Eötvös Loránd University Faculty of

Informatics Department of Media and

Educational Informatics

Dochouse

Supervisor:

Illés Zoltán

Associate Professor - PhD, habilitation

Author:

Sadi Mamedov

Computer Science BSc

Budapest, 2020

Contents

Chapter 1

Introduction

Over the past decades web development has been reached to advanced improvements, growing its influence as a proper solution and environment to build real world applications. Since the software development period, applications used to require specific software and hardware requirements only to serve features under the local machine environment. After the introduction of web services, several desktop applications have been migrated to the web environment for the reasons that it provides platform independence, device accessibility, and demanding few system requirements.

1.1 Background

Nowadays, information technology plays a significant role in human daily life to such an extent that enables us to manage time efficiently. Moreover, using digital applications we can solve issues related to public services. Traditional Health care services are considered one of the most time consuming, crowdy and nerve-racking for both doctors and patients due to long waiting queues, registering patients, scheduling workflow, unavailability of staff members, etc. On the other hand, It's very difficult to find professional doctors based on specific medical fields. Consulting with a doctor is one of the common activities in society, whereas due to aforementioned problems we consider it tedious practice that most of the time leads to ignoring our health issues.

1.2 Solution to the problem

Since web platforms are accessible from various types of devices including laptops, tablets, mobile phones, considering the large audience and necessity of this problem, I decided to build the application over the web surface. By accessing the homepage of this application, patients may benefit from making a medical appointment with qualified doctors, also from receiving answers to their health-related questions through written communication. At the same time, doctors may register to the trusted platform and continue to provide their services to the patients in need. The application intended to serve two sorts of user audience: patients and doctors, who are required to register in order to use services, whereas, guests will be able to collect general details and information about the provided facilities.

Chapter 2

User Documentation

There will be two different portals with distinct interfaces corresponding to the user type. Through the documentation, patient type users will be referred as ‘users’ , while doctor type users will be referred as ‘doctors’. Usage of the program consists of following functionalities:

1. Common interface:
   All kinds of users including unauthorized users referred to as ‘guests’ can register, navigate to the login transition page and homepage of the application. After the registration, the type of user is determined.
2. User interface:
   Users may sign in, sign out, search doctor, edit user profile, navigate to the doctor’s profile, make an appointment request, delete settled requests, review doctor, make payment, and initiate communication via chat.
3. Doctor interface:
   Doctors may sign in, sign out, edit doctor profile, accept a request, reject a request, delete settled requests, join communication via chat.

Some system requirements must be satisfied in order to use the application.

2.1 System requirements

Devices must have network access having bandwidth greater than 50Kb for page size. Also, from hardware minimum 2 GB of RAM, and 64-bit environment 2 GHz dual processor on MS-Windows (7 and later), MacOS (versions 10.2 and later) , latest Linux (64 bit) distributions, for displaying standard resolution of (1024 x 768) VGA are required.

2.1.1 Software requirements

All modern browsers support react-built web applications. For better performance latest releases of Chrome, Mozilla or Safari recommended. In order to run the client and server side application Node package manager (npm) must be installed in the operating system.

2.2 Installation process

In order to run the application in local environment follow instructions below:

1. Copy the following URL of github public repository of the project.
   https://github.com/MoneiBall/DocHouse.git
2. On your local machine, clone the repository to the working directory using
   'git clone <https-URL>' command.
3. Navigate to the cloned repository, and install all dependencies for the server application using the 'npm install' command.
4. Navigate to the client repository using 'cd doc-house'. Install all dependencies for the client application using 'npm install' command.
5. To test both client and server applications run 'npm test' command, otherwise skip this step.
6. Navigate back to the root repository, and run 'npm start' which will run both client and server application concurrently.
7. If there have not occurred prior errors until this step, npm successfully will redirect you to 'Dochouse' homepage on your default browser.

To review live demo version of application follow the URL below:

https://heroku.com/MoneiBall/DocHouse

2.3 Common Interface Guidelines

In this part of the chapter, a detailed explanation of the common user interface will be presented using figures. The common interface can be accessed by all kind of users after triggering the client side application.

2.3.1 Home page

After starting the program, the homepage of the website will appear in your default browser. Users may obtain general information about the application and its services. One can review all the details using component scroller. With the help of navigation bar , users may proceed to login transition page (top right corner), and by clicking Menu button (top left corner) to open Sidebar which provides navigation to the other pages (**Figure 2.2**).

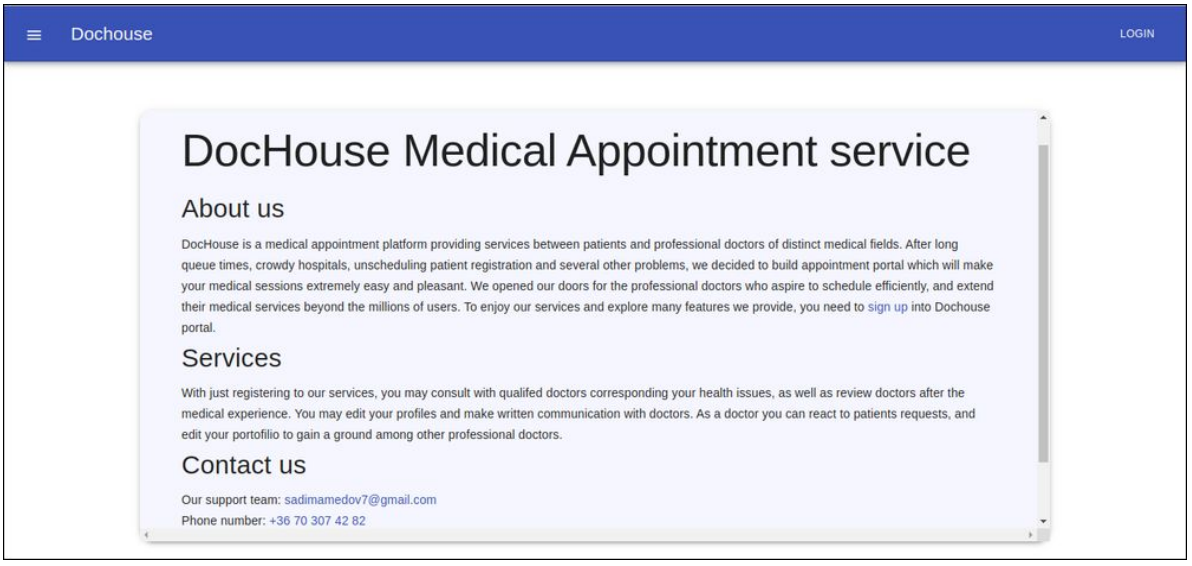

Figure 2.2: Home page of the website

Pages, other than those which are not allowed to access from unauthorized users are explorable from the home page.

2.3.2 Login Transition page

Transition page provides account options for users and doctors. By clicking on the named buttons, one will be redirected to the dedicated interface in order to register and sign in to the application **(Figure 2.3.1)**.

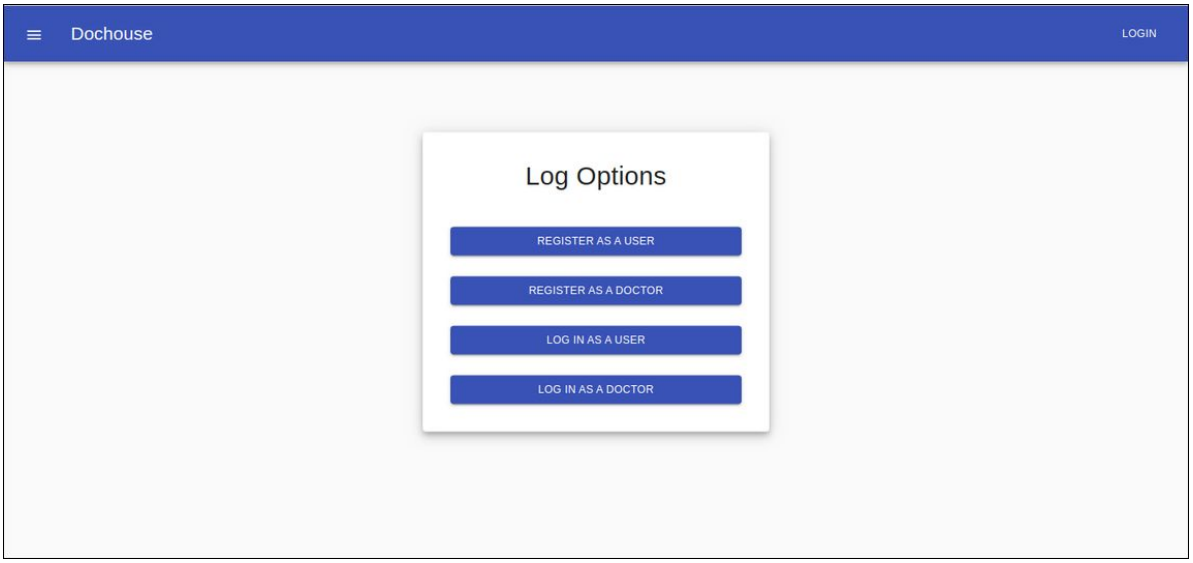

Figure 2.3.1: Login transition page of the website

2.3.3 User and Doctor Registration pages

After pressing on *Register as a user* button depicted on **(Figure 2.3.1),** users lead to the registration page. Users must fill out required fields and then submit the registration form. To continue further , users will be tempted to enter their data until all the validation rules are satisfied. In case valid data inserted, within the form snackbar appears illustrating successful registration. As the next step, By following the snackbar link, users may sign in to the application **(Figure 2.3.21)**

Similar to users, doctors are also required to register by filling out their personal data. Validation rules will apply until correct standard input is provided, and then a successful registration indicator will pop up **(Figure 2.3.22)**. The provided data by user and doctor, will be used in the authentication process to identify log entities.

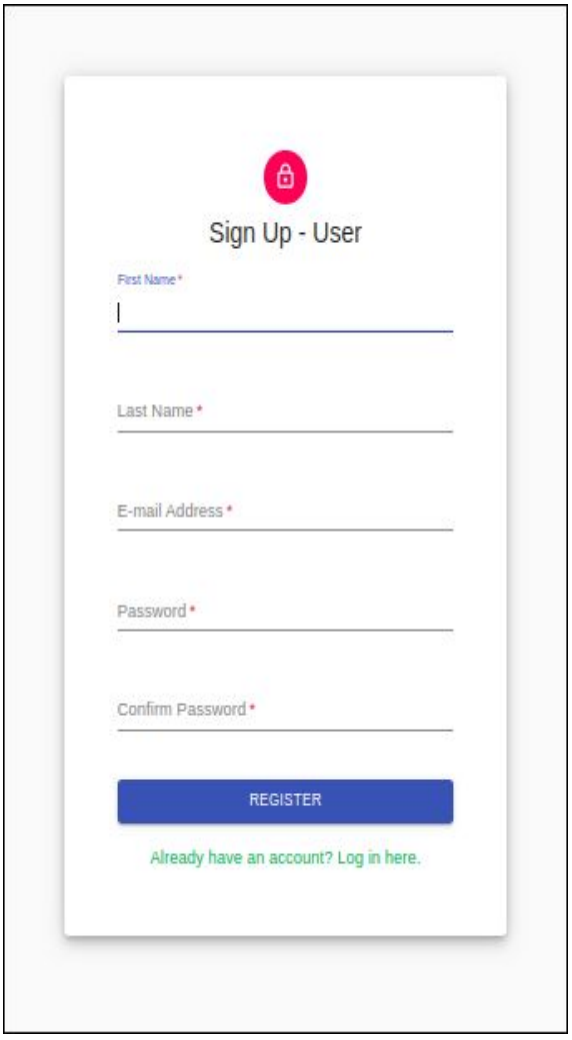

Figure 2.3.21: User sign up component

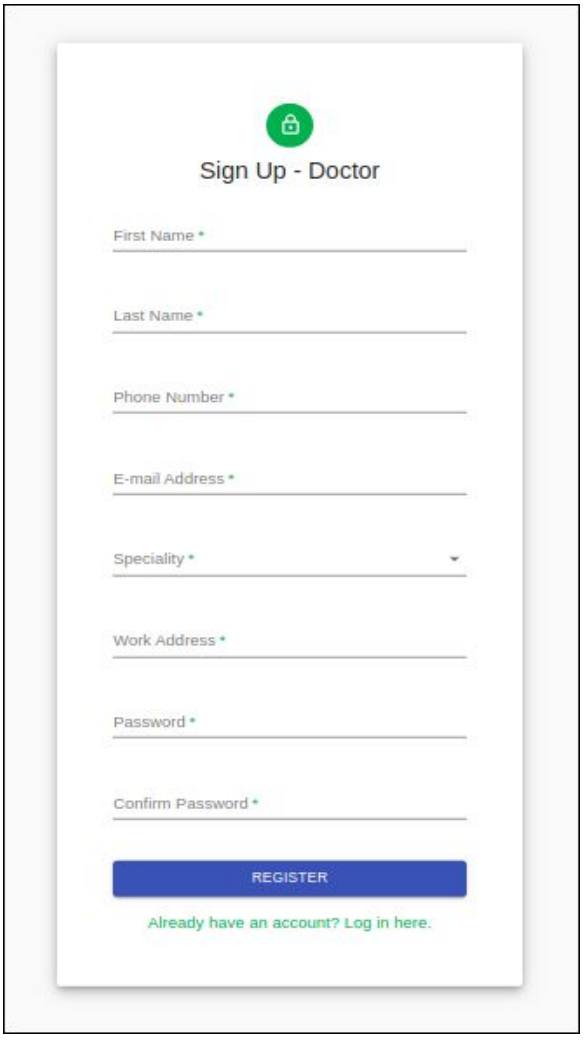

Figure 2.3.22: Doctor sign up component

2.3.4 User and Doctor Login pages

By following any of the routes to login pages mentioned in previous parts of chapter, users and doctors can access the corresponding login pages. To continue with the sign in process, the program requires account credentials those of which are acquired in the registration process. Thus, only authorized users and doctors may have account sessions. After entering credentials and pressing *Log In* button, program will redirect to profile page **(Figure 2.3.31, Figure 2.3.32).**

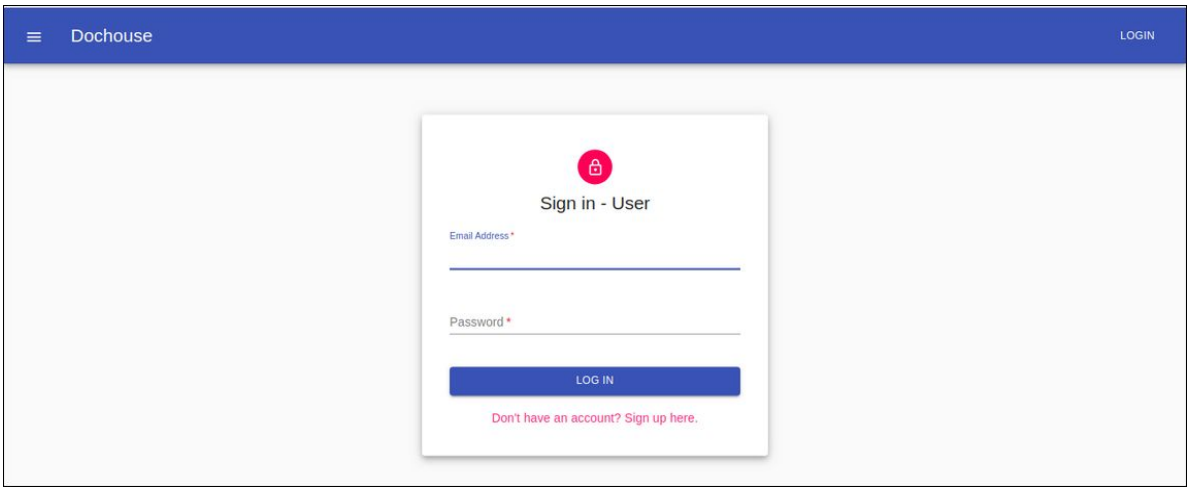

Figure 2.3.31: User sign in page

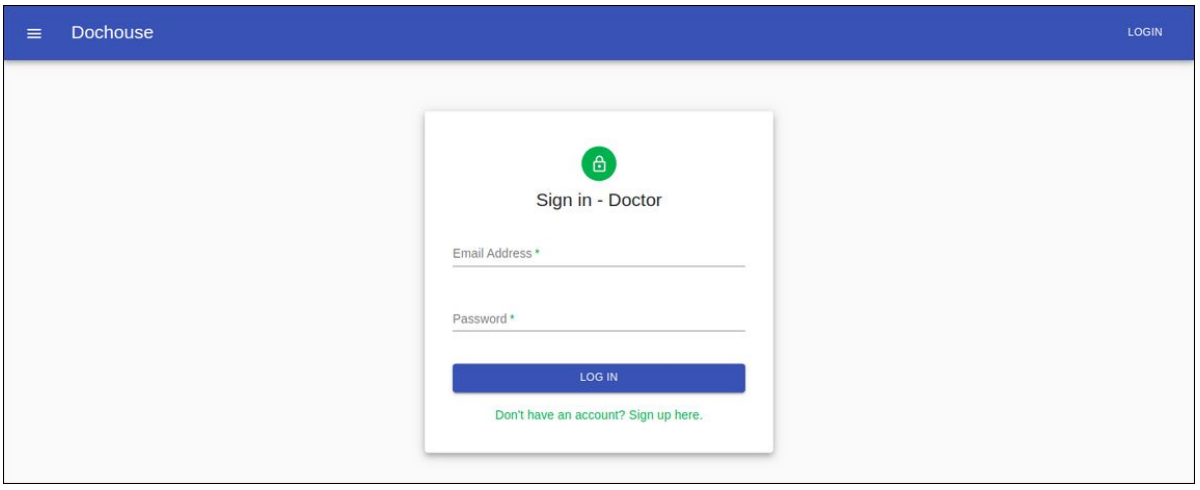

Figure 2.3.32: Doctor sign in page

2.4 User Interface Guidelines

2.5 Doctor Interface Guidelines

Chapter 3

Developer Documentation

The Developer Documentation (or Developer's Manual) should contain

- the detailed specification of the problem,

- the detailed description of the used methods, the definitions of the used no- tions,
- the description of the logical and physical structure of the software (data structures, databases, modules etc.),
- the testing plan and the results of the tests.

Bibliography

[1] Somebody, something.