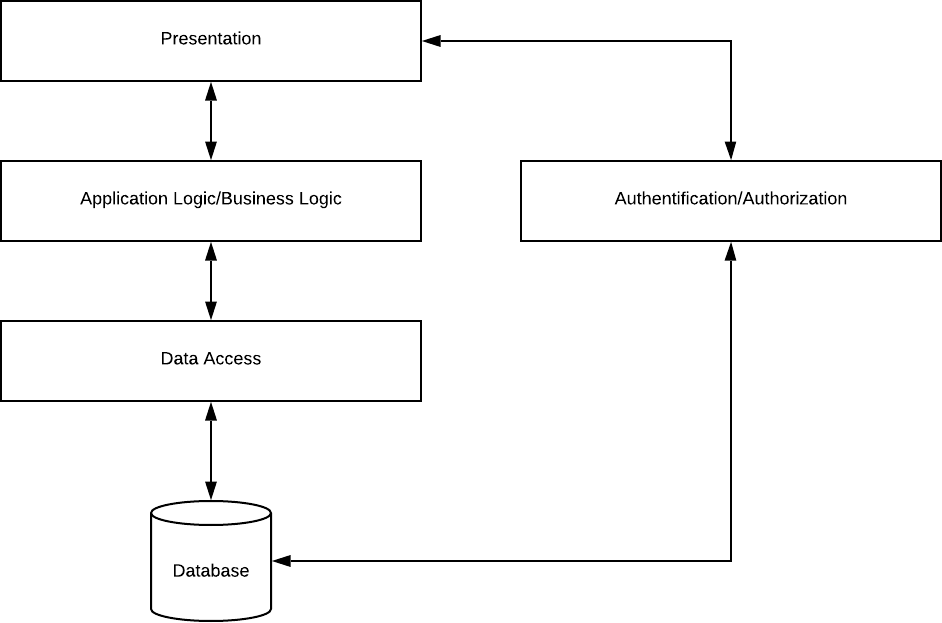
* Sequence Diagram



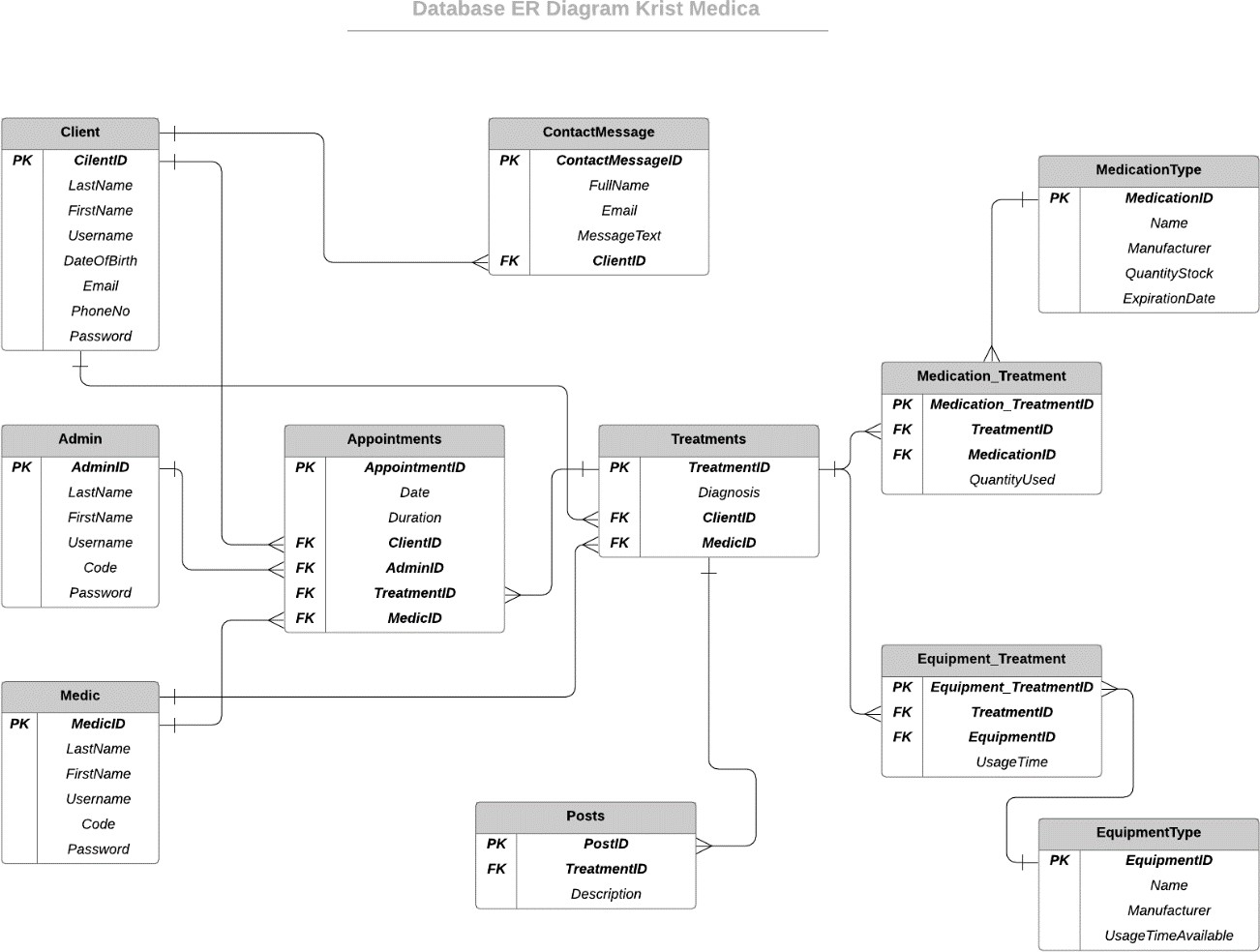
# System Architecture and Database Structure

My application uses a 5-layer architecture:

* The Presentation layer which is represented by the web-server
* The Business layer which includes the application logic with the services and models
* The Data Access layer which includes the repositories
* The Database of the application
* The Authentication - Authorization layer



The Database structure is presented in the diagram below. The Access to the database is made through repositories which update/manage the data.

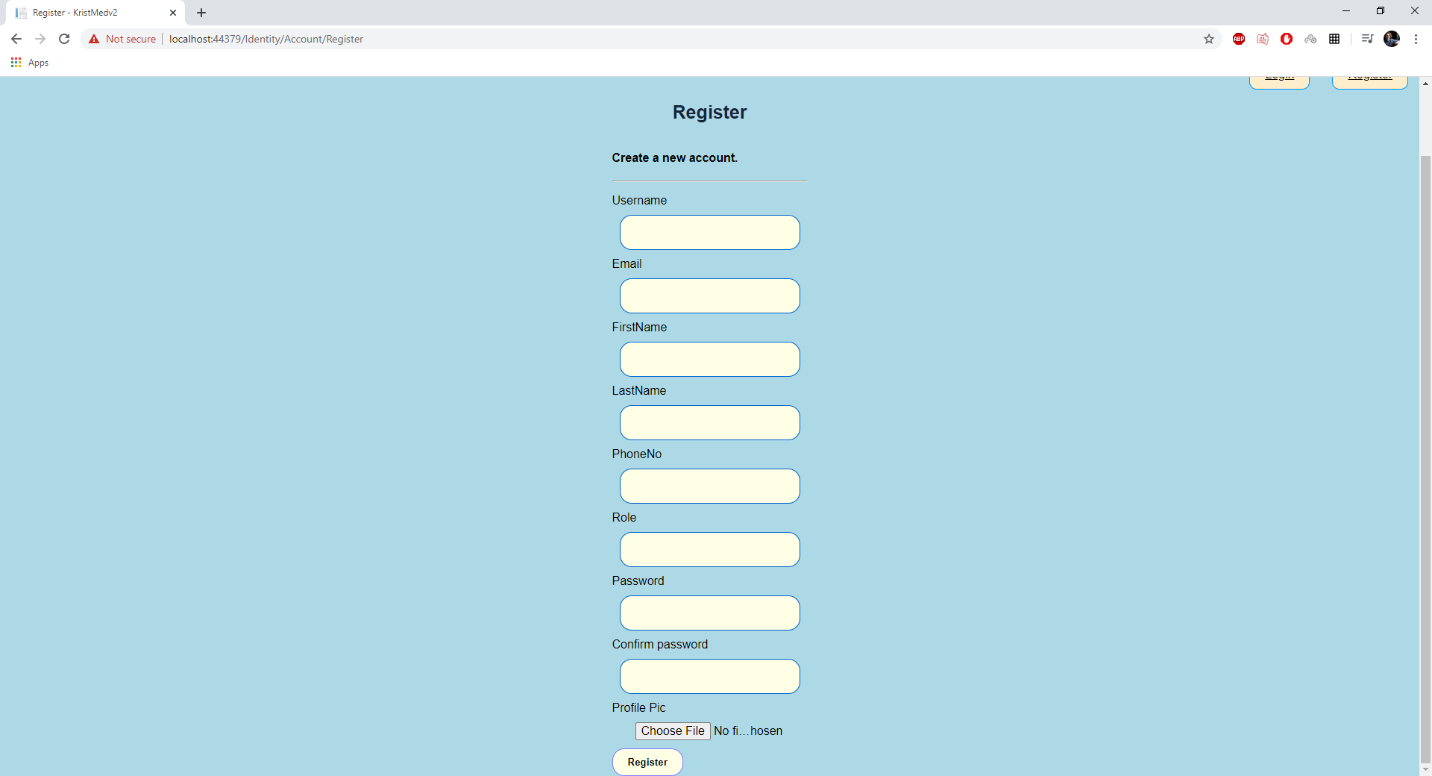


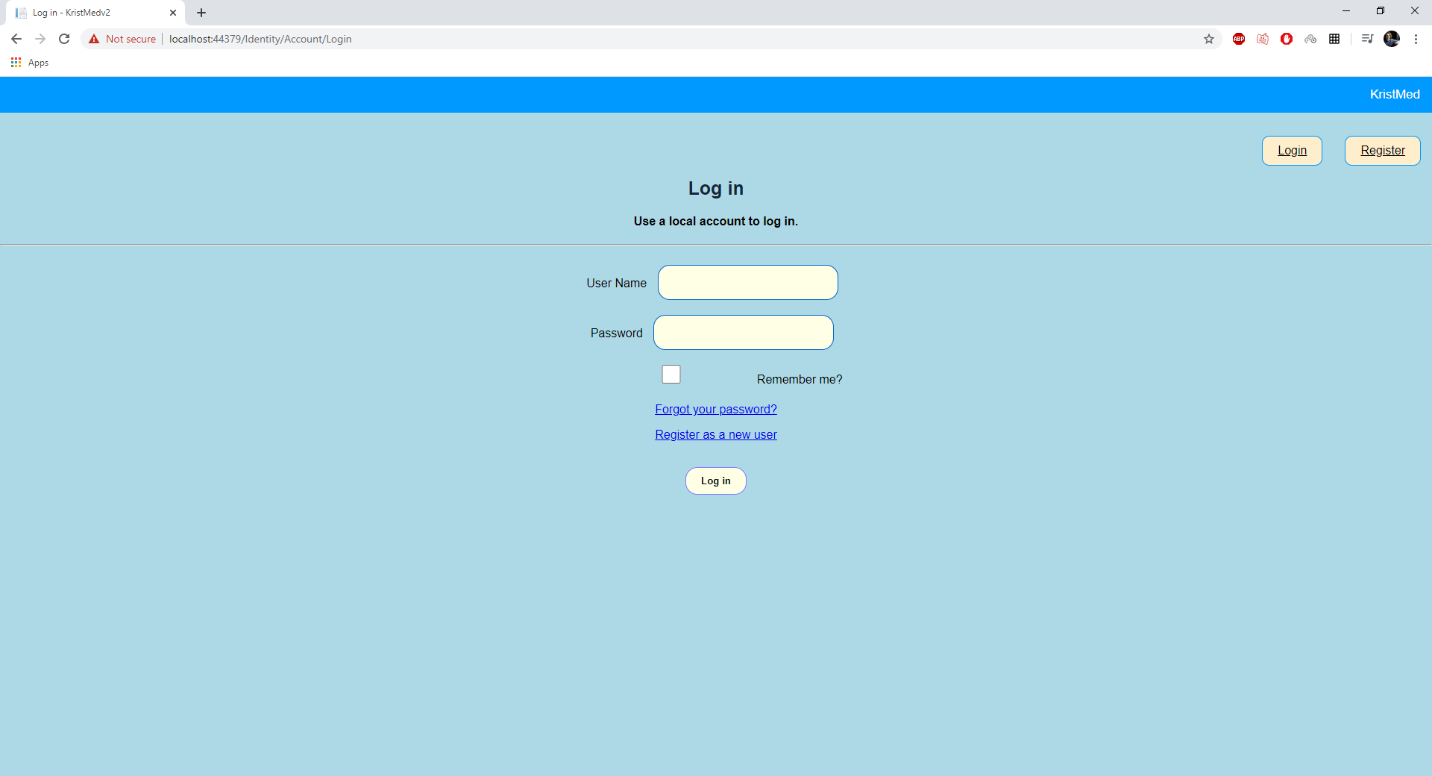
# Implemented Functionalities

I will separate the functionalities implemented in 4 parts:

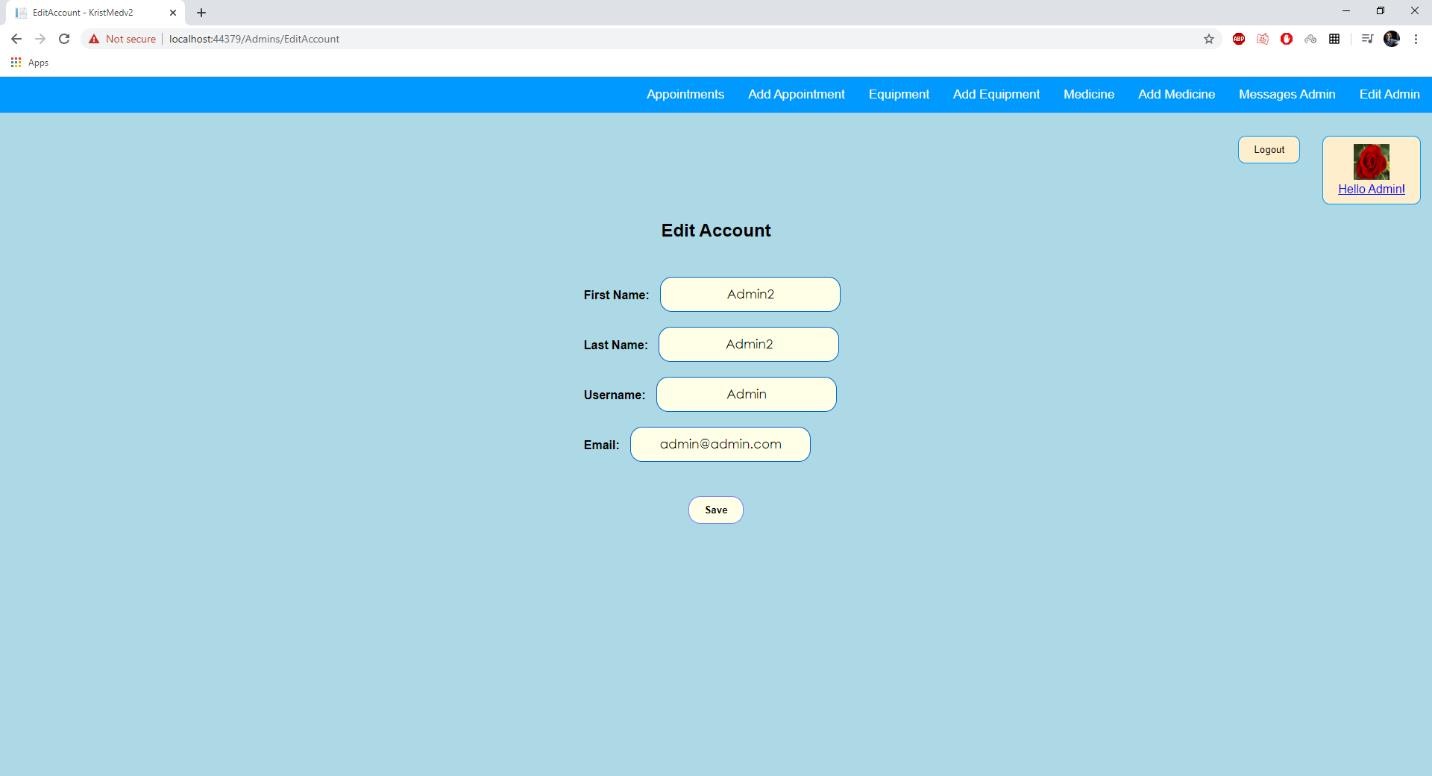
* 1. *Common functionalities (user authentication/authorization)*

The common functionalities of the application are represented by the login/register functionalities. The application uses ASP Identity for its user database and it has designated layouts and authorization for the 3 different user roles: client, admin, medic.

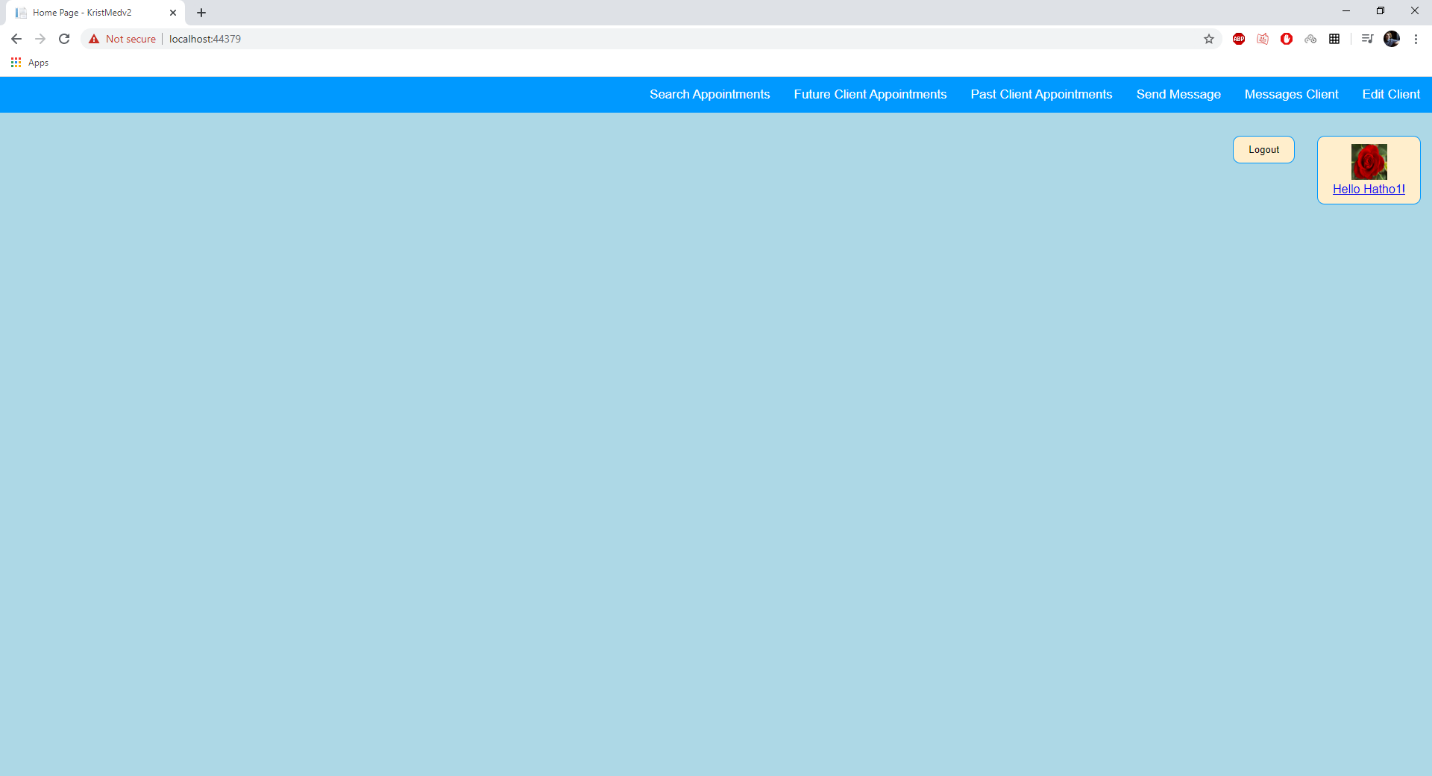


On the Register Page we need to add the required information to create our user. The user will have some of his information stored in the Identity Database (username, email password, photo path) and the rest of the information ( first name, last name, phone No, etc..) will be stored in the application’s database. The Profile picture is added directly to the identity user. The picture is copied and put in the application’s files while also adding a GUID to the name of the picture in order to uniquely identify all the pictures.

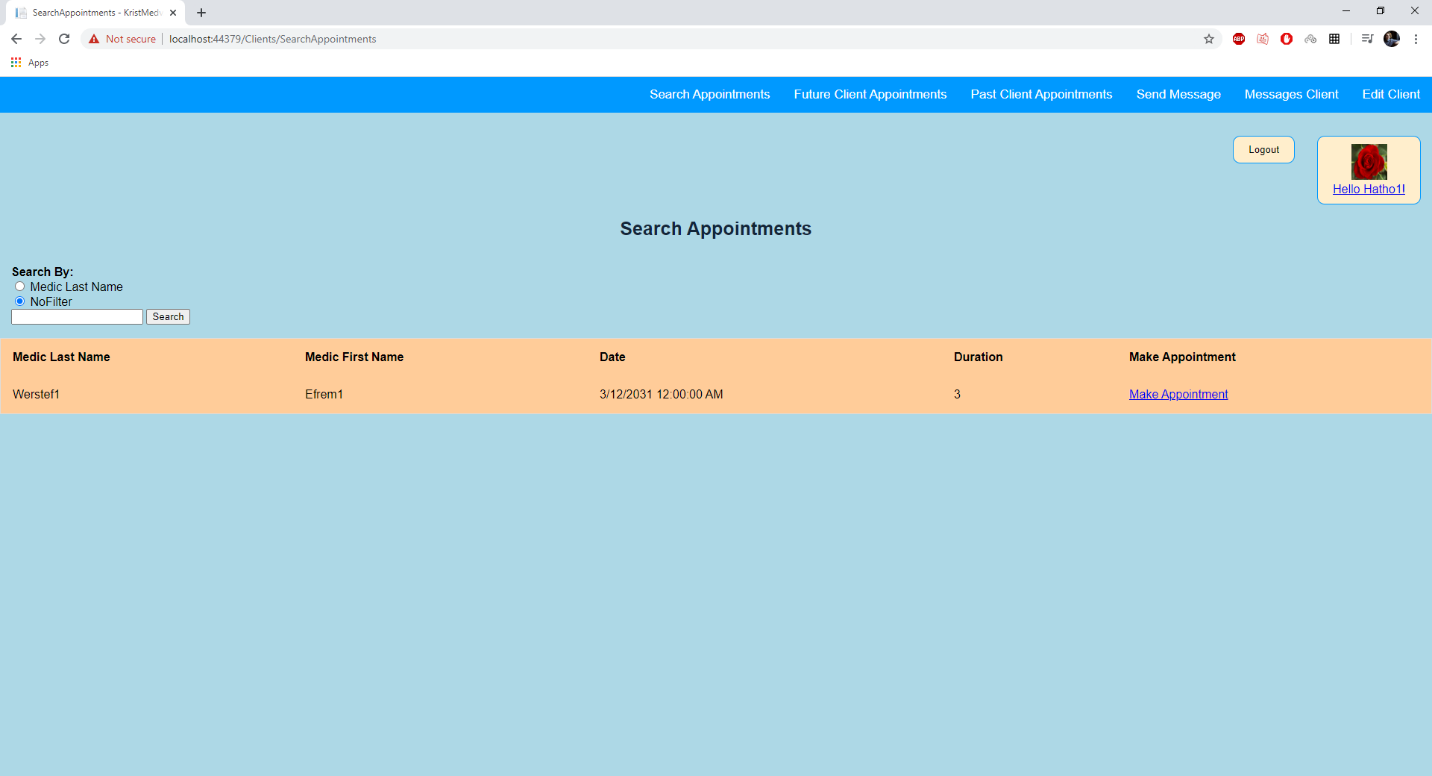
The Login page is a simple login page where the user has to login, depending on the type of the user, the interface will be different after login.



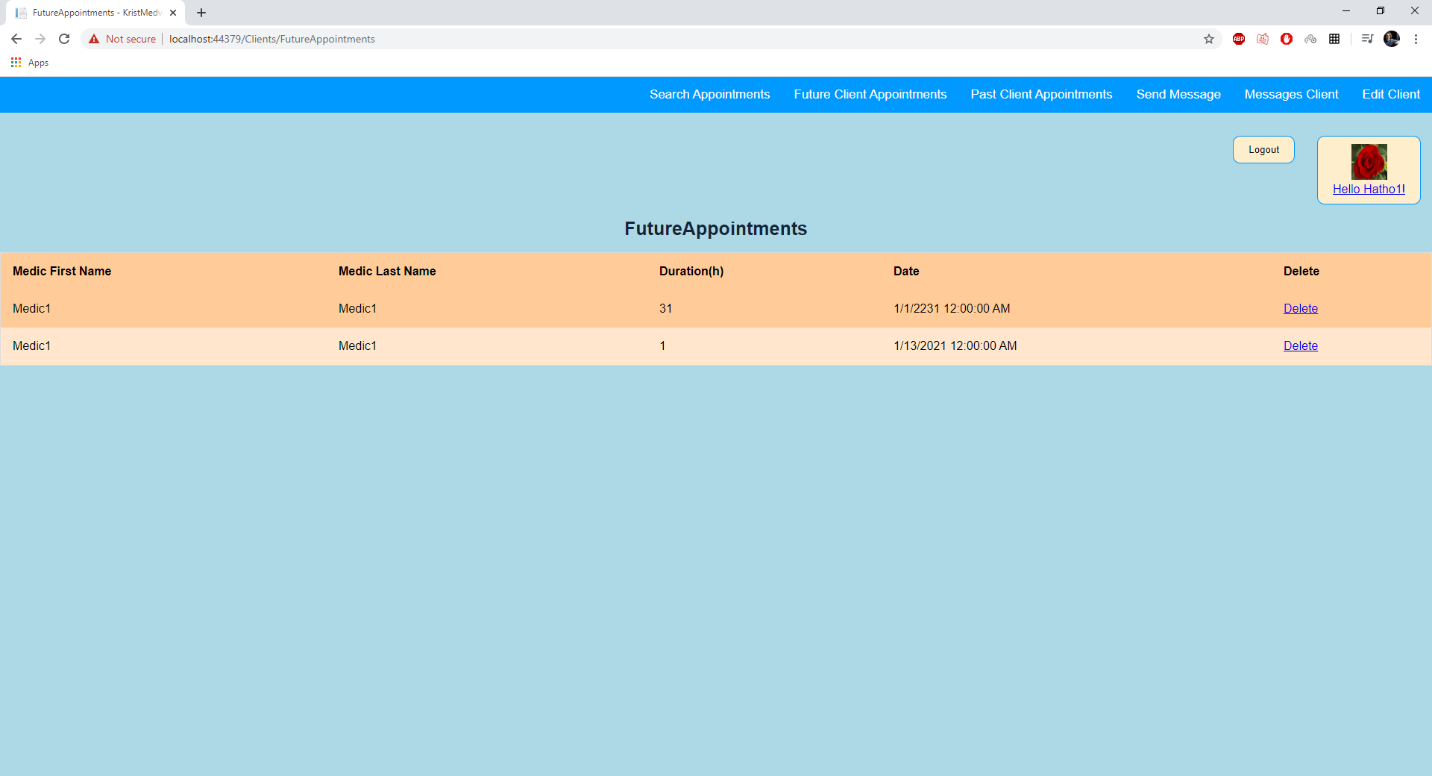
The Edit Account Page is also common to all the users and it allows them to modify their accounts. It has their current information as placeholders in order to help the visibility.

* 1. *Client functionalities*

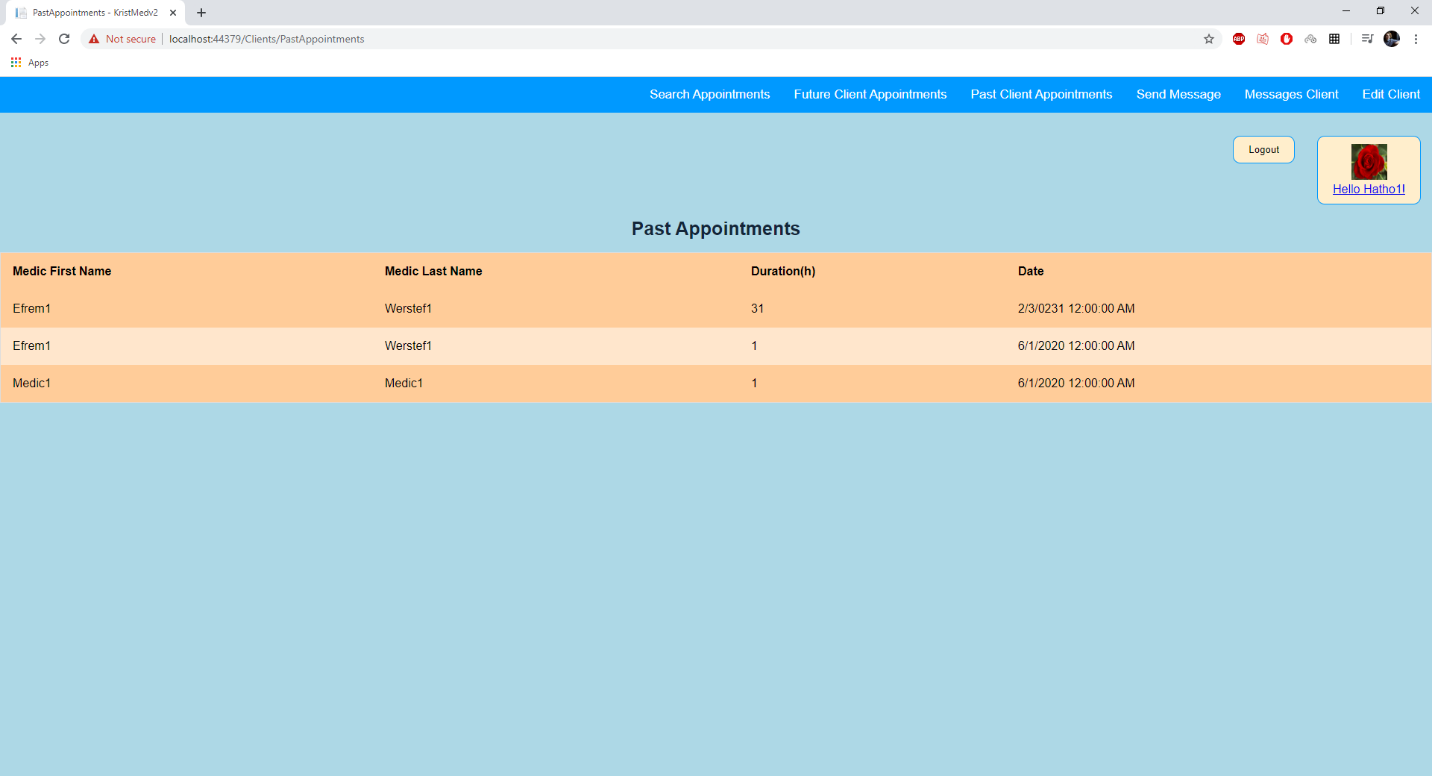
The Client Home Page has the menu with the pages that are available for the client.



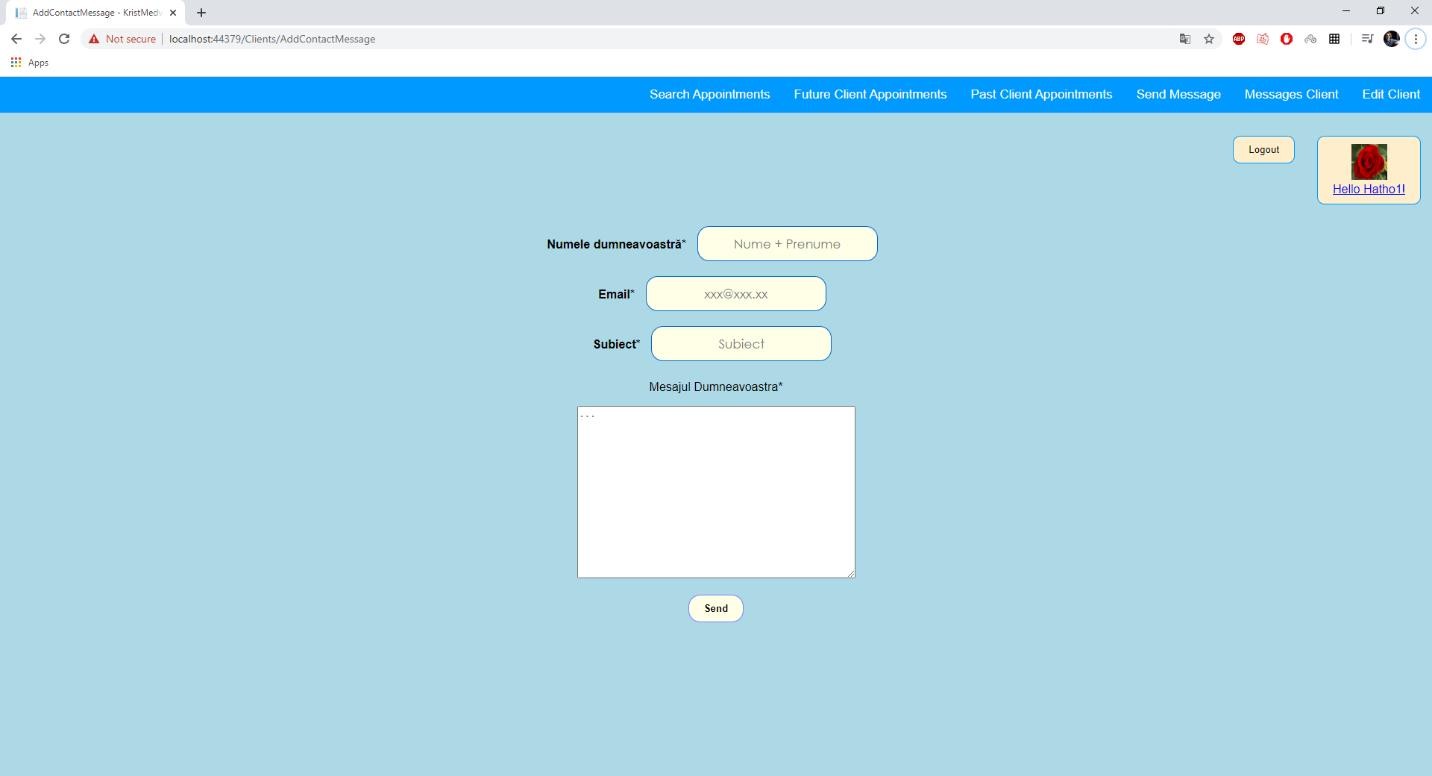
The “Search Appointments” page implements the search functionality, it allows us to search for all available appointments or just the ones of certain medic by writing his Last Name and choosing the corresponding filter. On the right of every available appointment we can press the “Make Appointment” button to make an appointment with that Medic at the specified Date.



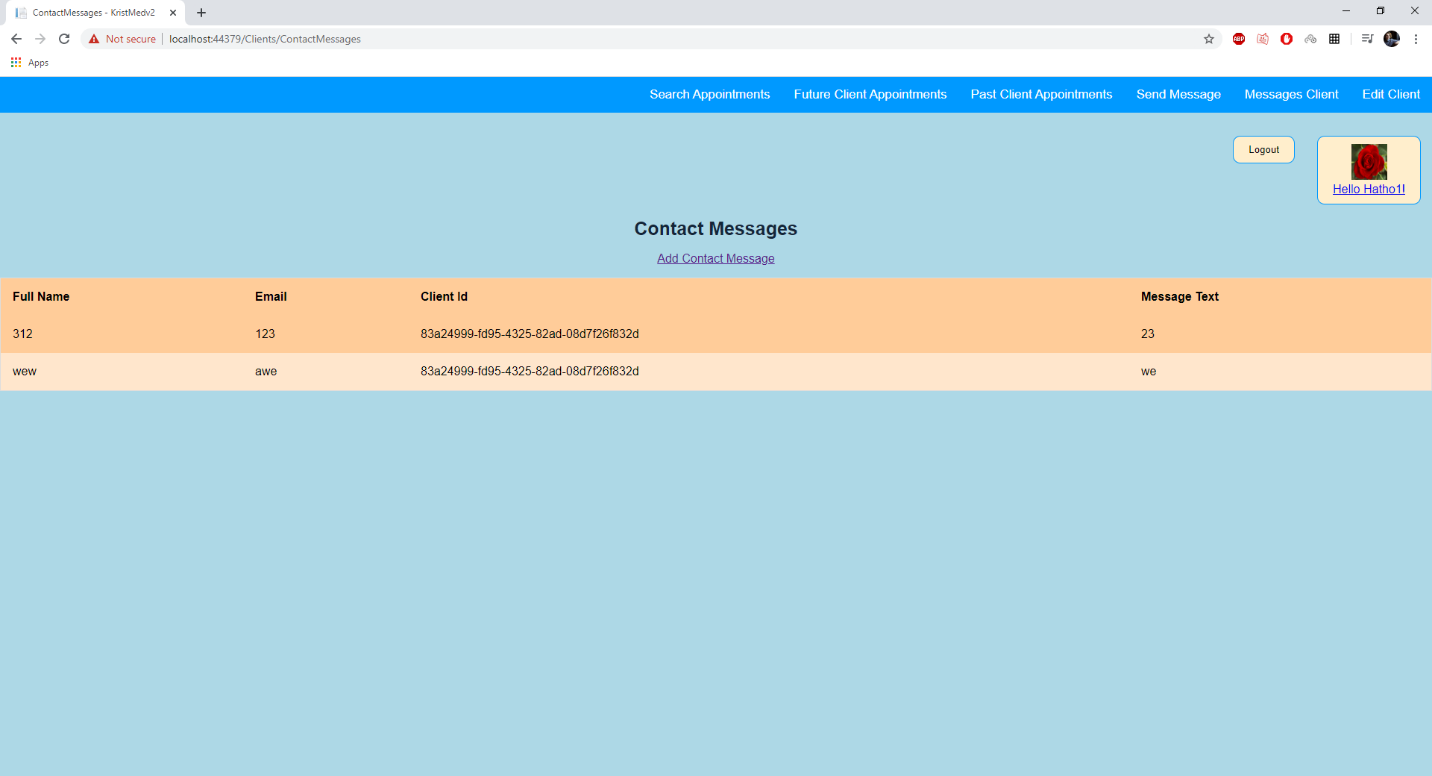
On the “Future Appointments” page we can see the client’s future appointments and an option to delete this appointment in case we want to cancel our meeting with the medic.



In the “Past Appointments” page we can see the past appointments of the client and the Duration of every appointment.

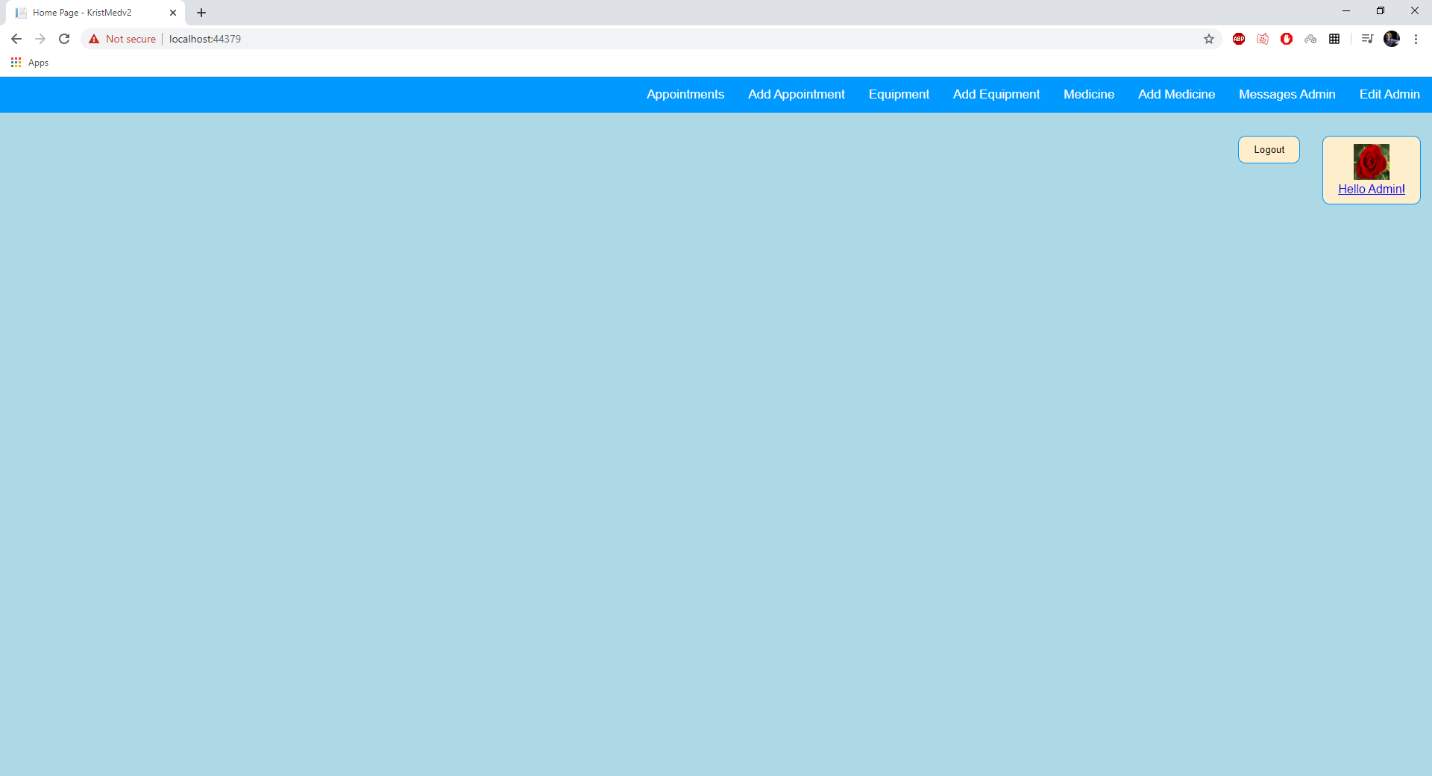


On the “Send Message” page we can send a message to the clinic, this is done in case we want some additional information about treatment or we want to ask some questions. The clinic should respond relatively fast.

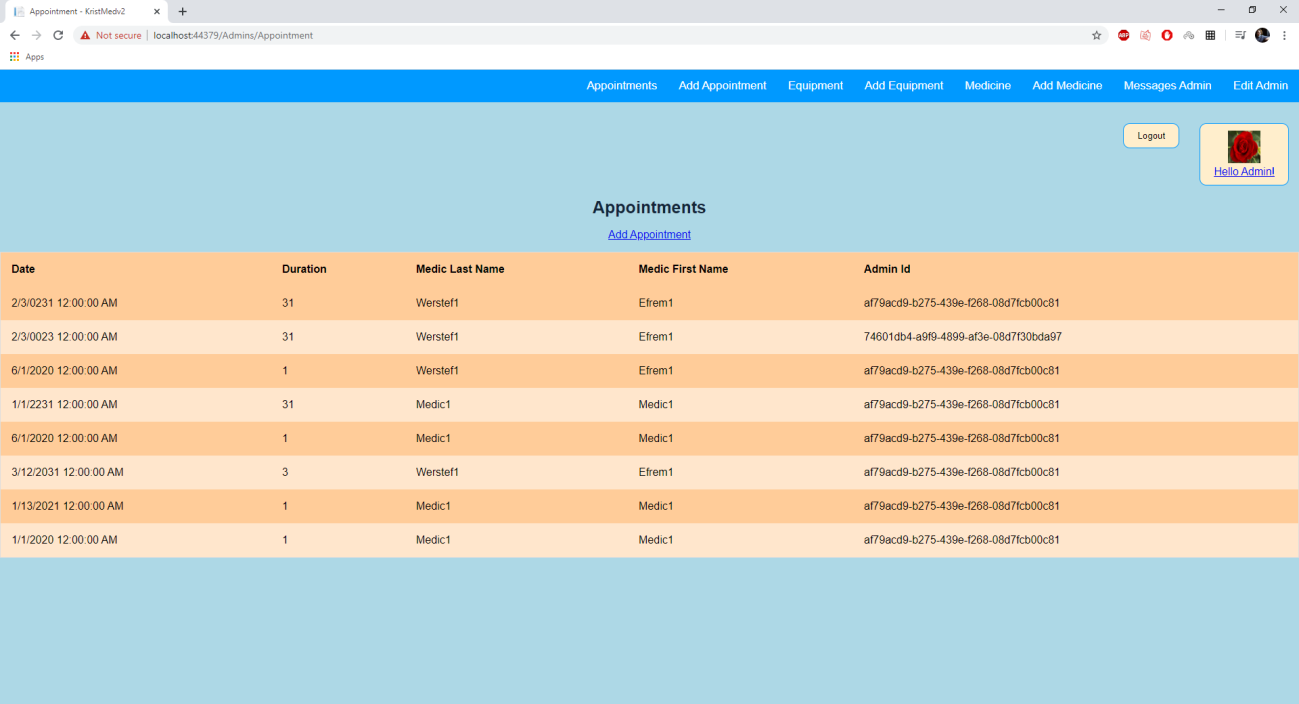


On the “Message Client” page we can see the client’s messages that have already been sent to the clinic.

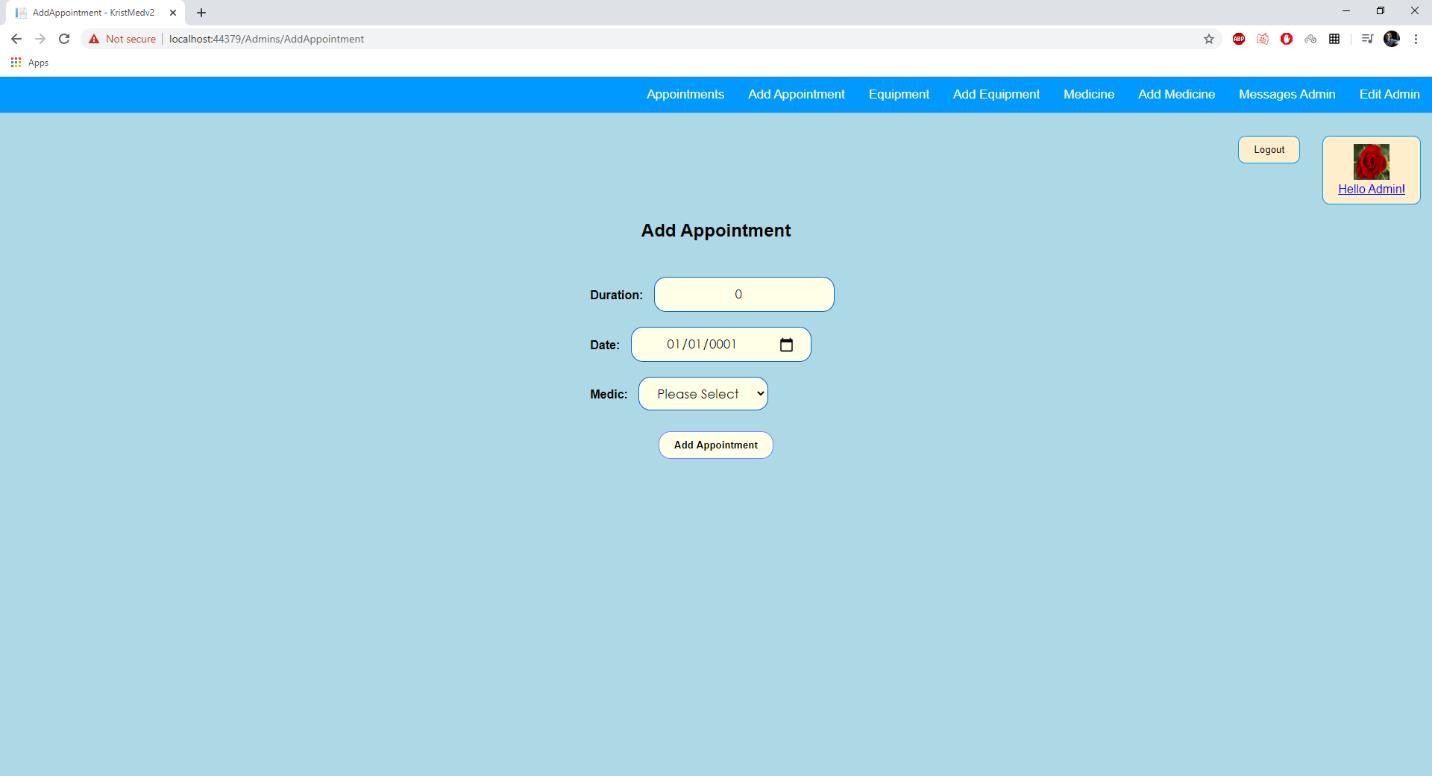
* 1. *Admin functionalities*



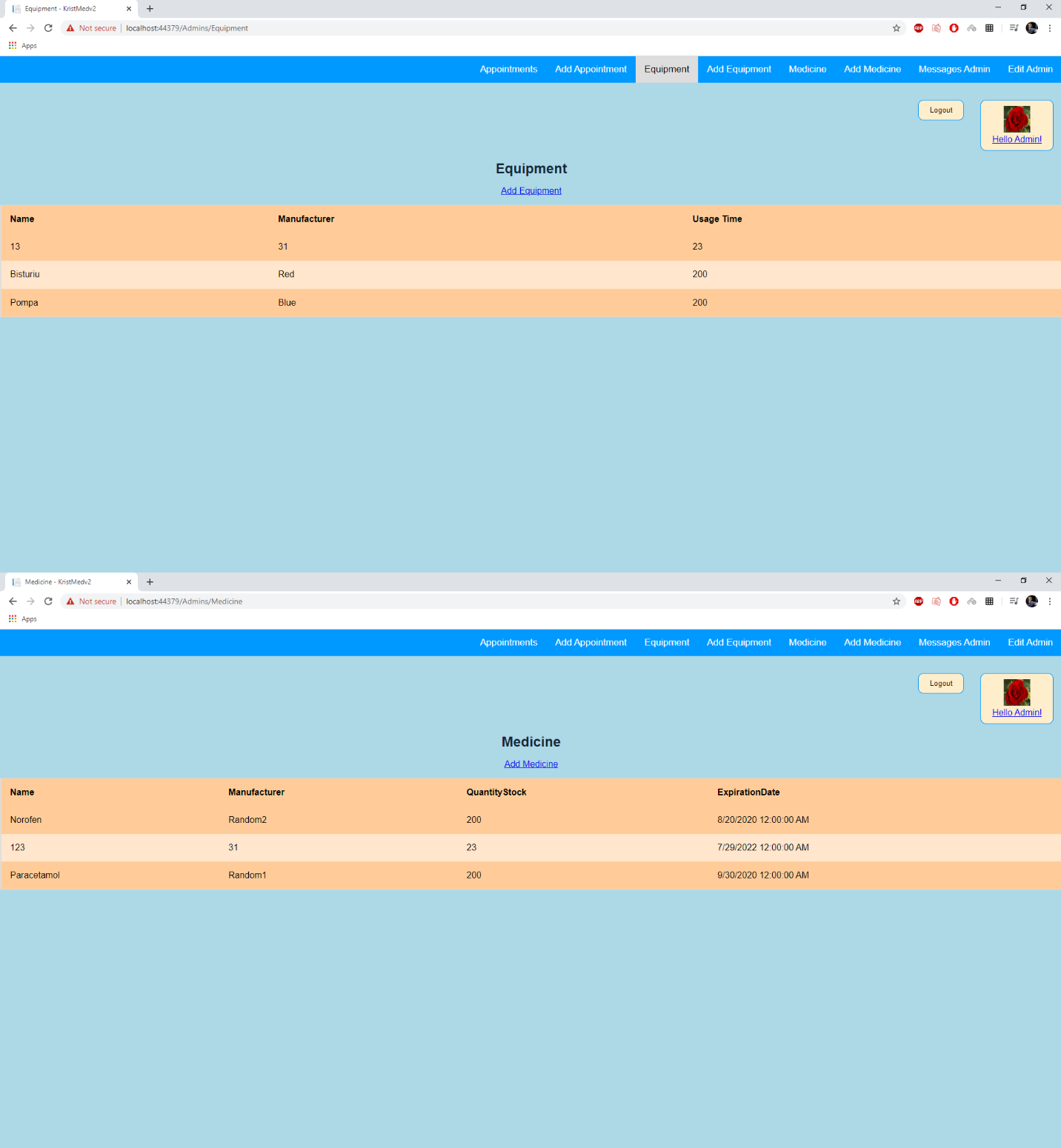
The Admin Home Page has the menu with the pages that are available for the admin.



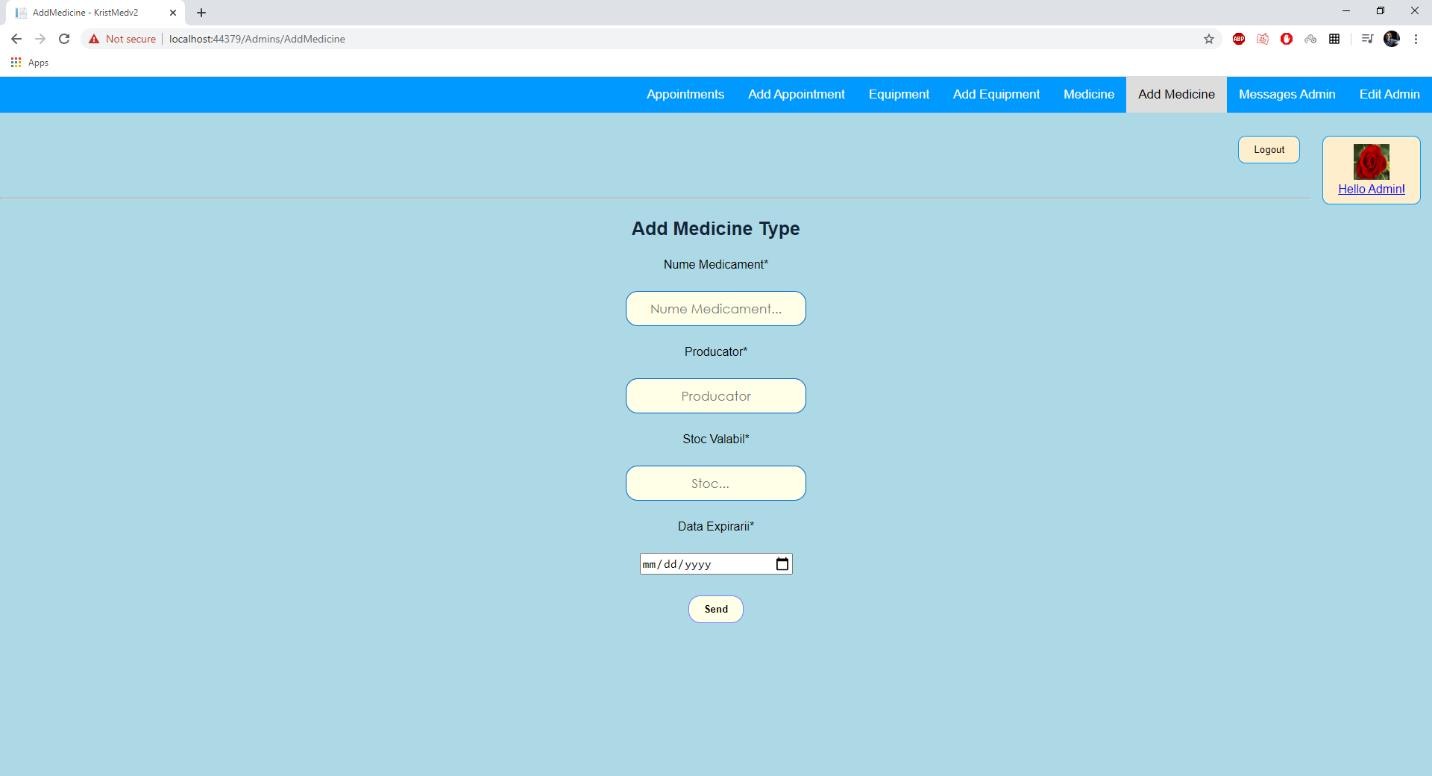
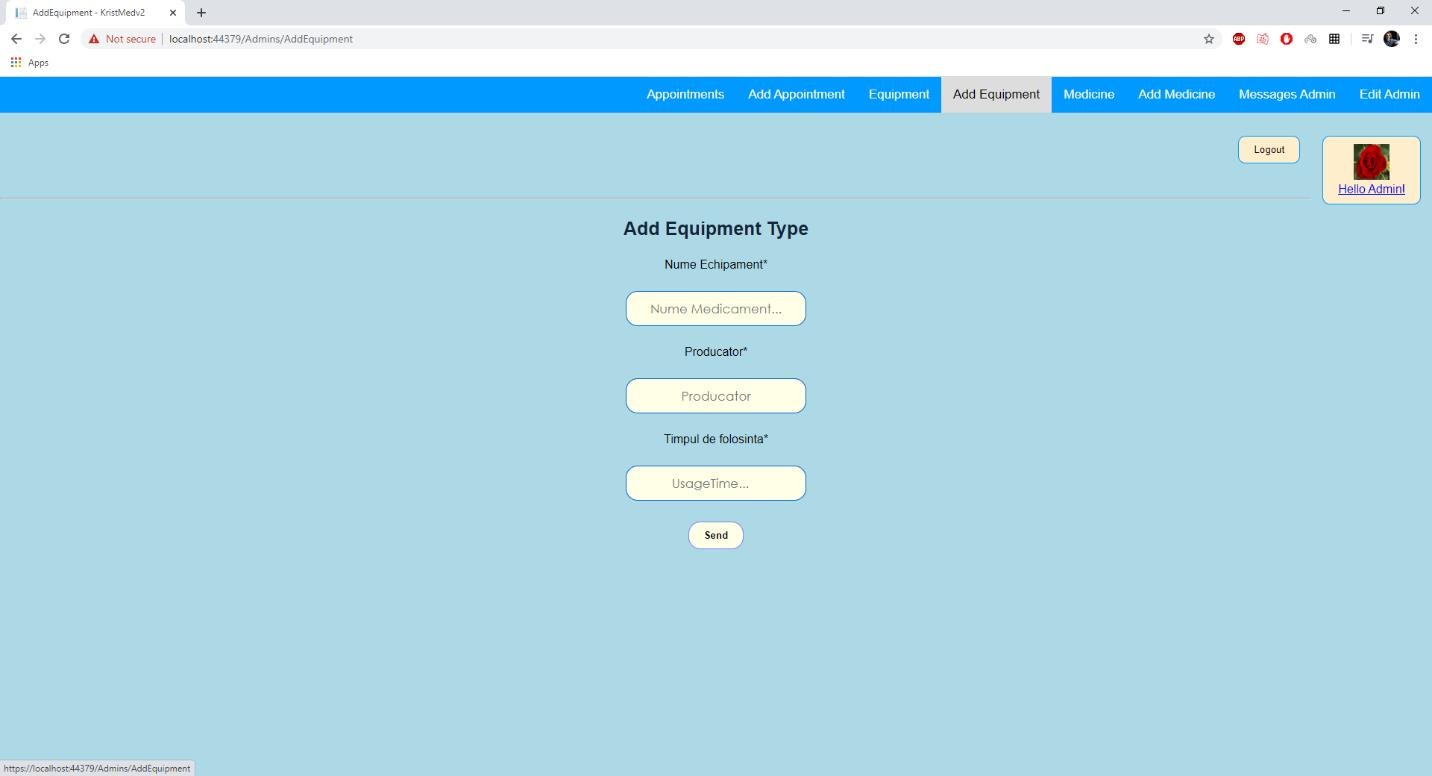
On the “Appointments” page he can see a list of all the appointments created and a button which sends him to a page where he can add more appointments.



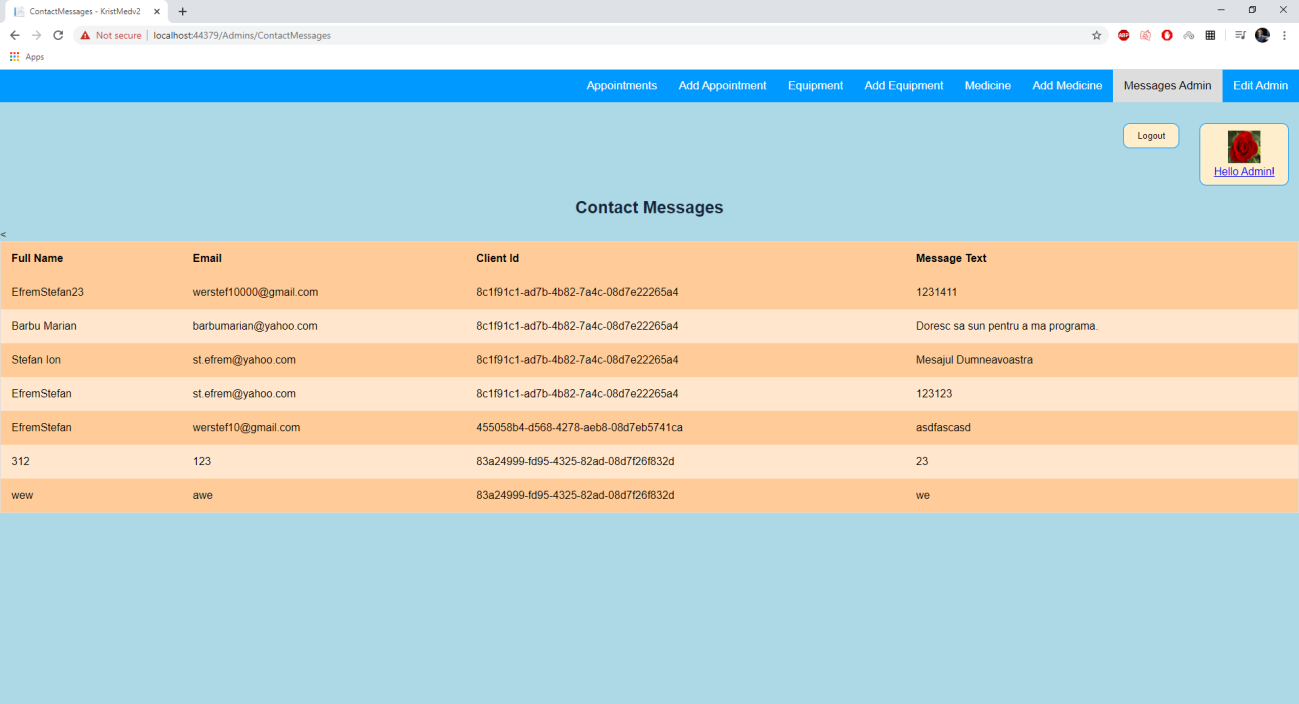
On the “Add appointment” page, the admin can add an appointment at a certain date, for a certain duration and with a certain medic which will be chosen from a select list with all the medics available.



The “Equipment” and “Medicine” pages are similar and they show a list of available equipment and medicine for the treatment of patients by the medic.

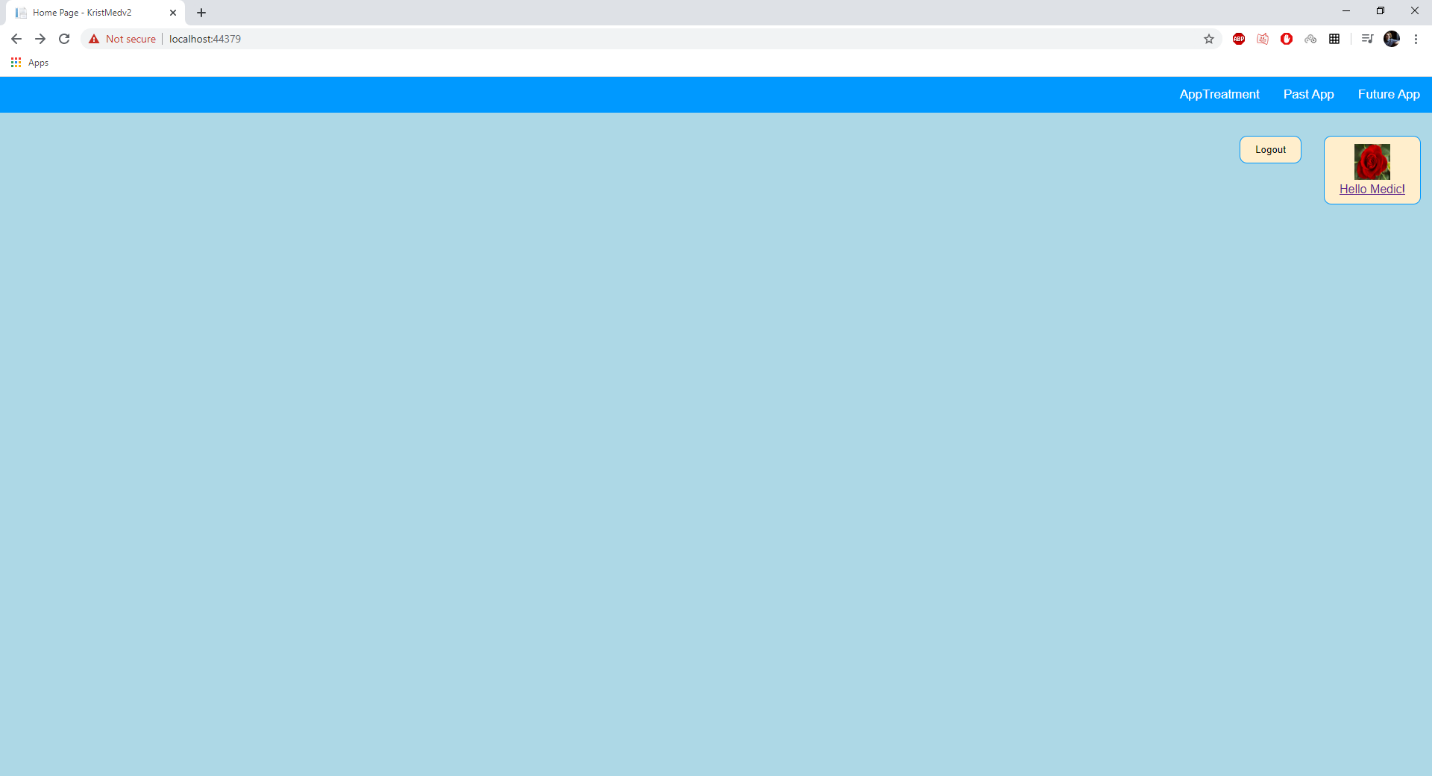


The “Add Equipment Type” and “Add Medicine Type” are also similar as on these pages the admin will add new equipment or medications on the clinics software after they arrive at the clinic.

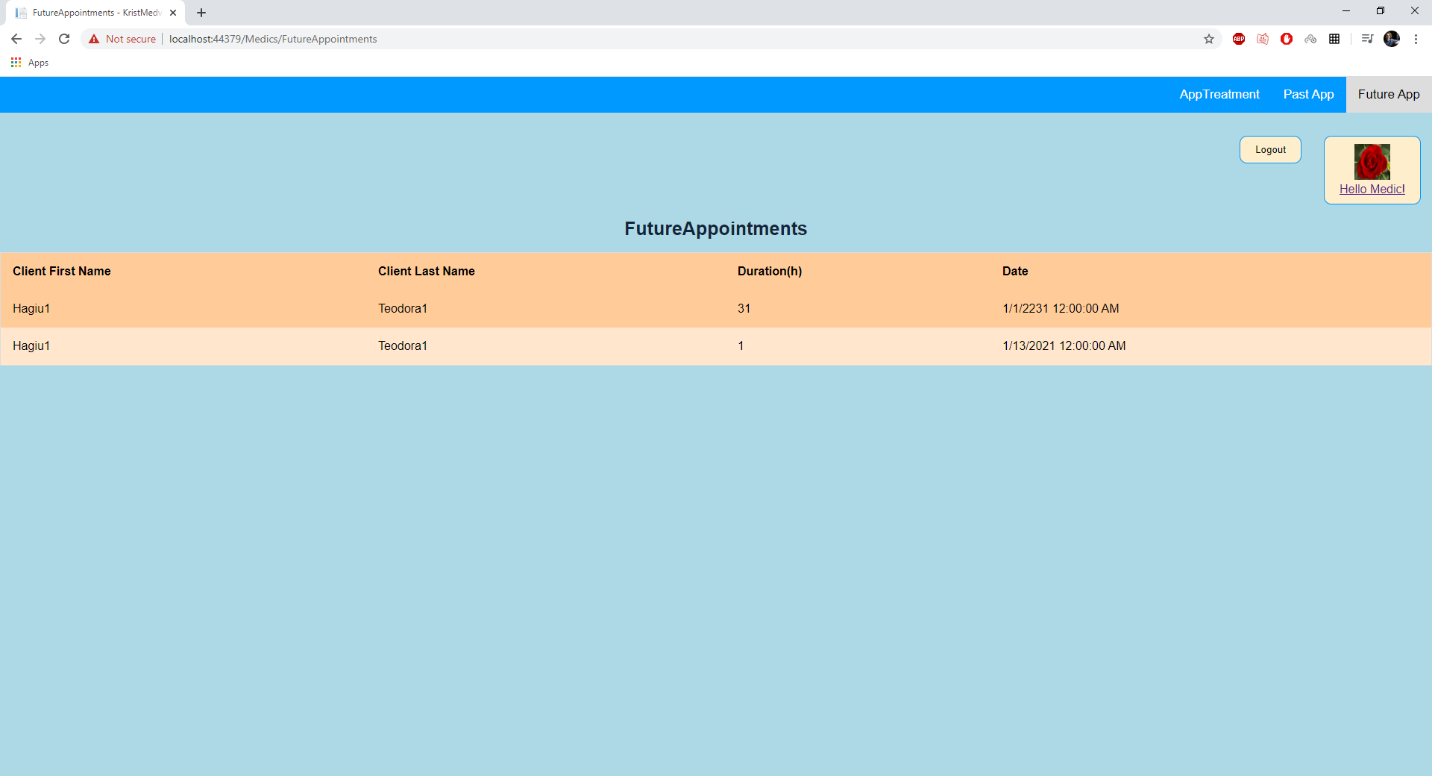


The “Contact Message” page holds a list of all the contact messages sent by the clients.

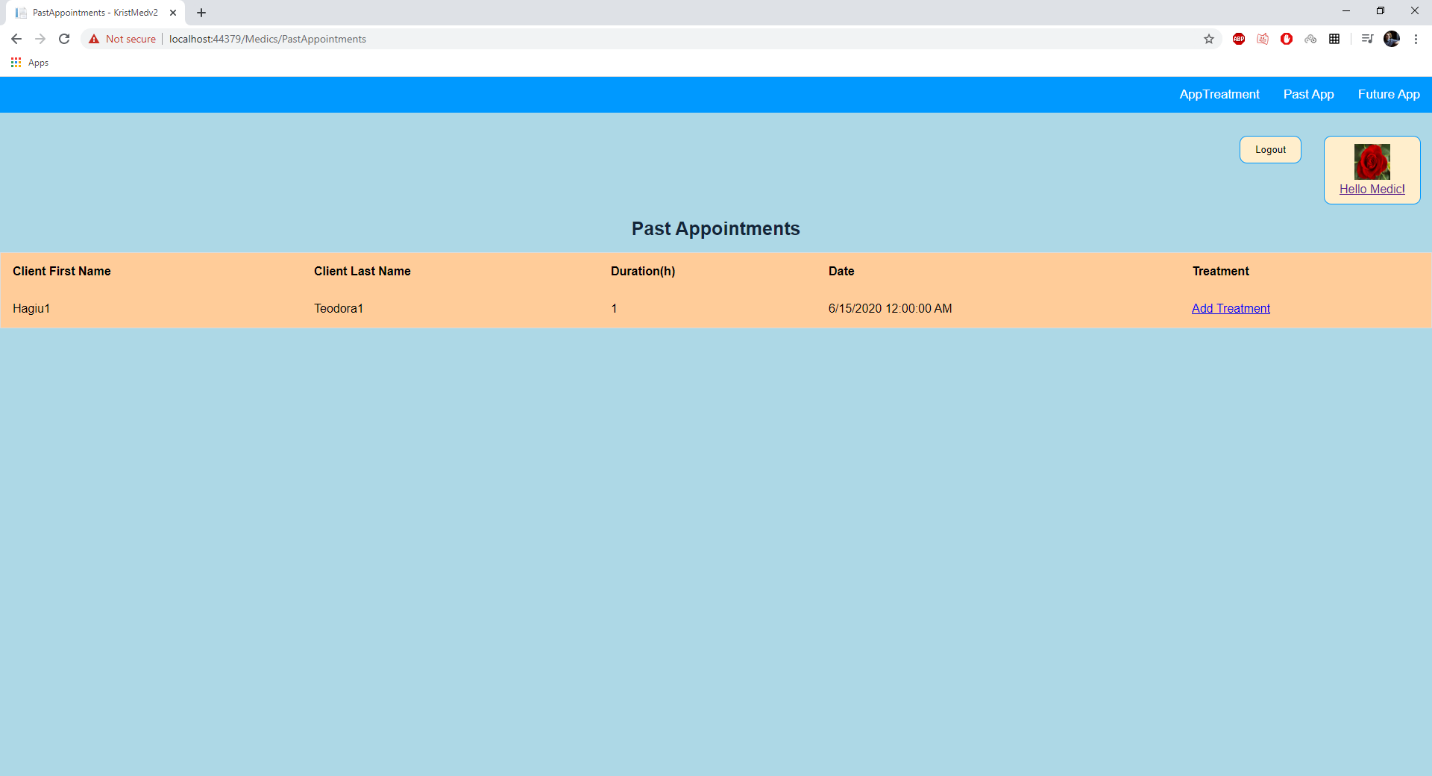
* 1. *Medic functionalities*

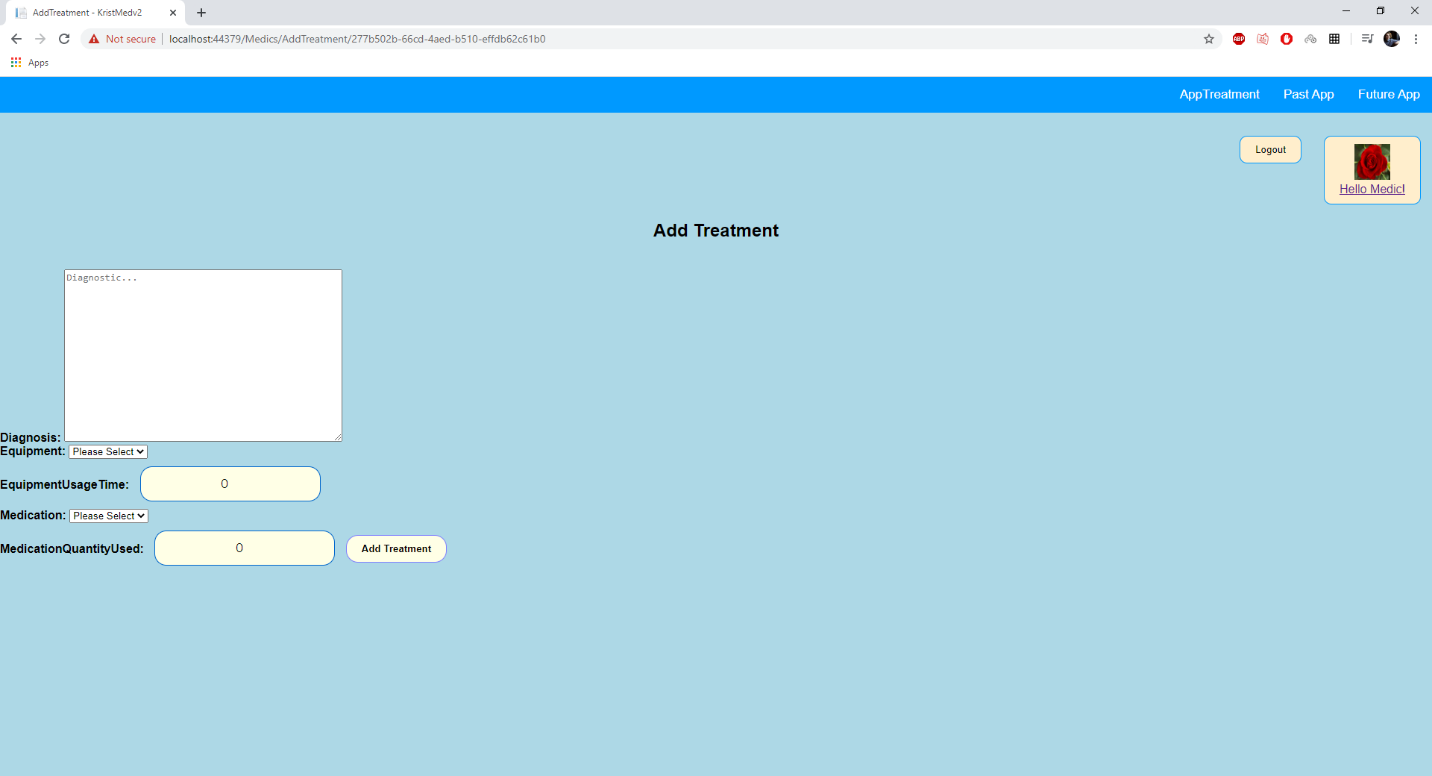


The Medic Home Page has the menu with the pages that are available for the medic.

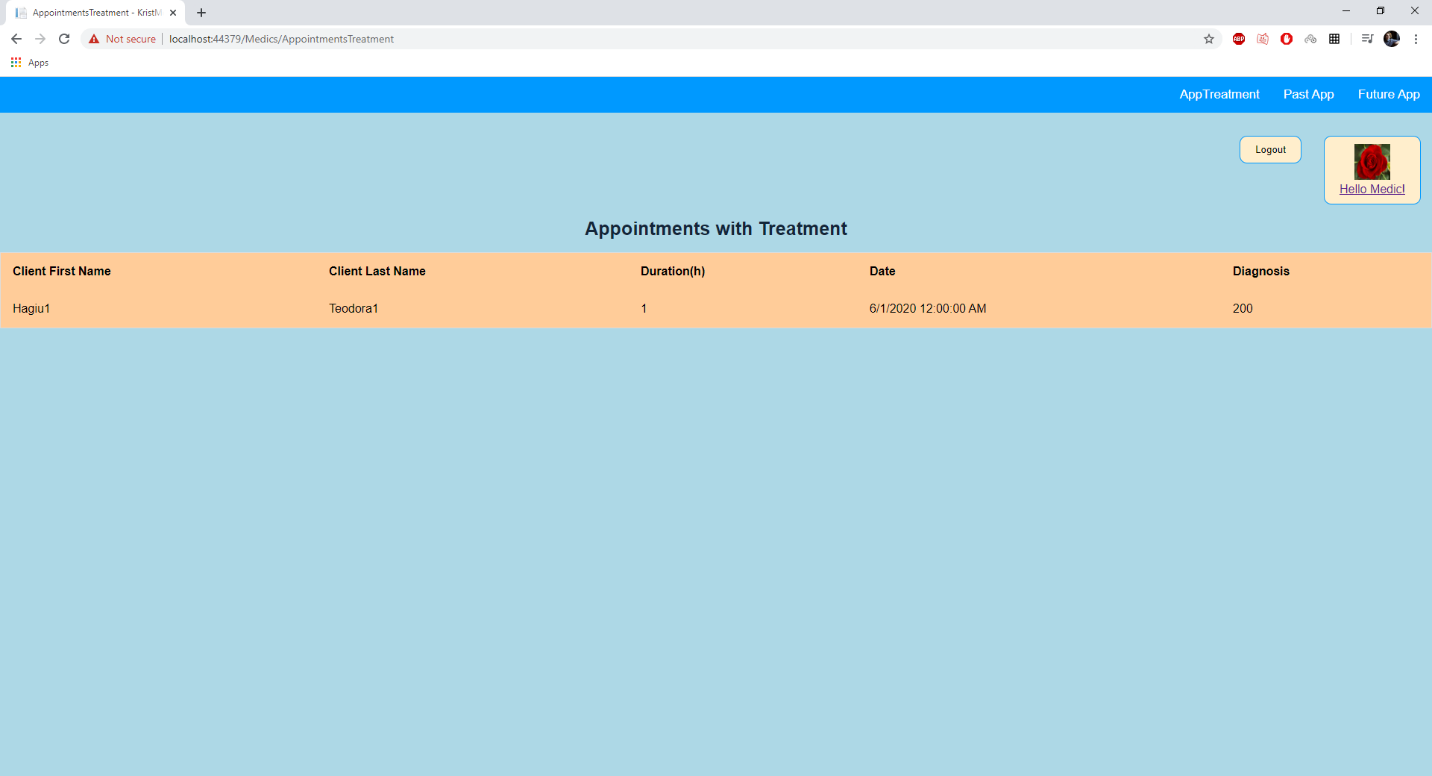


The “Future Appointments” page has a list of all the future appointments with clients of the medic. There are also so called “Fresh Appointments” created by admins which have no client yet and those will not be displayed here.





The “Past App” page displays a list of past appointments of the medic who haven’t had a treatment added to them as of now. There is a button next to every such an appointments that sends the medic to the “Add Treatment” Page where he will fill in the diagnosis of the patient.



The “Appointments with Treatment” page shows a list of past appointments that have treatments created by the medic.

# Plans to improve the Application

The next steps for the app will be implementing the Post Feature which will make possible chatting between the Medic and his/her patients. This is a necessary feature as many patients have certain problems that can be easily solved at the distance and they can receive advice from the medic in this format. The app will also store these messages for long term so the Medic can review the discussion he had with his patients before the appointment. This functionality will help the doctors tremendously in their quest to help patients. AngularJS should be used to make this functionality.

# Interface Design and Tools Used

The interface has well positioned sparse elements that do not overcrowd the screen; this is done to help the users so that they will not be distracted by a lot moving element and they will be able to easily use the services of the app without prior experience with it.

In order to ease the usability of the application, some design guidelines were imposed in the making of the app:

* the response time of the application is less than 0.3 sec
* the pages are sparsely populated with elements that have a decent distance between them, 100-150 px
* an easy to understand top menu bar that will ease the access to all the important services of the app
* the choice of a bicolor interface: a calming cold color (light blue) and a relaxing warm color (pale orange) which gives the website a calming effect which should help ease the stress of using a web application of someone who might not have such a diverse experience with computers or the internet.

In addition, Bootstrap has been added to the frontend to make its accessibility from mobile devices and tablets a lot easier. This has make the app more appealing and easier to access for people that do not own a laptop or a personal computer. The project was made in Visual Studio, using .NET Core in the development of the web application. Sql Server was also used for the database through the Server Explorer feature of Visual Studio. Github has been used to keep track of updates and have a cloud storage feature for the source code of the application.

The internet is a staple of globalization and a symbol of its triumph over the world. Thus it is also one of the most accessible products in the world. Since our application is a web-based one, its accessibility is already on a high note, to that, design decisions such as some of the ones written above, showcase my desire to make the app accessible to older people which might have trouble with computers or the internet, but who are also an important part of the demographic targeted by this application.

Because it is accessible from the browser, that means that it can be used on a personal normal desktop computer or laptop, or using a browser on the phone. So, from the start, by implementing the application as a website, it is widely accessible thanks to the browsers that readily available nowadays.

All in all, the usability and accessibility of the application can be improved (but the decisions in its design have definitely put it on the right track and it achieves all its purposes in regards to its user friendliness.

.