Online Informational System

for **Personalized Health**

Group Members

Vasilis Mavrogeorgis CSD3798

Melina Petousaki CSD3827

Contents

System Introduction:

The **personalized health system** provides a new and improved way for doctors to communicate with their patients and for patients to have easy and complete access to all their medical information.

There are **4 types** of different users that use our system:

- **The Central Administrator** has administrative access to every other type of user with the exception of the Visitors and verifies doctors.
- **Doctors** have access to all their information and partial access to their patients' information.
- **Patients** (Signed In users) have access to all their information and to some specific doctor information.
- **Visitors** have partial access to the system's functionality and don't have any information in the system.

Each type of user has access to their own corresponding UI and functionalities. This includes their own navbar and in most cases their own homepage.

Our implementation uses **Java servlets, html5**(bootstrap 5)**, Javascript, Java, SQL, css3, AJAX, jQuery** and **APIs** (mainly the **REST** API).

All main project goals were completed!

Chapter 1
Administrator

The system administrator has **2** main responsibilities:

1. The **certification** of doctors
2. The **deletion** of doctors and users

The process of certification is complete through the following **Javascript** functions:

- ➢ **adminFunc()** : The admin's main function that loads the **[adminUI.html]** and calls getUncertDocs(), getCertDocs(), and getUsers(), setting up the interface necessary to certify/delete.
- ➢ **getUncertDocs()** : Creates the boxes that hold the information of all the uncertified doctors, and gives the ability to certify and delete each through calls of certifyDoc(doctor_id) and deleteDoc(doctor_id) respectively. We get all those doctors through an ajax request to the **REST** API using a **GET** HTTP message.
- ➢ **certifyDoc(doctor_id)** : Certifies a doctor and recalls adminFunc() to get the updated doctors/patients. We achieve that through an ajax request to the **REST** API using a **PUT** HTTP message.
- ➢ **getCertDocs()** : Creates the boxes that hold the information of all the certified doctors, and gives the ability to delete each through calls of deleteDoc(doctor_id). We get all those doctors by calling findDocs()(see pg. 11) and waiting for it to return them in the form of a json.
- ➢ **deleteDoc()** : Deletes the doctor from the database and recalls adminFunc() to get the updated doctors/patients. We achieve that through an ajax request to the **REST** API using a **DELETE** HTTP message.
- ➢ **getUsers()** : Creates the boxes that hold the information of all the patients, and gives the ability to delete each through calls of the deleteUser(user_id) function. We get all those patients through an ajax request to the **REST** API using a **GET** HTTP message.
- ➢ **deleteUser(user_id)** : Deletes the patient from the database and recalls adminFunc() to get the updated doctors/patients. We achieve that through an ajax request to the **REST** API using a **DELETE** HTTP message.

Chapter 2
Doctor

The doctor has **5** main functionalities:

1. Creation of new appointments
2. Access and editability of their appointments
3. Access to their patient's medical records and history
4. Registration of new treatments
5. Display of their patient's messages

The creation and editing of appointments is done through the following functions:

- ➢ **makeCalendar()** : Loads the corresponding html file [**dcrandevouz.html**] and calls the renderDate() and checkRandevouz(date) functions.
- ➢ **renderDate() :** Creates the calendar for the interface and passes the date chosen as an argument to checkRandevouz(date).
- ➢ **loadCreateRand()** : Called when a new rendezvous is to be added; loads the corresponding html file [**createRandevouz.html**]
- ➢ **checkRandevouz(date)** : Gets all of the doctor's rendezvous for the given [date] by using an ajax request to the **REST** API using a **GET** HTTP message (if no date is chosen then the default is today with date value null) and by calling the **Doctor_checkRandevouz(randevouz_date,randevouz_status,randevouz_id, date)** function the appointments are displayed on the screen.
- ➢ **cancelRandevouz(randevouz_id, date)** : Changes the status of the chosen rendezvous to *'cancelled'* through an ajax request to the **REST** API using a **PUT** HTTP message.
- ➢ **completeRandevouz(randevouz_id, date)** : Changes the status of the chosen rendezvous to *'done'* through an ajax request to the **REST** API using a **PUT** HTTP message.
- ➢ **addRandevouz()** : Takes the inputted information by the doctor and creates a new rendezvous, and if the information meets the conditions (future date / 30 minute appointment) it is inserted to the database. This is done through an ajax request to the **REST** API and a HTTP **POST** message.
- ➢ **PDF Create** : A html hyperlink (see pg. 26) that shows the chosen date's rendezvous in pdf form. (available only when there are rendezvous on that date)

The access to and display of the patient's medical records and history is done through the following functions:

- ➢ **loadPatientCheck()** : Loads the corresponding html file [**docPatients.html**] and calls the checkPatients() function. In order for the html file to be loaded we use jQuery.
- ➢ **checkPatients()** : Creates the boxes that hold the information of all the patients with which the doctor has had at least one successful (*'done'*) rendezvous. We get the user_id of those patients through an ajax request to the **REST** API using a **GET** HTTP message.
- ➢ **loadMedicalRec(amka, userID,fname,lname)** : Loads the corresponding html file [**seeBloodtest.html**] that displays all the medical exams and treatments the patient has done. It also makes changes to the seeTreatments and seeBloodtest functions in order for these functions to know from where they have been called (doctor or user) since they are also being used by the user. In order for the html file to be loaded we use jQuery.
- ➢ **seeBloodtest(flag, amka, userID)** : Creates the table that shows and compares all or specific blood tests (the user can choose the dates). For the doctor, the table has an extra column that is a button that adds a new treatment to the corresponding blood test, using **loadaddNewTreatement(bloodtestID, doctorID, userID)**. The flag is used in order for the function to know what blood tests to show; whether to show all or specific tests based on their date. We gain access to the blood test's information through an ajax request to the **REST** API using a **GET** HTTP message.
- ➢ **createChart(User_amka)** : Creates a chart that visualises the chosen measurements as a graph, and creates the rows with the help of **createChartElements(blood_test)** . The measurements are chosen by the user through **biomarkersDisplay()**
  Chart creation functions courtesy of google charts.
- ➢ **seeTreatments(flag, location, userID)** : Creates the table that shows all or specific treatments that have been administered to the patient (the user can choose the dates). The flag is used in order for the function to know what treatments to show; whether to show the treatments of a specific chosen time period or all of them. The location is used to specify in which part of the html file the treatment table should appear. We use it because this function can be called in different parts of the code and by different users. We gain access to the treatment's information through an ajax request to the **REST** API using a **GET** HTTP message.
- ➢ **loadaddNewTreatement(bloodtestID, doctorID, userID) :** Loads the corresponding html file [**addNewTreatment.html**] and calls the function addNewTreatment(bloodtestID, doctorID, userID). In order for the html file to be loaded we use jQuery.

- ➢ **getUserData(userID,doctorID)** : Fills the boxes created by **checkPatients()** with the corresponding patient's information , using html and through an ajax request with a **GET** HTTP message.
- ➢ **addNewTreatment(bltestID, docID, userID)** : Creates a new treatment with the information inserted by the doctor and if the information meets the conditions (starting date is before end date/there is treatment information) it is inserted to the database . This is done using an ajax request to the **REST** API with a **POST** HTTP message.

The doctor is able to exchange messages with all patients that they have had at least one (*'done'*) successful rendezvous; this is possible through the following functions:

- ➢ **loadInbox()** : Loads the corresponding html file [**seeMessages.html**] and calls the inbox() function. In order for the html file to be loaded we use jQuery.
- ➢ **inbox()** : Creates the table that holds the messages that any eligible patient has sent to the doctor. The message's content is available to us through an ajax request to the **REST** API with a **GET** HTTP message.
- ➢ **msgUser(doctor_id,user_id)** : Creates the text area that is used to write the message and calls **clearDocMsg(user_id)** when the message is discarded and sendMsg(sender, receiver, type) when the message is to be sent**.**
- ➢ **sendMsg(sender, receiver, type)** : Sends the message written to the database using an ajax request to the **REST** API with a **POST** HTTP message. This function is also used by the user, as such we have the [type] variable so as to recognize what kind of user has called the function.

Chapter 3
Signed In User

A signed in user has **6** main functionalities:

1. Finding doctors based and sorted by 3 criteria (price, distance, car)
2. Booking , viewing and cancelling appointments
3. Adding new examinations
4. Comparing older examinations
5. Displaying treatments
6. Sending to and viewing messages from their doctor

Finding certified doctors as a signed user, sending messages, and booking appointments is completed with the following functions:

➢ **callUserFindDocs()** : Called when clicking on Find Doctors on the navbar, this function clears previously visible elements and calls userFindDoc().

➢ **userFindDoc()** : Calls visFindDoc() (see pg. 11), adds a form so the user can decide based on what to sort the doctors by (default being by price), and enhances the visitor's basic interface with the options available to the signed user by using the function addUserButtonsFindDoc();

➢ **sortDoctors()** : The function responsible for sorting the doctors. If the user chose to sort by price it simply calls findDocs() to get the doctors in json form and creates the boxes that hold their information, since the json that is returned has the doctors already sorted by price.

If the user chooses to sort by distance or car, it starts with calling findDocs() and getUser() and waiting for their results to return so that they can be stored. If the user has yet to certify his location, the function simply displays an error message. If he has certified his location, it divides the doctors into those with uncertified locations and those with certified. The ones that have had their location certified are converted to a valid request form to be sent to the trueway-matrix **API** in order to calculate the distance/time required to reach each doctor with a certified location.

It then creates the boxes that hold their information, sorted by the values returned from the API, with those that it could calculate a distance for first, those that couldn't find a valid path to them second, and those without a certified location last.

Finally it enhances the interface with the options available to the signed user by using addUserButtonsFindDoc().

- **addUserButtonsFindDoc()** : Calls addMsgRandvz(doctor_id, user_id) for each doctor's box.
- **addMsgRandvz(doctor_id, user_id)** : Starts by making an ajax request to the **REST** API with a **GET** HTTP message to see whether the user has completed a rendezvous with the doctor in question, allowing them to start sending messages to the doctor if the answer is yes, through the msgDoc(user_id, doctor_id).
  Afterwards it checks whether the doctor has any available rendezvous by making an ajax request to the **REST** API with a **GET** HTTP message. If the answer is yes, it enables them to activate randvzDoc(user_id, doctor_id) to see the doctor's available rendezvous.
- **msgDoc(user_id, doctor_id)** : calls clearMsg() and creates a textarea so that the user will be able to write his message using sendMsg(user_id, doctor_id, 0) to send it.
- **clearMsg()** : clears any open textarea for messaging and any doctor's available rendezvous.
- **sendMsg(sender, receiver, type)** : if type = 0, a user sent the message, else a doctor sent the message. Then through the use of an ajax request to the **REST** API with a **POST** HTTP message the new message gets stored to the database, ready to be read by the recipient. Also calls clearMsg() and its equivalent clearDocMsg(user_id).
- **randvzDoc(user_id, doctor_id)** : calls clearMsg() and then shows the doctor's available rendezvous in detail and gives the user the ability to start booking through the function showBook(randevouz_id, user_id).
- **showBook(randevouz_id, user_id)** : creates a textarea for the user to write special notes for the rendezvous to be read by the doctor and gives them the option to finalise their booking with bookRzv(randevouz_id, user_id).
- **bookRzv(randevouz_id, user_id)** : books the rendezvous by updating its status in the database to *'selected'* and adding the user's id and their notes to it. This is done by using an ajax request to the **REST** API with a **PUT** HTTP message. Finally the function calls clearMsg().

The insertion of a new blood test is completed with the use of an event listener and the following function:

➢ **Event Listener**: Positioned at the Examinations > Add New link, when activated it loads the corresponding html file [**addBloodtest.html**]. In order for the html file to be loaded we use jQuery.
➢ **addBloodtest()** : Takes the inserted blood test data and sends it to the database. *Due to conversion problems with GSON there could not be any empty data sent to the blood test, as such if there wasn't a value for a measurement we would set its value to zero.* This is done by using an ajax request to the **REST** API with a **POST** HTTP message.

The display and comparison of the user's blood test examinations is done through the following functions :

➢ **loadSeeBT()** : Loads the corresponding html file [**seeBloodtest.html**] and calls the seeBloodtest(flag) and seeTreatments(flag, "treatments") functions.
➢ **seeBloodtest(flag, amka, userID)** : Creates the table that shows and compares all or specific blood tests (the user can choose the dates). The flag is used in order for the function to know what blood tests to show; whether to show all or specific tests based on their date. We gain access to the blood test's information through an ajax request to the **REST** API using a **GET** HTTP message.
➢ **createChart(User_amka)** : Just like the doctor (see pg. 5) .
➢ **seeTreatments(flag, location, userID)** : Just like the doctor (see pg. 5) .

The display of treatments is done in the same way as in the doctor with the location changing in order for the function to know where to display the contents.

Each patient is able to exchange messages with any doctors that they have had at least one (*'done'*) successful rendezvous. The display of messages is done the same way as the doctor (see pg. 6) through the inbox tab in the navbar. However, sending messages is part of the Find Doctors tab and is explained above (see pg. 8).

Booking is done through the functions explained above (see pg.8) used in the find doctors tab of the navbar.

Viewing and cancelling appointments is done through the following functions :

➢ **getPending()** : Creates boxes that are filled with the information of any pending rendezvous the patient has.It also provides a way to cancel said rendezvous using the function cancelBook(randevouz_id). The function gets the rendezvous through an ajax request to the **REST** API with a **GET** HTTP message.
➢ **cancelBook(randevouz_id)** : Has the same base function as the cancelRandevouz(randevouz_id, date) from the doctor. Changes the status of the chosen rendezvous to *'cancelled'* through an ajax request to the **REST** API using a **PUT** HTTP message.
➢ **getDoneRandvz()** : Creates boxes that are filled with the information of any successfully completed (*'done'*) rendezvous the patient has. The information of the rendezvous is available to us through an ajax request to the **REST** API with a **GET** HTTP message.

Chapter 4

Simple User

The guest user doesn’t have any information in the system in order to personalise and enhance their experience. However they still have **1** main functionality at their disposal:

1. Finding Doctors on a map

Finding all doctors on a map -doesn’t have option to sort- is done with the following functions:

- ➢ **callVisFindDocs()** : Hides any previous UI and calls visFindDoc().
- ➢ **visFindDoc()** : Creates the html interface that will be used to display the doctors and the map. Calls findDocs() and makeMap() functions.
- ➢ **findDocs()** : Provides all the available doctor information through an ajax request to the **REST** API with a **GET** HTTP message.

Chapter 5
Basic System Interfaces

As mentioned on the system introduction, each type of user has their own interfaces for them to work on. These interfaces are based on a single main html file, the **index.html** which holds all the libraries used and has the main structure of each page. Our implementation uses the **Bootstrap 5** library, and more specifically the button class and the grid layout system.

Every page is made up of 4 parts, each with its own corresponding <div> :

- A navbar : Changes depending on the type of user and is always visible on every page.
    - **index.html** contains the default navbar which has only 5 tabs:
        - Home
        - Find Doctors
        - Top
        - Login
        - Sign Up
    - **user_navbar.html** is used for the signed in users and has 9 tabs:
        - Home
        - Find Doctors
        - Randevouz
            - Pending
            - Done
        - Examinations
            - Add new
            - See exams
        - Treatments
            - Active
            - Finished
        - Inbox
        - Profile
        - Top
        - Logout

- **Doc_navbar.html** is used for the doctor and has 7 tabs:
    - Home
    - Patients
    - Randevouz
    - Inbox
    - Profile
    - Top
    - Logout

- **Admin_navbar.html** is used for the admin and has 4 tabs:
    - Home
    - Profile
    - Top
    - Logout

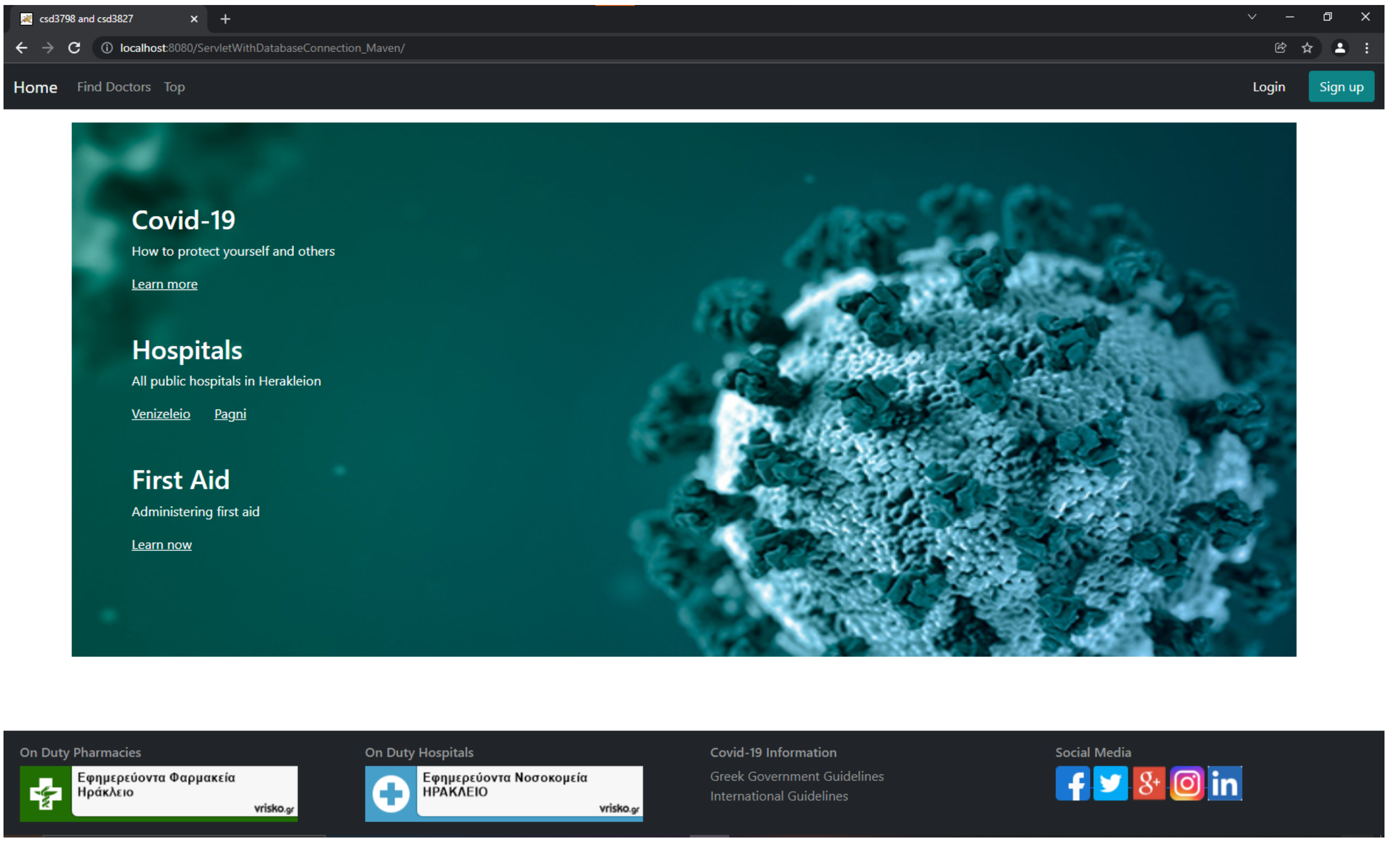
- A homepage
    - **index.html** a simple homepage used for both the guest user and the doctor.
    - **userUI.html** shows the calculation for BMI and Ideal Weight based on the user's information.
    - **adminUI.html** used by adminFunc to show all the users and doctors (see pg. 3)

- The main Part
    - The main Part is the centerpiece of our design as everytime we wish to display anything on the page we use it as the base on which we load the html files.

- A footer
    - The footer holds all the minor information that the user might want to have access to such as which hospitals/pharmacies are on duty, or what are the current guidelines about covid-19 and finally a way to connect using social media. The footer is ever present on every page we display.

Other interfaces used are the **sign up, login,** and **edit/profile** which were mostly kept the same as in assignment 3, with a few tweaks here and there.

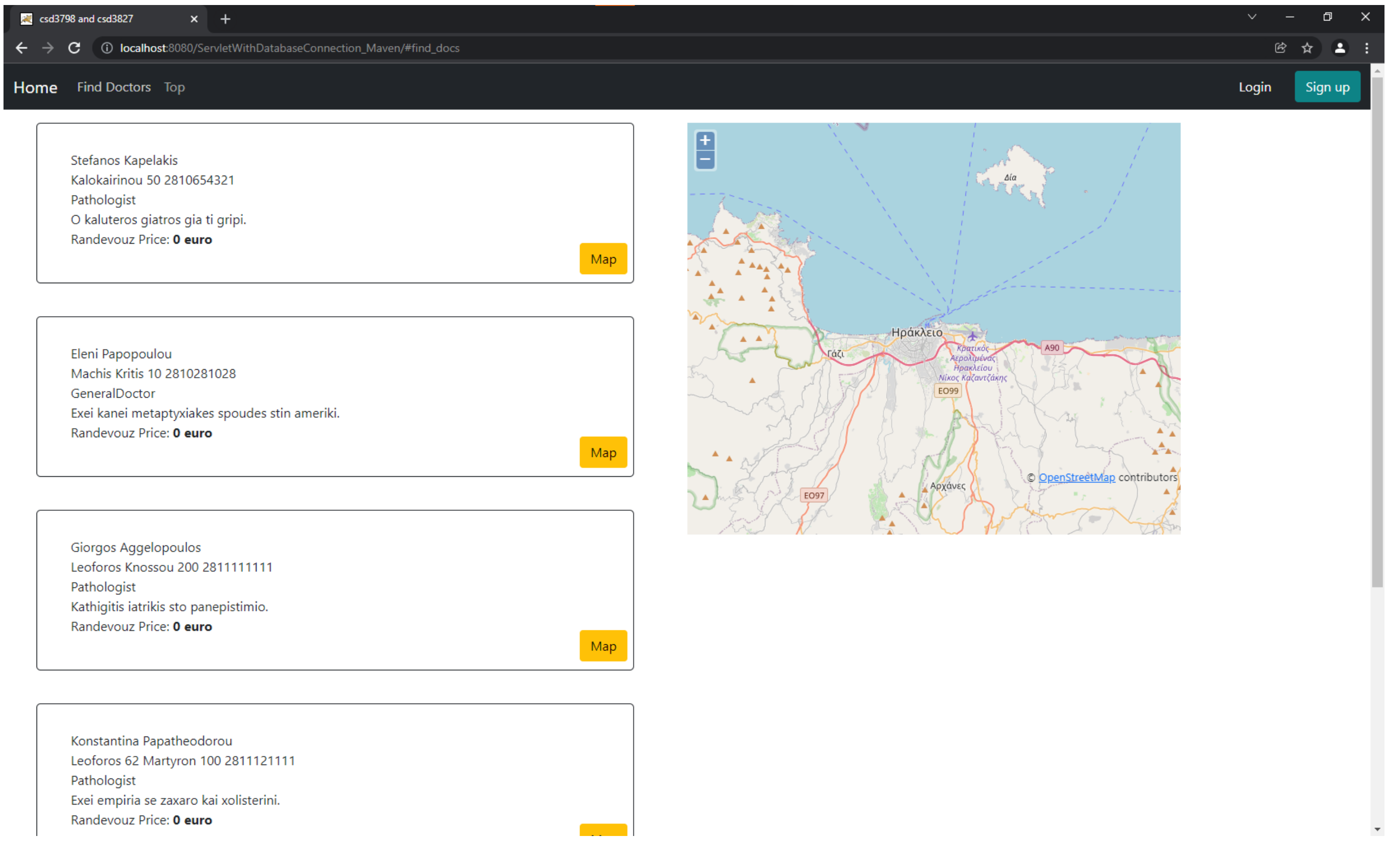
- **Index / Default Homepage**



Main divs/tags :

1. <nav class="navbar navbar-expand-sm bg-dark navbar-dark sticky-top">
   Creation of default navbar
2. <div id = "homepage">
   Uses bootstrap grid layout; contains the image with text and links on top of it.
3. <div id ="mainPart" >
   Used by other html files as the base
4. <div id = "footerPart" class="mt-auto">
   Footer; holds all the extra , not as important information

- **Find Doctors (guest user)**


csd3798 and csd3827
localhost:8080/ServletWithDatabaseConnection_Maven/#find_docs
Home
Find Doctors
Top
Login
Sign up
Map
Map
Map
Ηράκλειο
Δία
Γάζι
Αρχάνες
© OpenStreetMap contributors

Stefanos Kapelakis
Kalokairinou 50 2810654321
Pathologist
O kaluteros giatros gia ti gripi.
Randevouz Price: **0 euro**

Eleni Papopoulou
Machis Kritis 10 2810281028
GeneralDoctor
Exei kanei metaptyxiakes spoudes stin ameriki.
Randevouz Price: **0 euro**

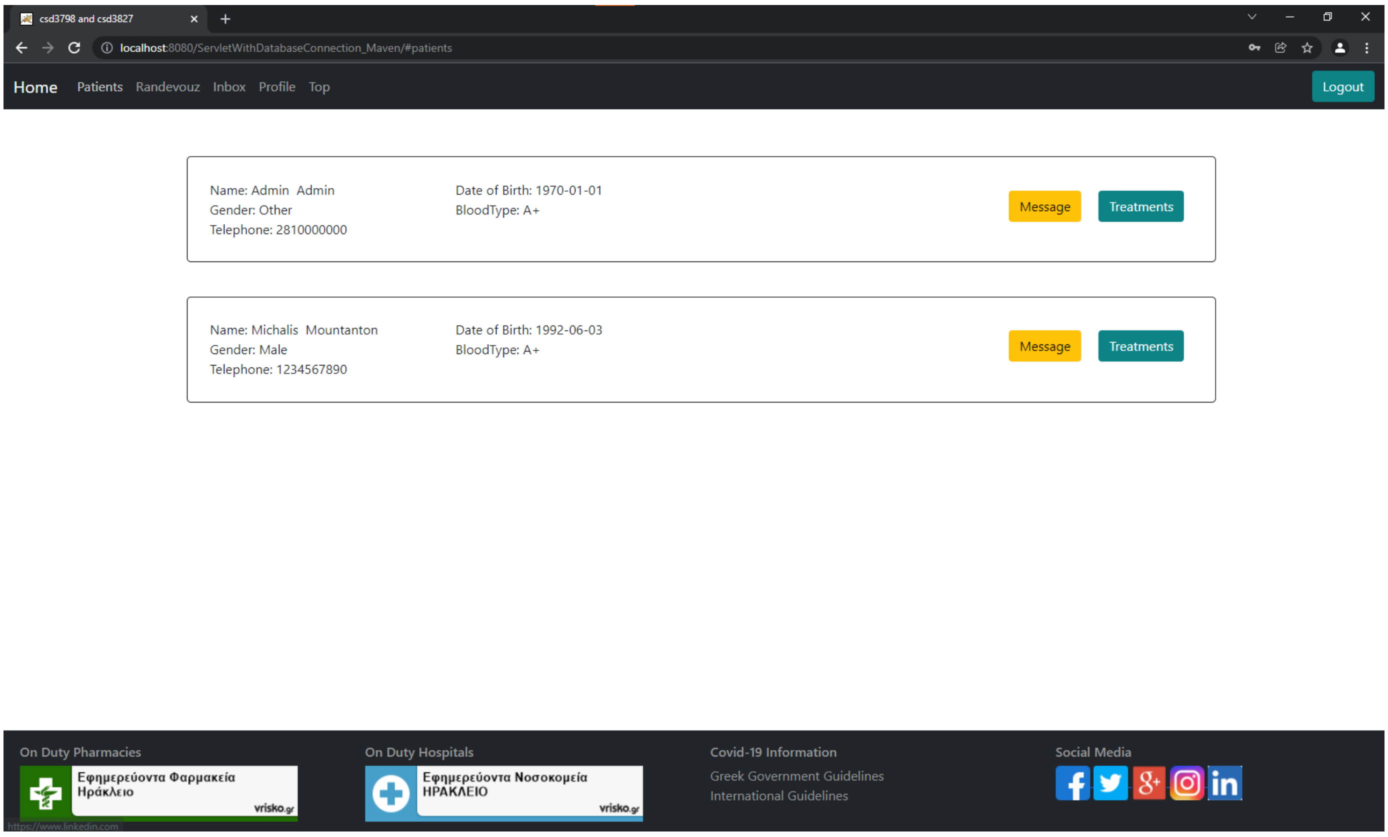
Giorgos Aggelopoulos
Leoforos Knossou 200 2811111111
Pathologist
Kathigitis iatrikis sto panepistimio.
Randevouz Price: **0 euro**

Konstantina Papatheodorou
Leoforos 62 Martyron 100 2811121111
Pathologist
Exei empiria se zaxaro kai xolisterini.
Randevouz Price: **0 euro**

Main divs/tags :

1. <div id ="mainPart" >
   Uses bootstrap grid layout to split page
2. <div id =”docTabl”>
   Holds the left part of the page and is the parent to
   <div class = “doc_buttons” id = “docID”>
3. <div id="Map" style="height:500px; width:600px; position:fixed; " class="olMap">
   Holds the right part of the page and has the map displayed

- **Patients (doctor)**



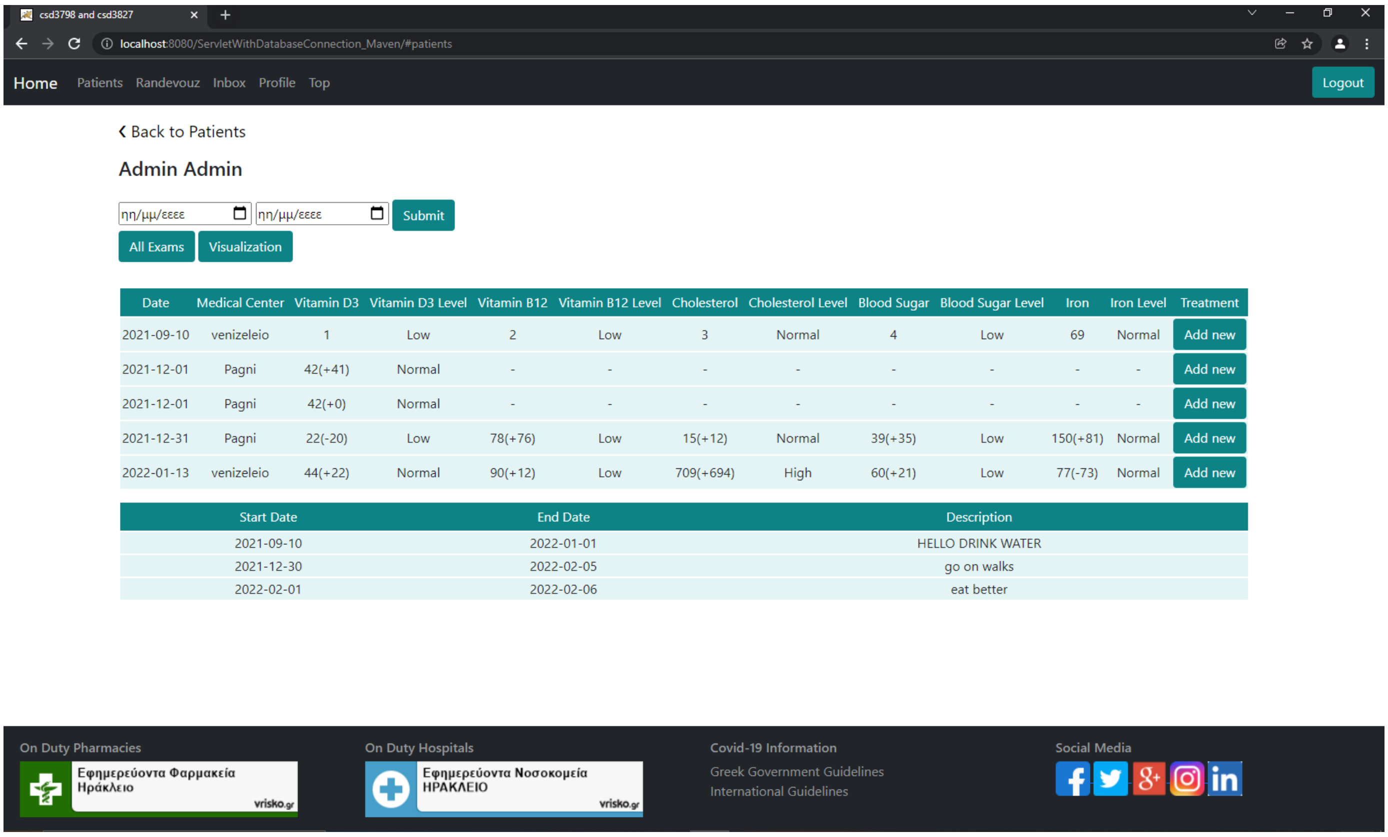
Name: Admin Admin
Gender: Other
Telephone: 2810000000
Date of Birth: 1970-01-01
BloodType: A+

Name: Michalis Mountanton
Gender: Male
Telephone: 1234567890
Date of Birth: 1992-06-03
BloodType: A+

Main divs/tags :

1. <div id ="mainPart" >
2. <div id ="patients"></div>
   Uses bootstrap to align data and buttons and is the parent to <div class="userID_pat" id="userid"> , which is each individual box

- **Patient > Treatments**



csd3798 and csd3827

localhost:8080/ServletWithDatabaseConnection_Maven/#patients

Home Patients Randevouz Inbox Profile Top Logout

‹ Back to Patients

Admin Admin

ηη/μμ/εεεε ηη/μμ/εεεε Submit

All Exams Visualization

Date	Medical Center	Vitamin D3	Vitamin D3 Level	Vitamin B12	Vitamin B12 Level	Cholesterol	Cholesterol Level	Blood Sugar	Blood Sugar Level	Iron	Iron Level	Treatment
2021-09-10	venizeleio	1	Low	2	Low	3	Normal	4	Low	69	Normal	Add new
2021-12-01	Pagni	42(+41)	Normal	-	-	-	-	-	-	-	-	Add new
2021-12-01	Pagni	42(+0)	Normal	-	-	-	-	-	-	-	-	Add new
2021-12-31	Pagni	22(-20)	Low	78(+76)	Low	15(+12)	Normal	39(+35)	Low	150(+81)	Normal	Add new
2022-01-13	venizeleio	44(+22)	Normal	90(+12)	Low	709(+694)	High	60(+21)	Low	77(-73)	Normal	Add new

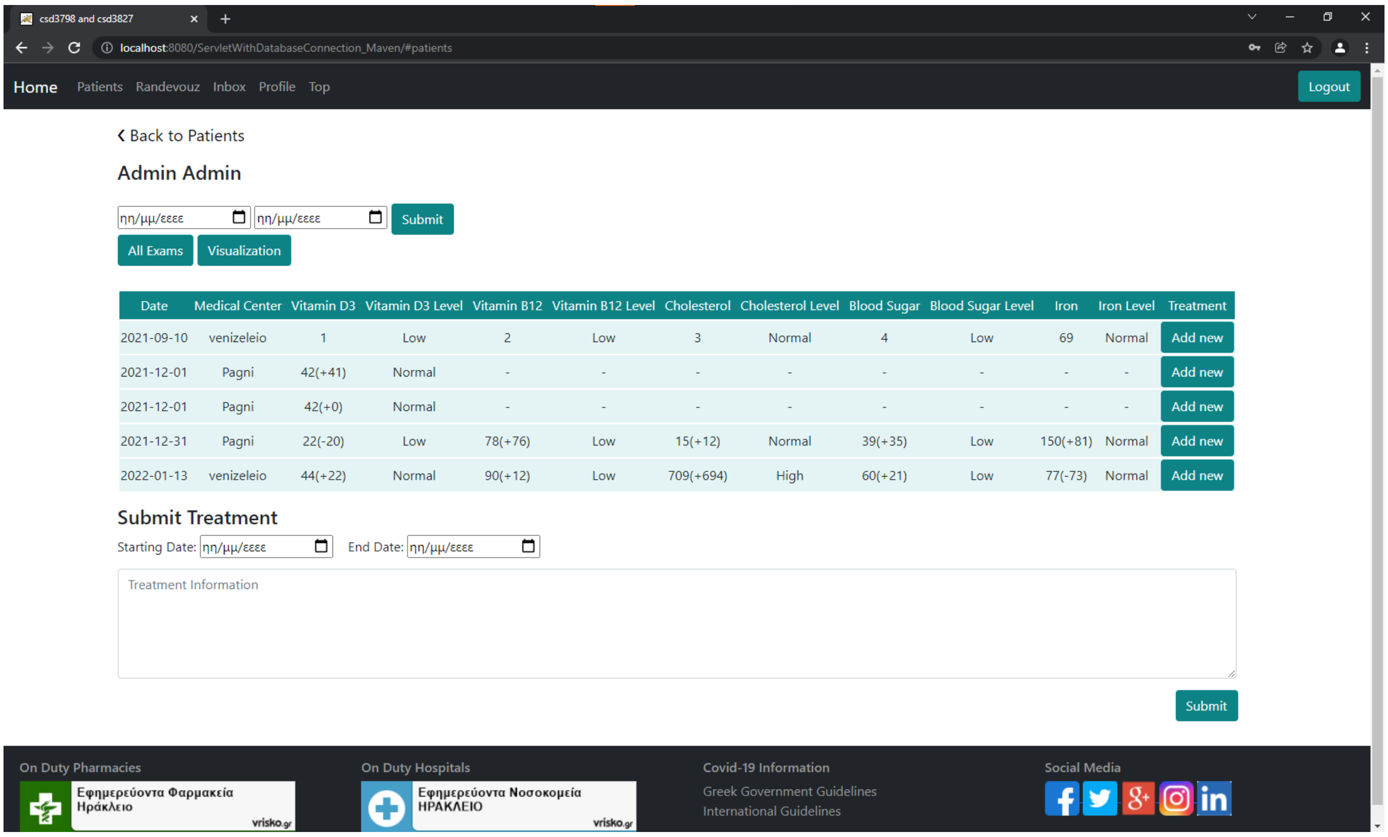
Start Date	End Date	Description
2021-09-10	2022-01-01	HELLO DRINK WATER
2021-12-30	2022-02-05	go on walks
2022-02-01	2022-02-06	eat better

On Duty Pharmacies — Εφημερεύοντα Φαρμακεία Ηράκλειο vrisko.gr

On Duty Hospitals — Εφημερεύοντα Νοσοκομεία ΗΡΑΚΛΕΙΟ vrisko.gr

Covid-19 Information — Greek Government Guidelines, International Guidelines

Social Media



csd3798 and csd3827

localhost:8080/ServletWithDatabaseConnection_Maven/#patients

Home Patients Randevouz Inbox Profile Top Logout

‹ Back to Patients

Admin Admin

ηη/μμ/εεεε ηη/μμ/εεεε Submit

All Exams Visualization

Date	Medical Center	Vitamin D3	Vitamin D3 Level	Vitamin B12	Vitamin B12 Level	Cholesterol	Cholesterol Level	Blood Sugar	Blood Sugar Level	Iron	Iron Level	Treatment
2021-09-10	venizeleio	1	Low	2	Low	3	Normal	4	Low	69	Normal	Add new
2021-12-01	Pagni	42(+41)	Normal	-	-	-	-	-	-	-	-	Add new
2021-12-01	Pagni	42(+0)	Normal	-	-	-	-	-	-	-	-	Add new
2021-12-31	Pagni	22(-20)	Low	78(+76)	Low	15(+12)	Normal	39(+35)	Low	150(+81)	Normal	Add new
2022-01-13	venizeleio	44(+22)	Normal	90(+12)	Low	709(+694)	High	60(+21)	Low	77(-73)	Normal	Add new

Submit Treatment

Starting Date: ηη/μμ/εεεε End Date: ηη/μμ/εεεε

Treatment Information

Submit

On Duty Pharmacies — Εφημερεύοντα Φαρμακεία Ηράκλειο vrisko.gr

On Duty Hospitals — Εφημερεύοντα Νοσοκομεία ΗΡΑΚΛΕΙΟ vrisko.gr

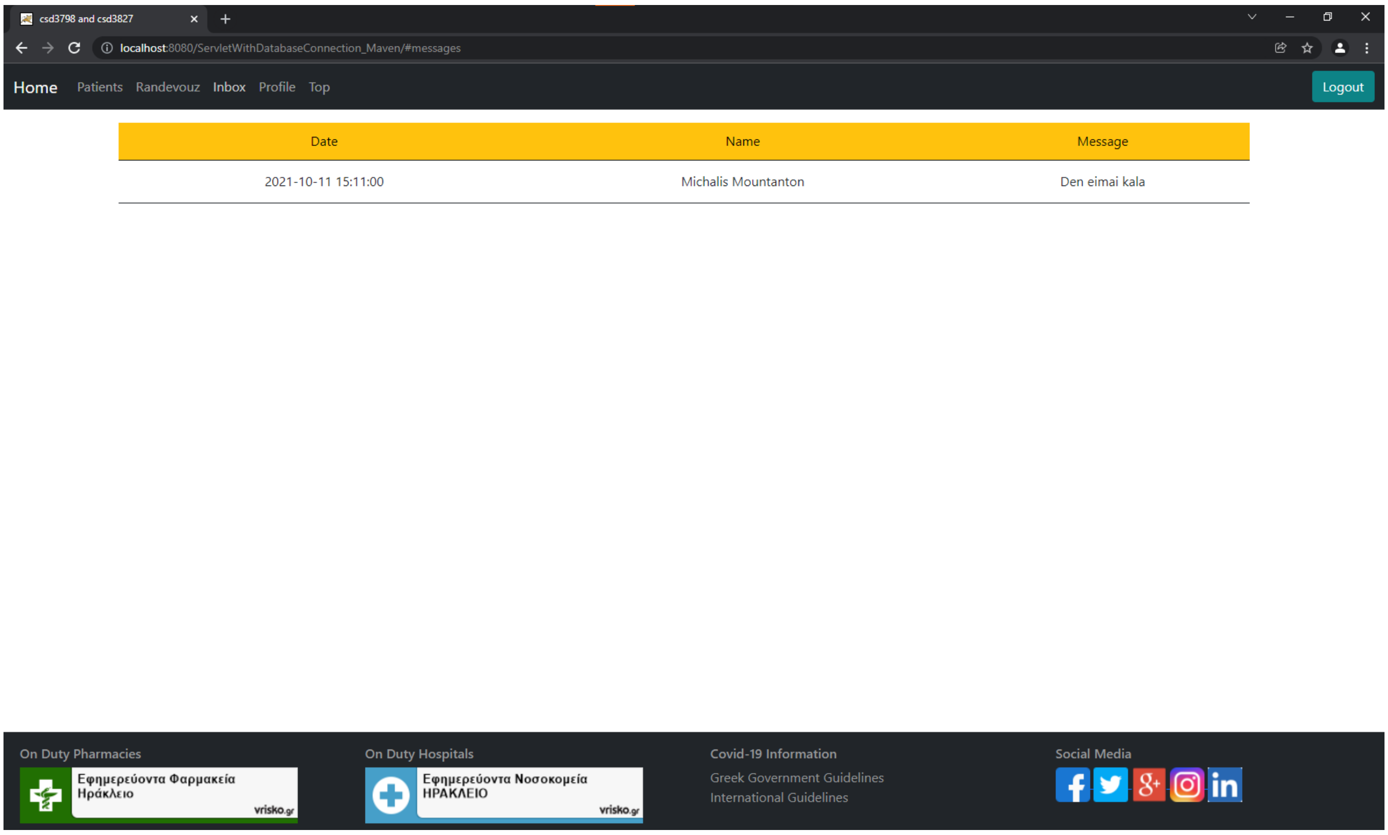
Covid-19 Information — Greek Government Guidelines, International Guidelines

Social Media

Main divs/tags :

1. <div id ="mainPart" >
   Uses bootstrap grid layout to have different rows of content
2. <div id="filters">
   Has the date selection filters
3. <div id="bloodtests" style="margin-top:1%;margin-bottom: 1%;">
   Holds the table with the blood tests and the add new buttons
4. <div id="treatments" style="margin-top:1%;margin-bottom: 1%;">
   Holds the table with the treatments and is changed to an add new treatment form when the add new button is clicked

- ## Inbox

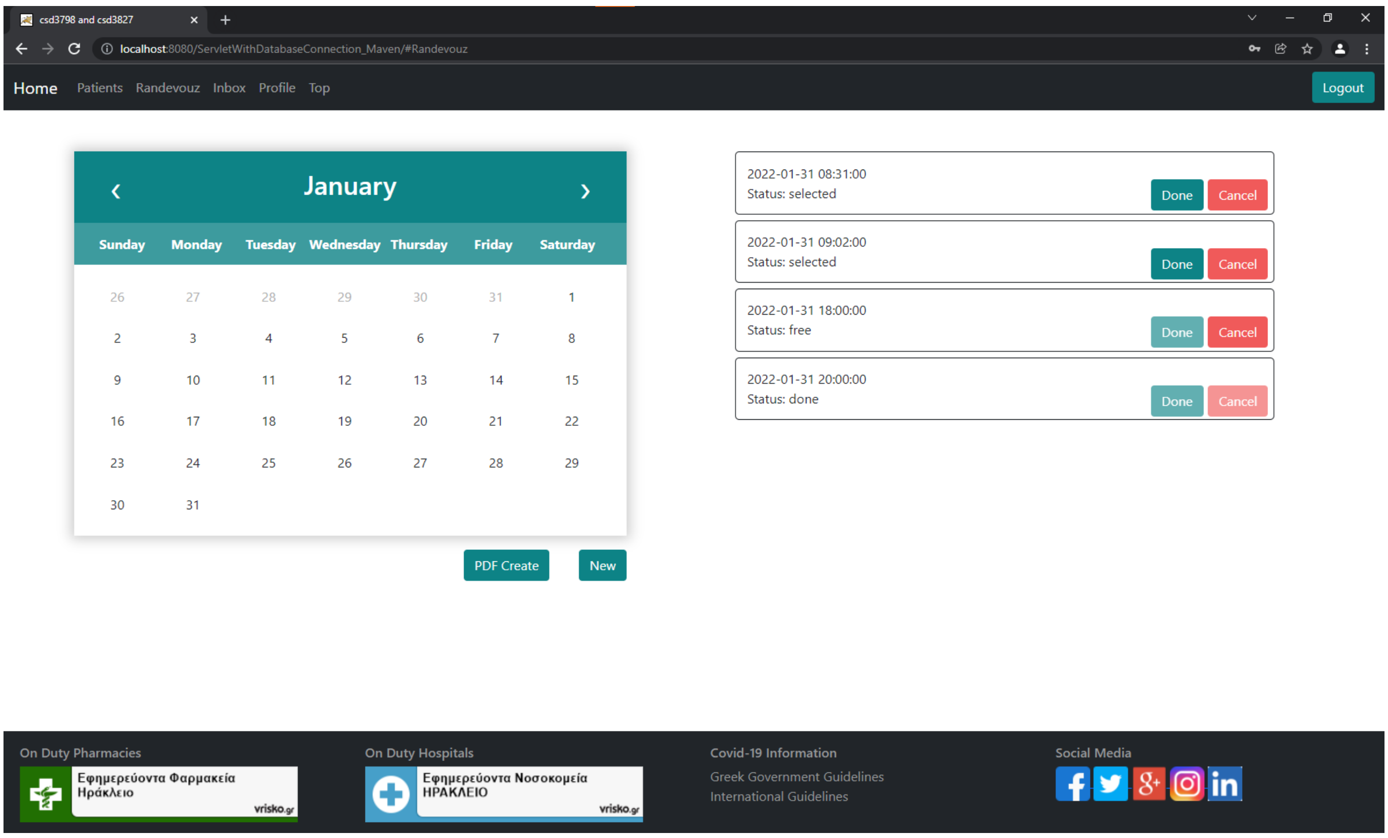


Date	Name	Message
2021-10-11 15:11:00	Michalis Mountanton	Den eimai kala

Main divs/tags :

1. <div id ="mainPart" >
   Uses bootstrap for spacing
2. <div id ="seeMsgs"></div>
   Holds the table of messages

- **Randevouz**



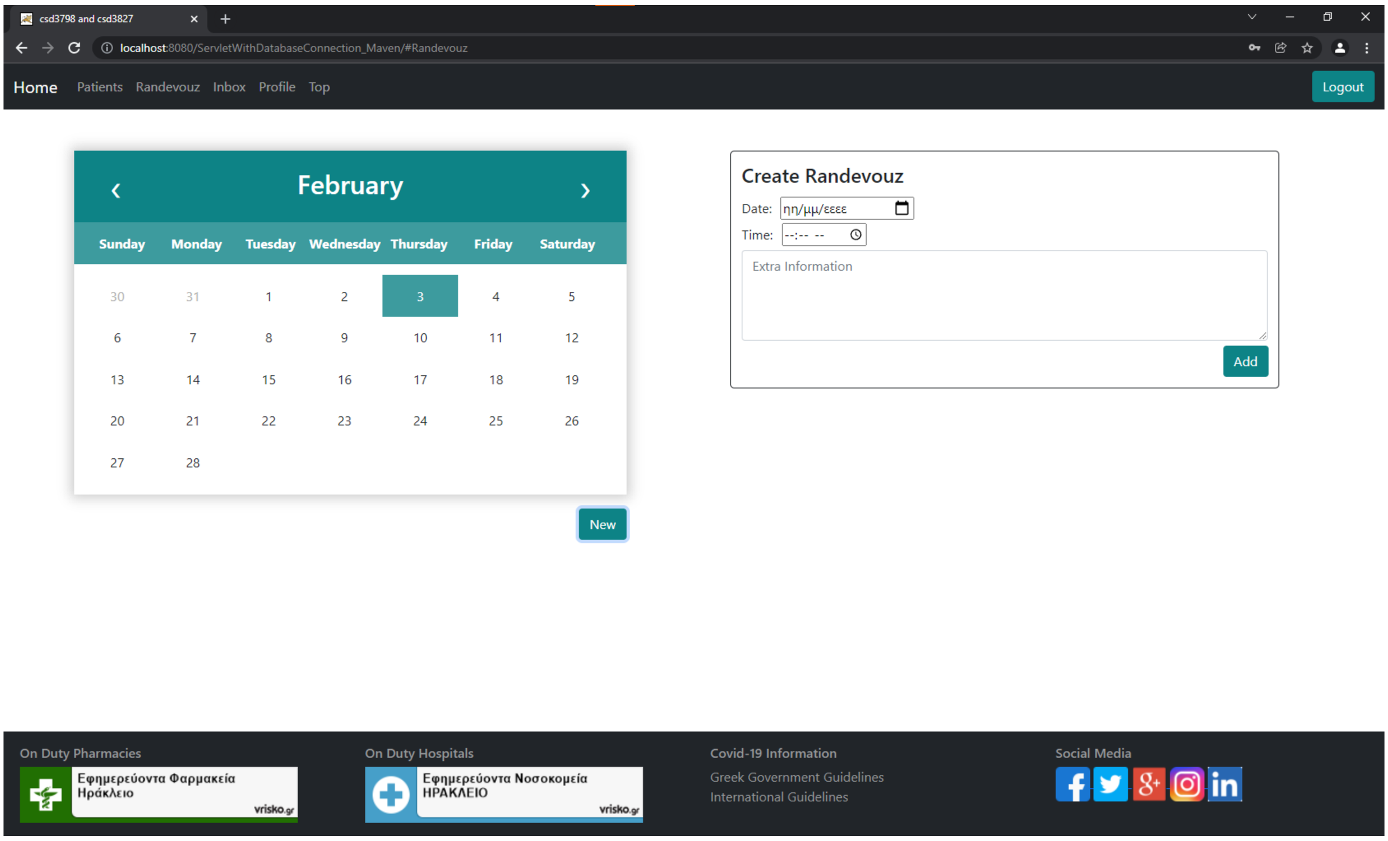
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

2022-01-31 08:31:00
Status: selected

2022-01-31 09:02:00
Status: selected

2022-01-31 18:00:00
Status: free

2022-01-31 20:00:00
Status: done



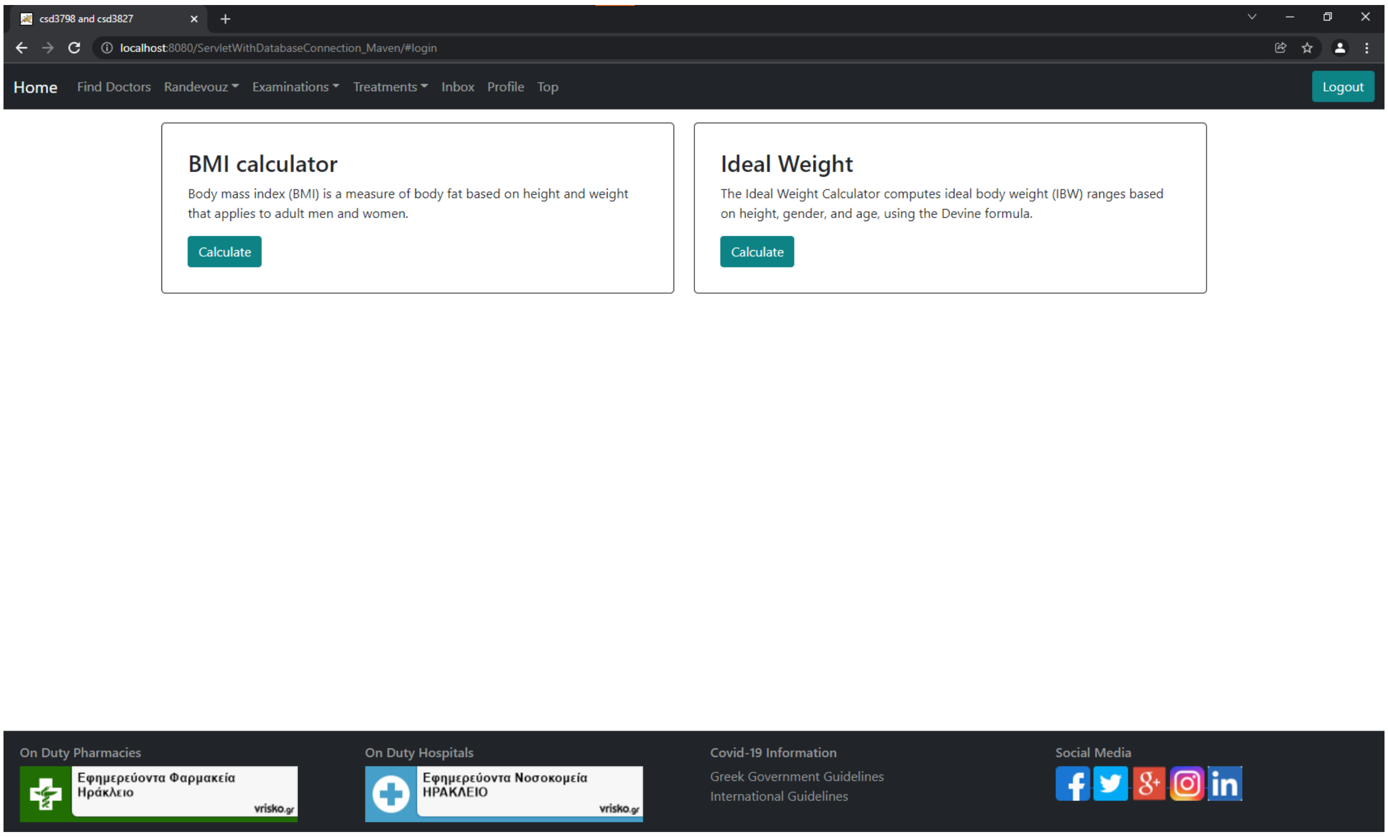
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28					

Create Randevouz
Date: ηη/μμ/εεεε
Time: --:-- --
Extra Information

Main divs/tags :

1. <div id ="mainPart" >
2. <div id="calendar">
3. <div id = "randevouz"> is the parent to <div id="createRand">

- **User Homepage**



BMI calculator

Body mass index (BMI) is a measure of body fat based on height and weight that applies to adult men and women.

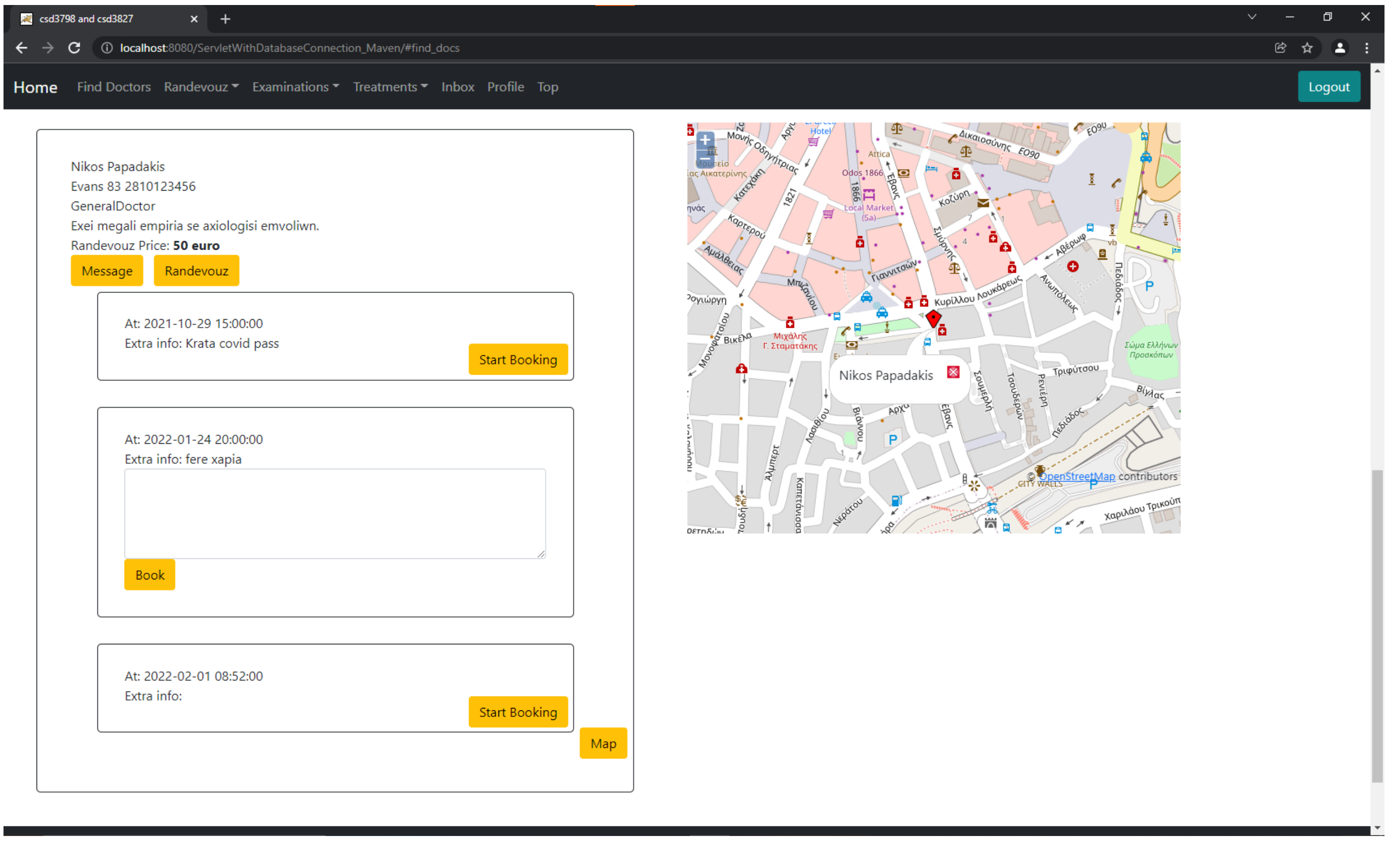
Ideal Weight

The Ideal Weight Calculator computes ideal body weight (IBW) ranges based on height, gender, and age, using the Devine formula.

Main divs/tags :

1. <div class = "container">
   Uses bootstrap to split page for the following divs (is parent)
2. <div id = "BMI">
3. <div id = "Ideal_weight">

- **User Find Doctors**


csd3798 and csd3827
localhost:8080/ServletWithDatabaseConnection_Maven/#find_docs
Home Find Doctors Randevouz Examinations Treatments Inbox Profile Top Logout

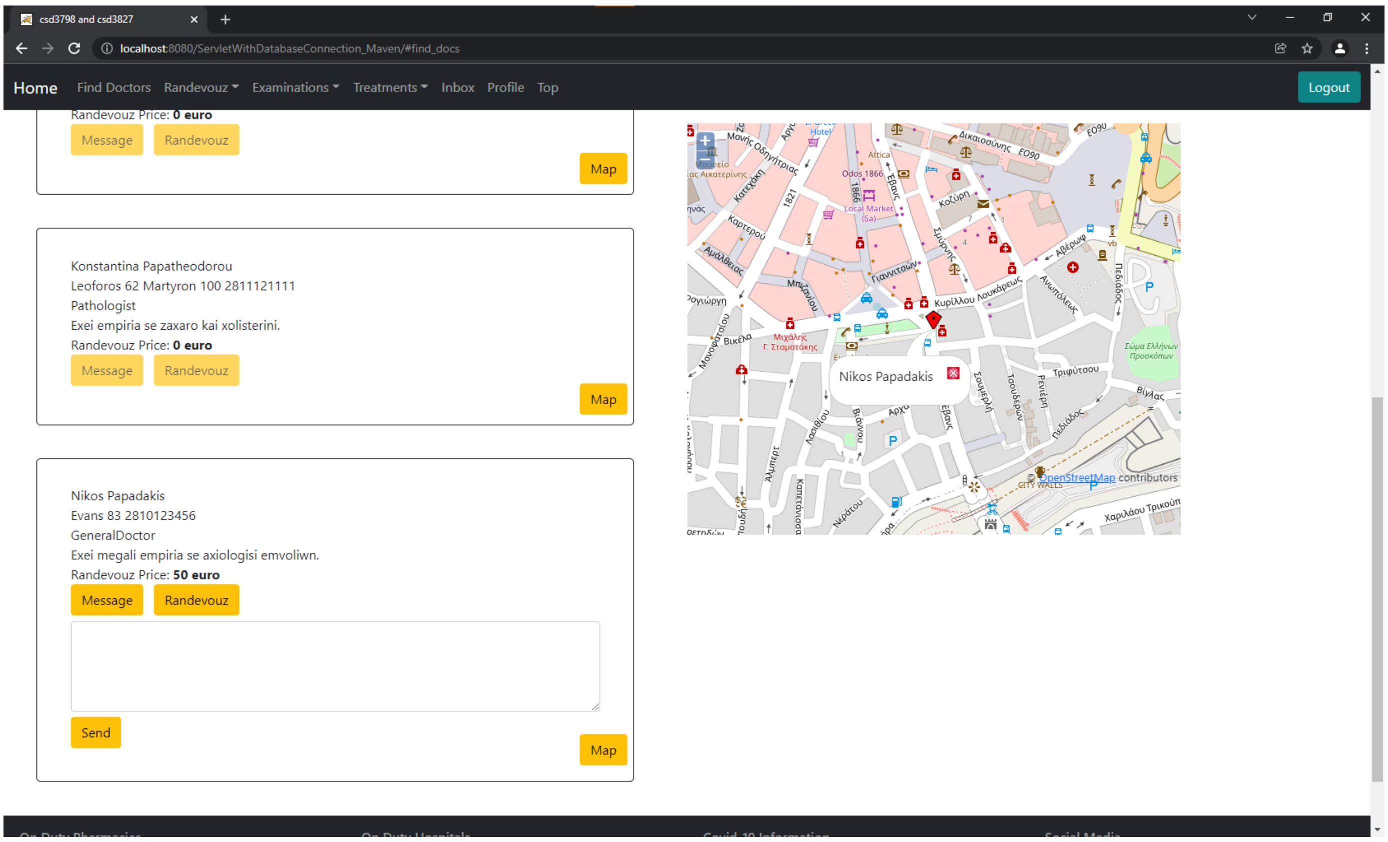
Nikos Papadakis
Evans 83 2810123456
GeneralDoctor
Exei megali empiria se axiologisi emvoliwn.
Randevouz Price: **50 euro**
Message Randevouz

At: 2021-10-29 15:00:00
Extra info: Krata covid pass
Start Booking

At: 2022-01-24 20:00:00
Extra info: fere xapia
Book

At: 2022-02-01 08:52:00
Extra info:
Start Booking

Map


csd3798 and csd3827
localhost:8080/ServletWithDatabaseConnection_Maven/#find_docs
Home Find Doctors Randevouz Examinations Treatments Inbox Profile Top Logout

Randevouz Price: **0 euro**
Message Randevouz
Map

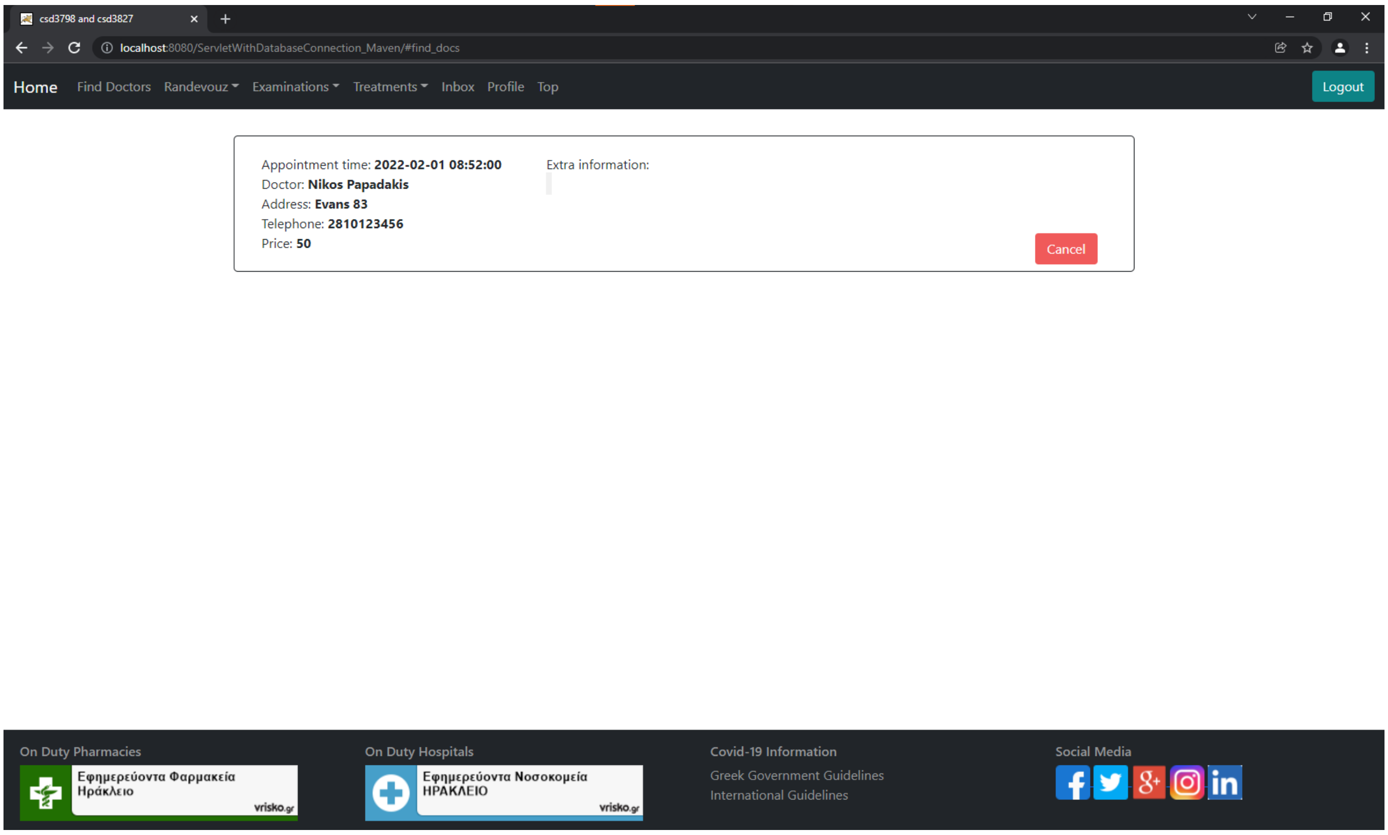
Konstantina Papatheodorou
Leoforos 62 Martyron 100 2811121111
Pathologist
Exei empiria se zaxaro kai xolisterini.
Randevouz Price: **0 euro**
Message Randevouz
Map

Nikos Papadakis
Evans 83 2810123456
GeneralDoctor
Exei megali empiria se axiologisi emvoliwn.
Randevouz Price: **50 euro**
Message Randevouz
Send
Map

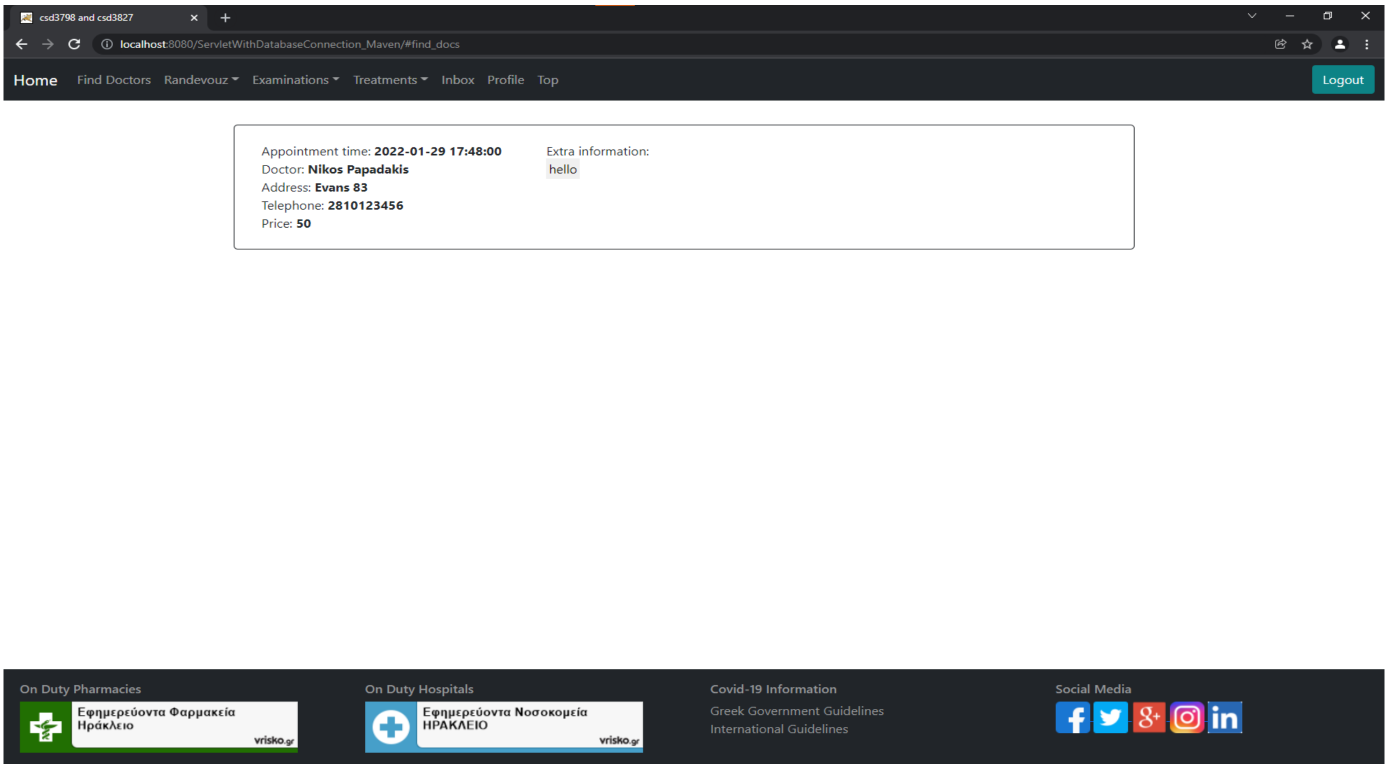
Main divs/tags :

1. Has the same main divs as Find Doctors (Guest user)
2. <div class = “doc_buttons” id = “docID”>
   Parent to <div class="extra_buttons" id="EB_docID"> has all the necessary buttons to book an appointment.

- **Randevouz (user)**



Appointment time: **2022-02-01 08:52:00**
Doctor: **Nikos Papadakis**
Address: **Evans 83**
Telephone: **2810123456**
Price: **50**
Extra information:

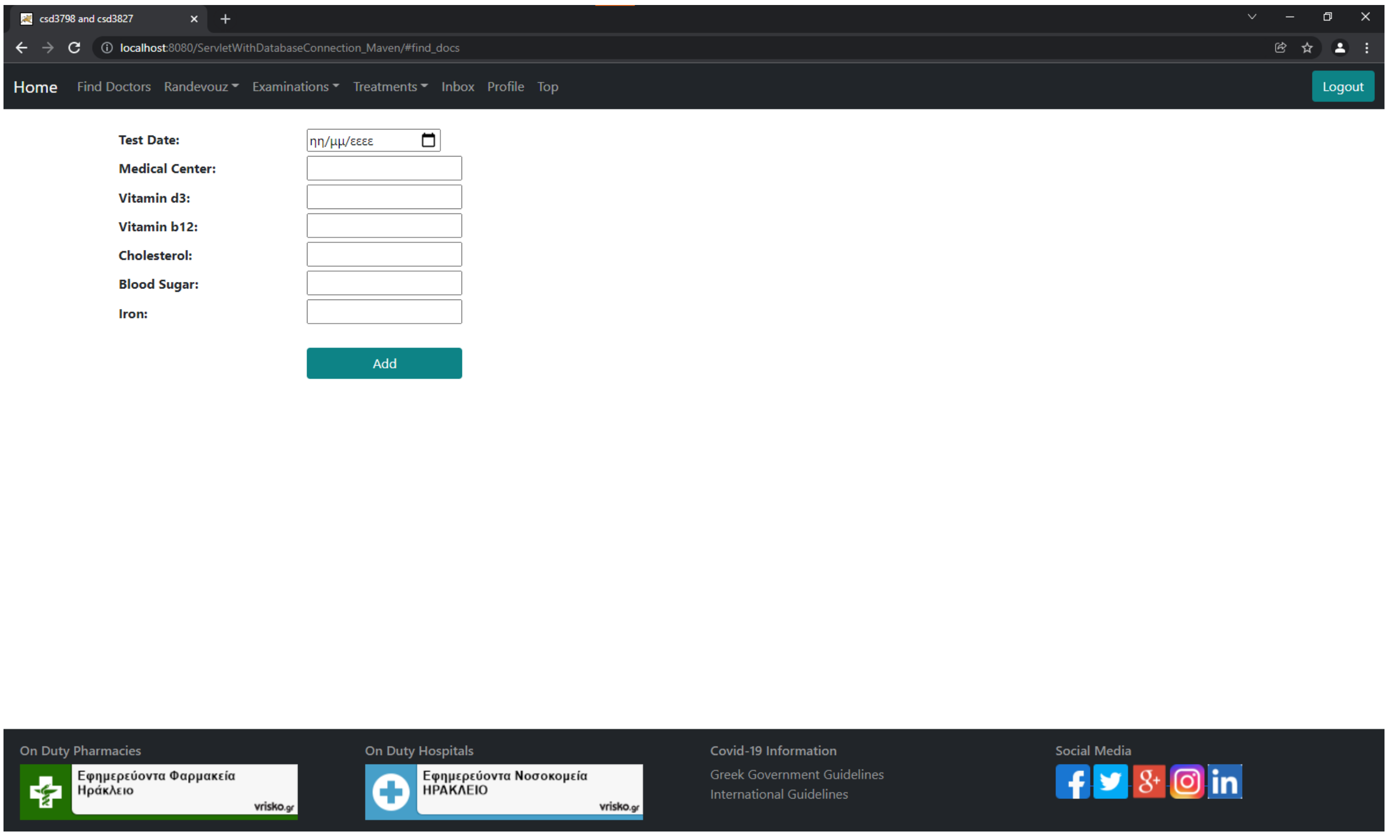


Appointment time: **2022-01-29 17:48:00**
Doctor: **Nikos Papadakis**
Address: **Evans 83**
Telephone: **2810123456**
Price: **50**
Extra information:
hello

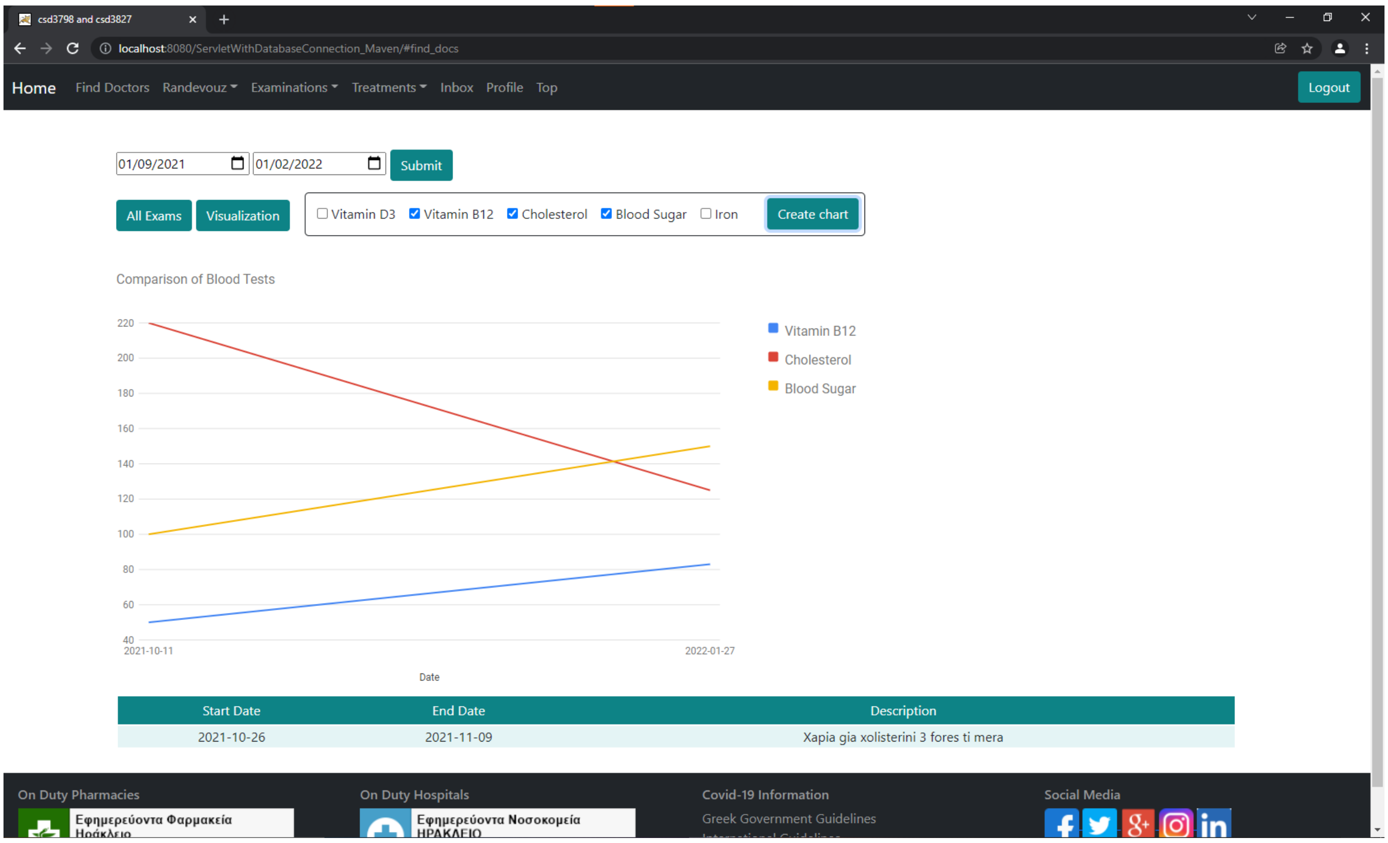
Main divs/tags :

1. <div class="pendingUserRands">
   Holds the information of the rendezvous

- **Examinations**



Test Date:	ηη/μμ/εεεε
Medical Center:	
Vitamin d3:	
Vitamin b12:	
Cholesterol:	
Blood Sugar:	
Iron:	

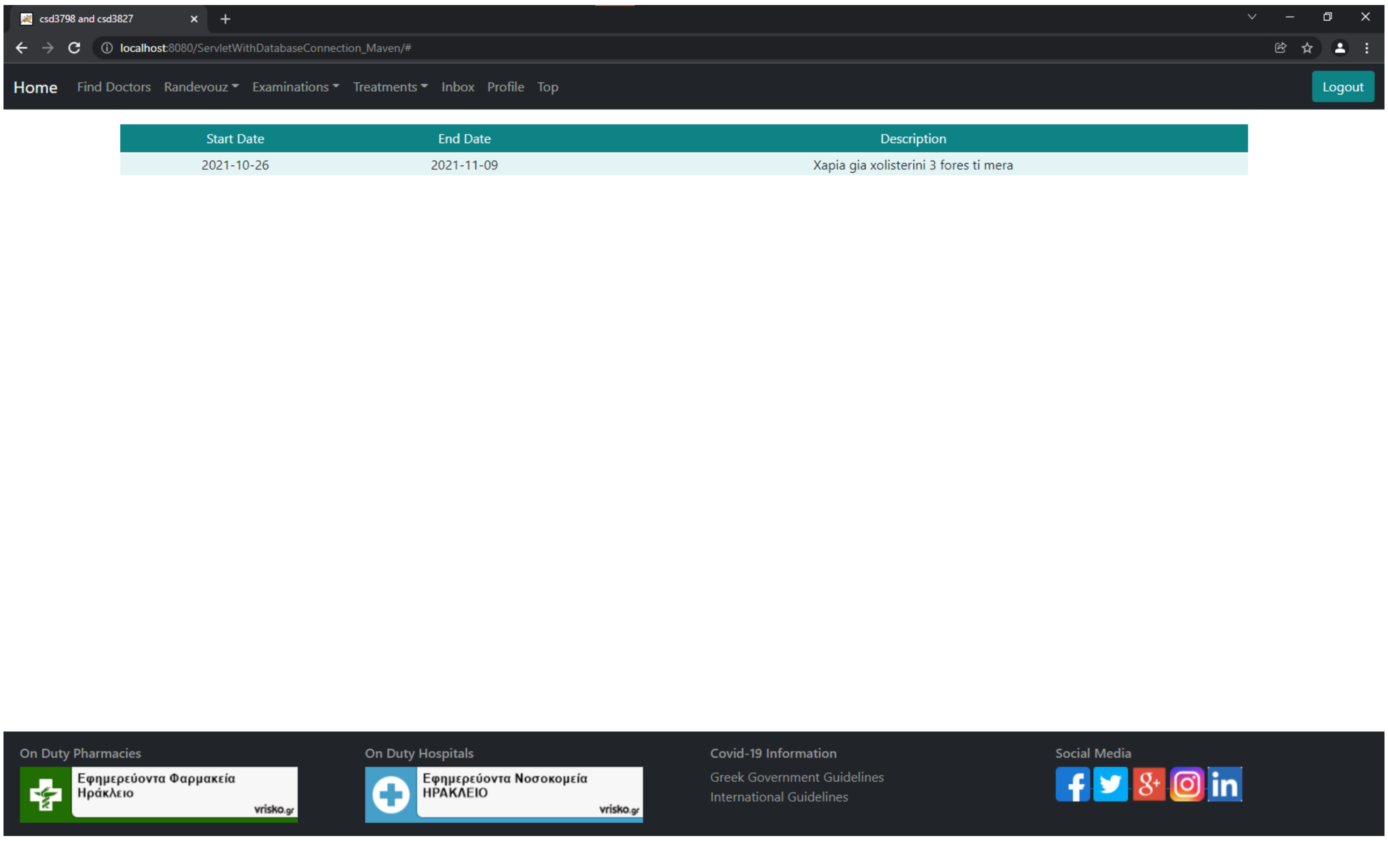


Start Date	End Date	Description
2021-10-26	2021-11-09	Xapia gia xolisterini 3 fores ti mera

Main divs/tags :

Same as patient treatments.

- **Treatments**

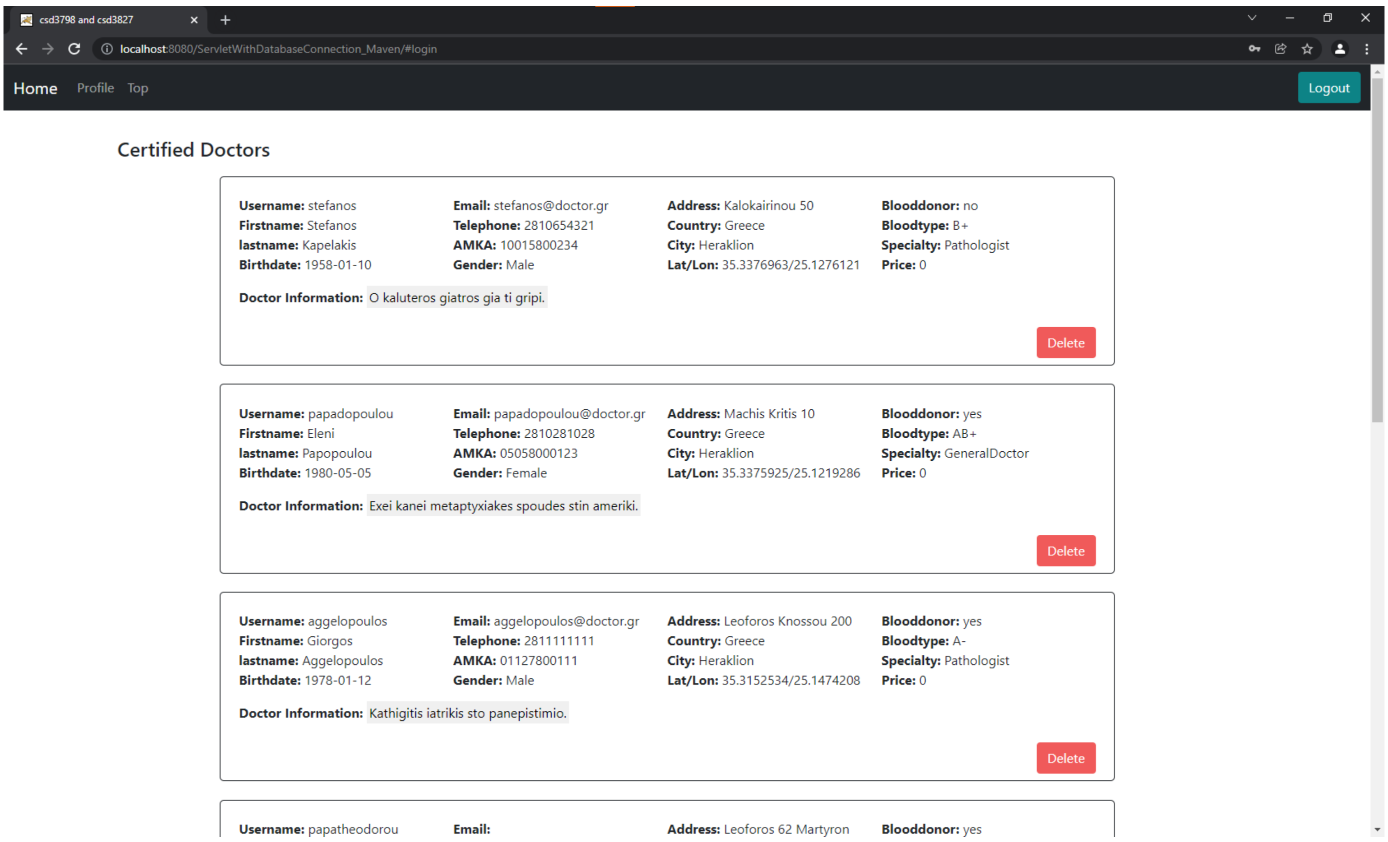


Start Date	End Date	Description
2021-10-26	2021-11-09	Xapia gia xolisterini 3 fores ti mera

Main divs/tags :

1. <div id="seeTreatments">
2. <table id="treattable">

- ## Admin Homepage



Certified Doctors

Username: stefanos	**Email:** stefanos@doctor.gr	**Address:** Kalokairinou 50	**Blooddonor:** no
Firstname: Stefanos	**Telephone:** 2810654321	**Country:** Greece	**Bloodtype:** B+
lastname: Kapelakis	**AMKA:** 10015800234	**City:** Heraklion	**Specialty:** Pathologist
Birthdate: 1958-01-10	**Gender:** Male	**Lat/Lon:** 35.3376963/25.1276121	**Price:** 0

Doctor Information: O kaluteros giatros gia ti gripi.

Delete

Username: papadopoulou	**Email:** papadopoulou@doctor.gr	**Address:** Machis Kritis 10	**Blooddonor:** yes
Firstname: Eleni	**Telephone:** 2810281028	**Country:** Greece	**Bloodtype:** AB+
lastname: Papopoulou	**AMKA:** 05058000123	**City:** Heraklion	**Specialty:** GeneralDoctor
Birthdate: 1980-05-05	**Gender:** Female	**Lat/Lon:** 35.3375925/25.1219286	**Price:** 0

Doctor Information: Exei kanei metaptyxiakes spoudes stin ameriki.

Delete

Username: aggelopoulos	**Email:** aggelopoulos@doctor.gr	**Address:** Leoforos Knossou 200	**Blooddonor:** yes
Firstname: Giorgos	**Telephone:** 2811111111	**Country:** Greece	**Bloodtype:** A-
lastname: Aggelopoulos	**AMKA:** 01127800111	**City:** Heraklion	**Specialty:** Pathologist
Birthdate: 1978-01-12	**Gender:** Male	**Lat/Lon:** 35.3152534/25.1474208	**Price:** 0

Doctor Information: Kathigitis iatrikis sto panepistimio.

Delete

Username: papatheodorou	**Email:**	**Address:** Leoforos 62 Martyron	**Blooddonor:** yes

Main divs/tags :

1. <div id ="uncertifiedDocs">
   Holds the boxes with all the uncertified doctors
2. <div id ="certifiedDocs">
   Holds the boxes with all the uncertified doctors
3. <div id ="allUsers">
   Holds the boxes with all the users

Chapter 6

Other Functionalities

Servlets: 7 servlets in total, each with either a doGet or a doPost, the exception being Login.java that has both.

1. **EditUser**(POST) : Receives as a request a json containing a Signed User/ Doctor that has the new edited information for that user. Checks whether the user that is logged in is a Signed User or a Doctor, and then gets the relevant information for said user. This is achieved by reading the session cookies. It then compares the new email, the only editable information that is also a key in the database for every other Signed User's and Doctor's email. If there is a match, the request is rejected, otherwise the Signed User/Doctor gets updated with the new info.
2. **FindDocs**(GET) : Returns in json format all the information for certified doctors.
3. **Login**

   (GET) : Returns the information of the logged user/admin/ doctor in json format, uses cookies to figure out who he is.

   (POST) : tries to match input password, username to an account within the database, If it finds it, returns a success message and writes a cookie that other servlets can use to see what the user is, does not allow uncertified doctors to login.
4. **Logout**(POST) : invalidates the cookies set up by Login(POST).
5. **PdfMaker**(GET) : sees that it has both query parameters before moving on, makes a request at the database for the input day's rendezvous and writes them to a pdf, ignoring the cancelled rendezvous.
6. **Register**(POST) : Checks whether the username, email, or amka is in use by any existing user, doctor. If not, add the user to the database.
7. **RegisterDoctor**(POST) : Checks whether the username, email, or amka is in use by any existing user, doctor. If not, add the doctor to the database.

REST API

@GET functions in use

1. **/bloodtest/{amka} :** returns bloodtests of said amka, can have fromDate & toDate as part of its Query.
2. **/treatments/{user_id}:** get treatments for said user id, can have fromDate & toDate as part of its Query.
3. **/CanMessage/{user_id}/{doctor_id}:** checks if user and doctor have at least one done rendezvous, returns yes or no.
4. **/FreeRandevouz/{doctor_id}:** returns doctor's free rendezvous.

5. **/pendingRandevouz/{id}/{type}:** returns user's selected rendezvous if type=user or doctor's selected rendezvous if type=doctor.
6. **/allRandevouz/{id}/{type}:** returns all of user's rendezvous if type=user or all of doctor's rendezvous if type=doctor.
7. **/pendingRandevouz/{id}/{type}:** returns user's done rendezvous if type=user or doctor's done rendezvous if type=doctor.
8. **/docDoneRandevouz/{id}:** returns all user ids that have done a rendezvous with the doctor with input id.
9. **/doctor/{id}:** returns doctor with matching id.
10. **/uncertified:** returns all uncertified doctors.
11. **/users:** returns all users.
12. **/user/{id}:** returns user with input id.
13. **/cancelledRandevouz/{id}/{type}:** returns user's cancelled rendezvous if type=user or doctor's cancelled rendezvous if type=doctor.
14. **/inbox/{id}/{type}:** returns user's inbox if type=user or doctor's inbox if type=doctor.

@POST functions in use

1. **/newBloodTest:** checks if input is valid, if yes, adds bloodtest.
2. **/newTreatment:** checks if input is valid, if yes, adds treatment.
3. **/newMessage:** checks if input is valid, if yes adds message with current date.
4. **/newRandevouz:** checks if input is valid, if yes, adds randevouz..

@POST functions in use

1. **/upadateRandevouz/{randevouz_id}/{user_id}/{user_info}/{status}:** if input is valid, updates randevouz with user's input.
2. **/upadateRandevouz/{randevouz_id}/{user_id}//{status}:** same as above for empty user_info.
3. **/updateRandevouz/{randevouz_id}/{status}:** turn a randevouz into cancelled, selected, free, or done (change to free and selected not used with this)
4. **/certifyDoctor/{doctorID}:** updates a doctor into certified;

@DELETE functions in use

1. **/userDelete/{userID}:** deletes user with input id.
2. **/doctorDelete/{doctorID}:** deletes doctor with input id.
3. **/RandevouzDeletion/{RandevouzID}:** deletes randevouz with input id.