



University of Debrecen

Faculty of Informatics, Computer Science Engineering

Online Doctor Appointments Web Application (FindDoc)

Candidate name: Hesham Ibrahim

Supervisor: Dr. Adamkó Attila

Debrecen, 2022

Table of content

1. Introduction	4
2. Technologies	6
2.1. Client side technologies	6
2.1.1. React.js	6
2.1.2. Styled-components	7
2.2. Server side technologies.....	7
2.2.1. Node.js.....	8
2.2.2. MongoDB	8
2.2.3. Mongoose	9
2.3. Third party services	9
2.3.1. Cloudinary	9
2.3.2. PubNub.....	10
2.3.3. MongoDB Atlas	10
2.3.4. EmailJS.....	10
3. Database models	11
3.1. User model	11
3.2. Appointment model	12
3.3. Review model.....	14
3.4. Specialty model	15
4. User interface and features	17
4.1. Guest features	17
4.1.1. Home page	17

4.1.1.1. Home section	18
4.1.1.2. Service section	19
4.1.1.3. Doctors section	20
4.1.1.4. Contact section	21
4.1.2. Register.....	22
4.1.3. Login	23
4.2. Patient features	24
4.2.1. Overview	24
4.2.2. Doctors	25
4.2.3. Bookings.....	28
4.2.4. Appointments.....	29
4.2.5. Profile	33
4.3. Doctor features	35
4.3.1. Overview	36
4.3.2. Bookings.....	37
4.3.3. Appointments.....	38
4.3.4. Reviews	40
4.3.5. Profile	41
5. Conclusion.....	43
6. Bibliography	44

1. Introduction

One of the global industries with the fastest growth rates is health care. Prior to the last few years, appointments for medical care were typically made over the phone or by physically visiting the hospitals. The availability of schedulers, phone lines, or the physical presence of a person were the only ways to accept appointments because this process required the involvement of people. Everyone demanded enduring and effective medical care delivery as time went on because manual appointments which require both individuals to be physically present and lengthy waiting lines created a frustrating situation for healthcare facilities. Because of this, there is a need for an integrated health care system that can provide seamless care to both inpatients and outpatients.

The objective of this web application was to create a system where patients are able to easily discover the right doctor and schedule an appointment, while at home or anywhere. The second objective was to also assist doctors in better managing their practices and developing their internet reputation. This application also aims to enable effective communication between patients and doctors while also enabling the patients' ease of access and comfort. It also seeks to solve issues that patients encounter when scheduling appointments and maintaining medical records. On the basis of doctor's professional profile and reviews from previous patients, patients can select the right doctor for their medical requirements.

Using well-known technologies and programming languages, the application was successfully implemented. The user interface of this application was kept simple and user-friendly to help patients easily find what they're looking for, moreover, help both doctors and patients easily join a live chat appointment, and manage their appointments using the application's dashboard.

This application is a full stack application with a client side(frontend) and a server side(backend), built using the MERN technology stack (What Is The MERN Stack? Introduction & Examples | MongoDB).

Basic functionality has been put in place for the time being, but work will be done in the future to improve users medical experience to the best.

2. Technologies

In order to have a better understanding of the application, it is important to know what technologies and programming languages have been used to implement the application, and what makes these technologies a good choice to build this application with.

To provide a thorough understanding of the application, each technology, programming language and its use are described below in detail.

2.1. Client side technologies

In web development, 'client side' refers to everything in a web application that is displayed or takes place on the client (end user device). This includes what the user sees, such as text, images, and the rest of the UI, along with any actions that an application performs within the user's browser (What do client side and server side mean? | Client side vs. server side).

The client side of this application is developed using React.js, and styled using styled-components.

2.1.1. React.js

React.js is a front-end library that has gradually become the go-to framework for modern web development within the JavaScript community. React.js is an open-source JavaScript framework

and library developed by Facebook. It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript ([What is React.js? \(Uses, Examples, & More\)](#)).

In React, you develop your applications by creating reusable components that you can think of as independent Lego blocks. These components are individual pieces of a final interface, which, when assembled, form the application's entire user interface.

2.1.2. Styled-Components

Using tagged template literals and arrow functions in ES6+ and CSS, styled-components is a React-specific CSS-in-JS styling solution that creates a platform for developers to write actual CSS code to style React components ([How To Use React Styled Components in React Efficiently](#)).

Using styled-components makes it simple to use actual CSS to style components by writing JavaScript code. These components are known as “styled components” which are, in actuality, a React component with styles.

2.2. Server side technologies

The client side communicates with the server side using Hyper Text Transfer Protocol (HTTP) ([HTTP | MDN](#)). When a user clicks a link on a web page, submit a form, or run a search, an HTTP request is sent from your browser to the target server.

The server side of this application is developed using Node.js, and connected to MongoDB Atlas using Mongoose (Express Tutorial Part 3: Using a Database (with Mongoose)).

2.2.1. Node.js

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications (Node.js | beecrowd).

Node.js is basically used as an open-source and cross platform JavaScript runtime environment, for running the server side applications, for building the I/O intensive applications like video streaming sites, online chatting applications and many other applications, and it is perfect for data-intensive applications since it uses an asynchronous, event-driven model. Node.js operates on a single-thread, allowing it to handle thousands of simultaneous event loops.

2.2.2. MongoDB

MongoDB is a scalable and flexible document database. MongoDB's document model is simple for developers to learn and use, while still providing all the capabilities needed to meet the most complex requirements at any scale (MongoDB: The Developer Data Platform | MongoDB).

MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time (surazova / mongoDB: MongoDB stores data in flexible, JSON-like ...). MongoDB is also a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use.

2.2.3. Mongoose

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB (Mongoose ODM v6.8.1).

2.3. Third party services

To improve the efficiency of the application development process, and increases the quality of the service, certain functionalities in this application were implemented using third-party services that are built by experts.

The main third party services that are used in this application are Cloudinary, PubNub, MongoDB Atlas, and EmailJS.

2.3.1. Cloudinary

Cloudinary is an end-to-end image- and video-management solution for websites and mobile apps, covering everything from image and video uploads, storage, manipulations, optimizations to delivery (Cloudinary Image & Video Management - Documentation).

Cloudinary is used in this application to host media uploaded by users.

2.3.2. PubNub

PubNub supports use cases including chat, virtual events, collaboration, alerting, IoT device control, real-time updates, and push notifications to help companies build engaging, scalable, and reliable apps. Learn how PubNub can help you build better virtual spaces ([About PubNub, The Real-time Communication Platform | PubNub](#)).

PubNub is used in this application enable 1-on-1 chatting between patients and doctors.

2.3.3. MongoDB Atlas

MongoDB Atlas is a multi-cloud database service by the same people that build MongoDB ([What is MongoDB Atlas? — MongoDB Atlas](#)). Atlas simplifies deploying and managing your databases while offering the versatility users need to build resilient and performant global applications on the cloud providers of the user's choice.

2.3.4. EmailJS

EmailJS is a JavaScript library that helps send emails using only client-side technologies. The cool thing about EmailJS is that no servers are required; all you have to do is connect it to one of the supported email services, create an email template, and use EmailJS to trigger an email.

EmailJS is used in this application to allow users to send emails even in case the server failed to work, since EmailJS allows email delivery from the client-side (Sending Emails From React With EmailJS).

3. Database models

The database of this application has 4 models: user, appointment, review, and specialty model. These models are created and managed using mongoose.

3.1. User model

The user model is used to handle two roles of users, that is patients and doctors.

The user model consists of 5 fields:

- email: this field is required, must be unique, and of 'String' type.
- username: this field is required, and of 'String' type.
- password: this field is required, and of 'String' type.
- role: this field is required, and of 'String' type.
- profileImage: this field is optional, and of 'String' type.

The 'profileImage' field must be of 'String' type because images are hosted on Cloudinary, and each hosted image has a URL string which is used to view users' images. The 'role' field stores whether the user is a doctor or a patient.



```
1  const mongoose = require('mongoose');
2
3  const User = new mongoose.Schema(
4    {
5      email: {type: String, required: true, unique: true},
6      username: {type: String, required: true},
7      password: {type: String, required: true},
8      role: {type: String, required: true},
9      profileImage: {type: String}
10   }, {
11     collection: 'user'
12   }
13 )
14
15 const model = mongoose.model('user', User);
16
17 module.exports = model;
```

3.2. Appointment model

The appointment model consists of 5 fields:

- booked_for: this field is required, and of 'String' type.
- booked_by: this field is required, and of 'String' type.
- date: this field is required, and of 'String' type.
- reason: this field is required, and of 'String' type.
- confirmed: this field is required, and of 'Boolean' type.

The 'booked_for' field stores the doctor id, and the 'booked_by' field stores the patient id. The 'reason' field stores the reason that the patient booked the appointment for. As for the 'confirmed' field, it's used to check whether the appointment has been confirmed by the doctor that the patient booked for.


```
1  const mongoose = require('mongoose');
2
3  const Appointment = new mongoose.Schema(
4    {
5      booked_for: {type: String, required: true},
6      booked_by: {type: String, required: true},
7      date: {type: String, required: true},
8      reason: {type: String, required: true},
9      confirmed: {type: Boolean, required: true}
10   }, {
11     collection: 'appointment'
12   }
13 )
14
15 const model = mongoose.model('appointment', Appointment);
```

3.3. Review model

The review model consists of 4 fields:

- reviewed_by: this field is required, and of 'String' type.
- review_for: this field is required, and of 'String' type.
- review: this field is required, and of 'String' type.
- rating: this field is required, and of 'Number' type.

The 'reviewed_by' field stores the patient id that submitted a doctor review, and the 'review_for' field stores the doctor id that was reviewed. The 'review' field stores the patient review text. The 'rating' field stores the rating of the doctor given by the patient.



```
1  const mongoose = require('mongoose');
2
3  const Review = new mongoose.Schema(
4    {
5      reviewed_by: {type: String, required: true},
6      review_for: {type: String, required: true},
7      review: {type: String, required: true},
8      rating: {type: Number, required: true},
9    }, {
10     collection: 'review'
11   }
12 )
13
14 const model = mongoose.model('review', Review);
15
```

3.4. Specialty model

The specialty model is used to store the specialty of doctors.

The specialty model consists of 2 fields:

- `doctor_id`: this field is required, must be unique, and of 'String' type.
- `specialty`: this field is required, and of 'String' type.

The 'doctor_id' stores the id of the doctor that the 'specialty' field belongs to. The 'specialty' field stores the doctor's specialty / field of expertise.



```
1  const mongoose = require('mongoose');
2
3  const Specialty = new mongoose.Schema(
4    {
5      doctor_id: {type: String, required: true, unique: true},
6      specialty: {type: String, required: true}
7    }, {
8      collection: 'specialty'
9    }
10 )
11
12 const model = mongoose.model('specialty', Specialty);
```


4. User interface and features

The user interface and features of each actor in this application will be explained in detail in this section. This application has 3 actors: the guest, patient, and doctor. Each actor has their own features and some features are shared between more than one actor.

The user interface for this application is responsive on all screens (desktop, tablet, and mobile screen).

4.1. Guest features

The guest actor is a user that is not registered, or a public user with very limited features. The guest can access 3 pages: the home (landing), login, and a register page.

4.1.1. Home page

The home page is the landing page that is first displayed when a user opens the application. The home page is made of 4 sections: home, service, doctors, and contact.

4.1.1.1. Home section

This section is an introduction about the application. Along with the introduction this section includes a 'Get Started' button which redirects the user to the registration page.



[Home](#) [Service](#) [Doctors](#) [Contact](#)

[Log in](#) [Register](#)

Get better connected to healthcare.

Find a doctor, make an appointment, and clear up any health concerns. We want people to be able to easily discover the right doctor and schedule an appointment.

We also assist doctors in better managing their practices and developing their internet reputation.

[Get Started](#)



4.1.1.2. Service section

In the section, the service provided by the application for patients is explained in 4 simple steps. The service section also includes the application's mission with two buttons, a 'Get Started' button which redirects the user to the registration page, and a 'Get in Touch' button that scrolls down the page to the 'Contact' section when the button is clicked.

FindDoc Home Service Doctors Contact Log in Register

How FindDoc works?

- 1 Register**
 To get started, click 'Register' in the navigation bar and fill in the required fields.
- 2 Find a Doctor**
 Begin your search for the ideal doctor for your requirements.
- 3 Book an Appointment**
 Select a day and time for your appointment and then click 'Book Appointment.'
- 4 Join Live Consultation**
 Once your appointment has been set, join the live chat consultation on the appointed day.

Our mission
Simplifying healthcare needs.

Our goal is to make access to high-quality care simple, friendly and transparent for consumers. We are building a world where we can all access healthcare that is convenient and affordable.

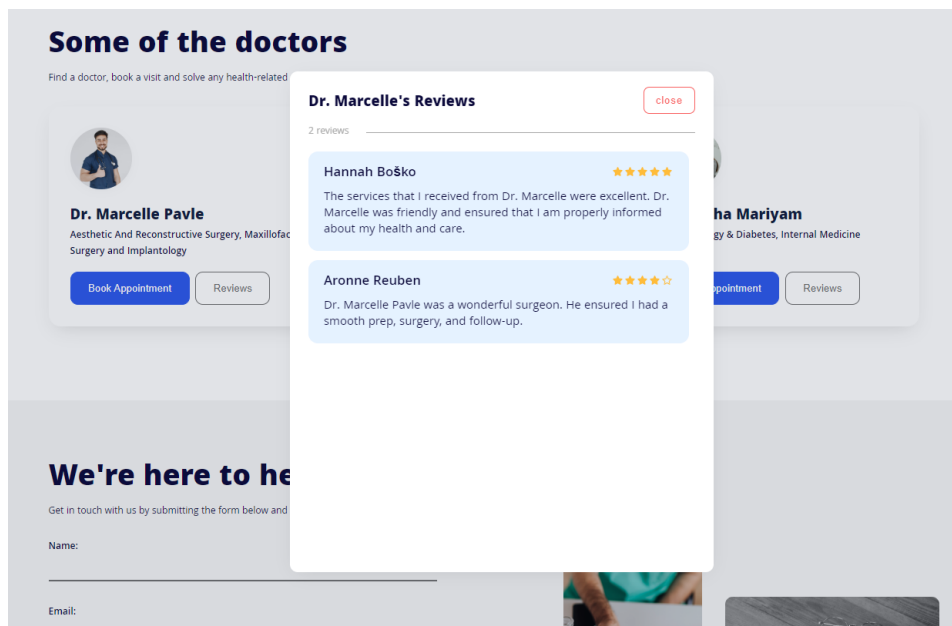
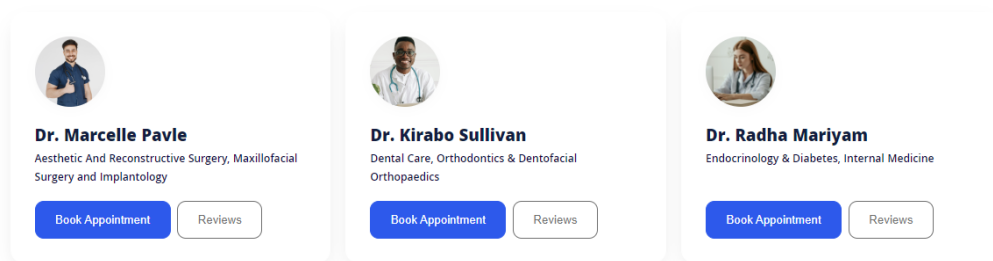
[Get Started](#) [Get in Touch](#)

4.1.1.3. Doctors section

This section displays a list of doctors. Each doctor is displayed in a container that includes the doctor's username, doctor's specialty, and two buttons 'Book Appointment' and 'Reviews'. The 'Book Appointment' redirects the user to the login page, since the user needs to be logged in to book a doctor. The 'Reviews' button displays a list of the doctor's reviews.

Some of the doctors

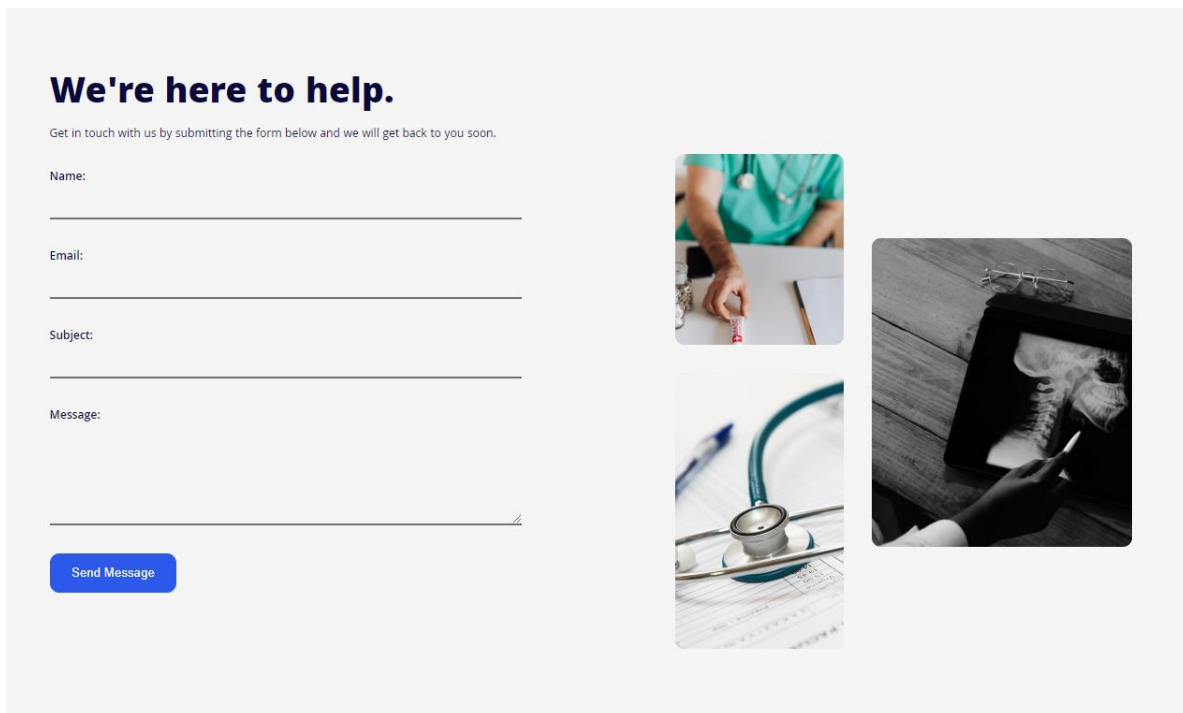
Find a doctor, book a visit and solve any health-related doubt. Log in to your account to find more doctors.



4.1.1.4. Contact section

Users can submit their feedback, request support, or send any type of message using the contact form in this section.

The user first needs to fill in the required fields (Name, Email, Subject, Message), then click the ‘Send Message’ button. Once the user clicks ‘Send Message’, if the data is valid, an http post request will be made from the client side to the backend (server side). Then the backend handles the request and sends the user’s message to server’s registered email.



We're here to help.

Get in touch with us by submitting the form below and we will get back to you soon.

Name:

Email:

Subject:

Message:

[Send Message](#)

The form is part of a contact section. To the right of the form are three images: a doctor's hands in green scrubs, a stethoscope on a white surface, and a hand pointing at a screen displaying a medical image.

The email is sent using a library called EmailJS. EmailJS email delivery is secured using TLS/STARTTLS.

4.1.2. Register

For a user to register, the user needs to fill in the required registration form fields. The registration form includes the following fields: Profile Image (optional), Select Role, Email, Username, Password, Confirm Password, and a 'Create Account' button.

[Log in](#)[Register](#)

Register

Profile Image

Choose File No file chosen

Select role

☐ Patient ☐ Doctor

Email

example@email.com

Username

test@gmail.com

Password

Confirm password

Create Account

[Already have an account?](#)

After the user clicks 'Create Account', the form data is sent to the backend and validated, if the data is valid and the email does not already exist in the database then the user is registered in the database. After the user is registered, a success response is sent back to the client side and the user is redirected to the dashboard.

4.1.3. Login

The user can login by selecting their role then filling in their account email and password then clicking the 'Login' button.

[Log in](#)[Register](#)

Login

Select role

☐ Patient ☐ Doctor

Email

test@gmail.com

Password

Login

[Forgot your password?](#)

[Don't have an account?](#)

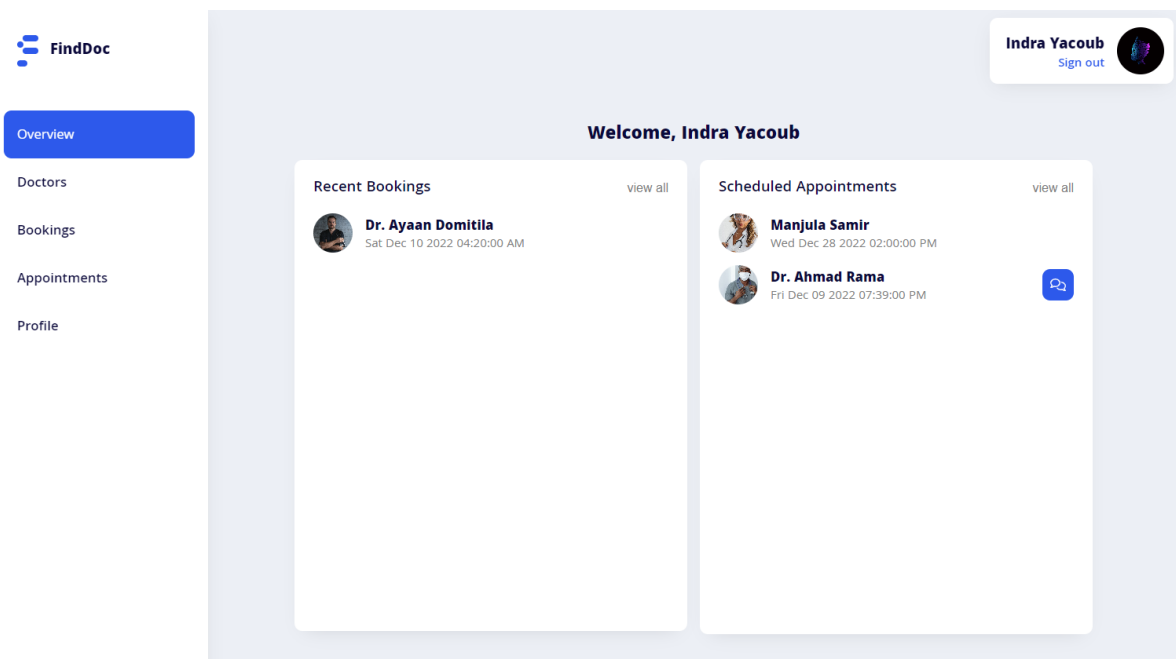
After that the data is sent to the backend for authentication, if the data is valid and the user is found in the database, a success response is sent back to the client side then the user is authenticated and redirected to the dashboard.

4.2. Patient features

When users registered as a 'Patient' role login to the application, the user is redirected to the patient's dashboard. The patient's dashboard includes 5 pages: Overview, Doctors, Bookings, Appointments, and Profile page.

4.2.1. Overview

The overview page is the first page that the patient joins after logging in. This page shows an overview of the bookings made by the patient and the scheduled appointments.

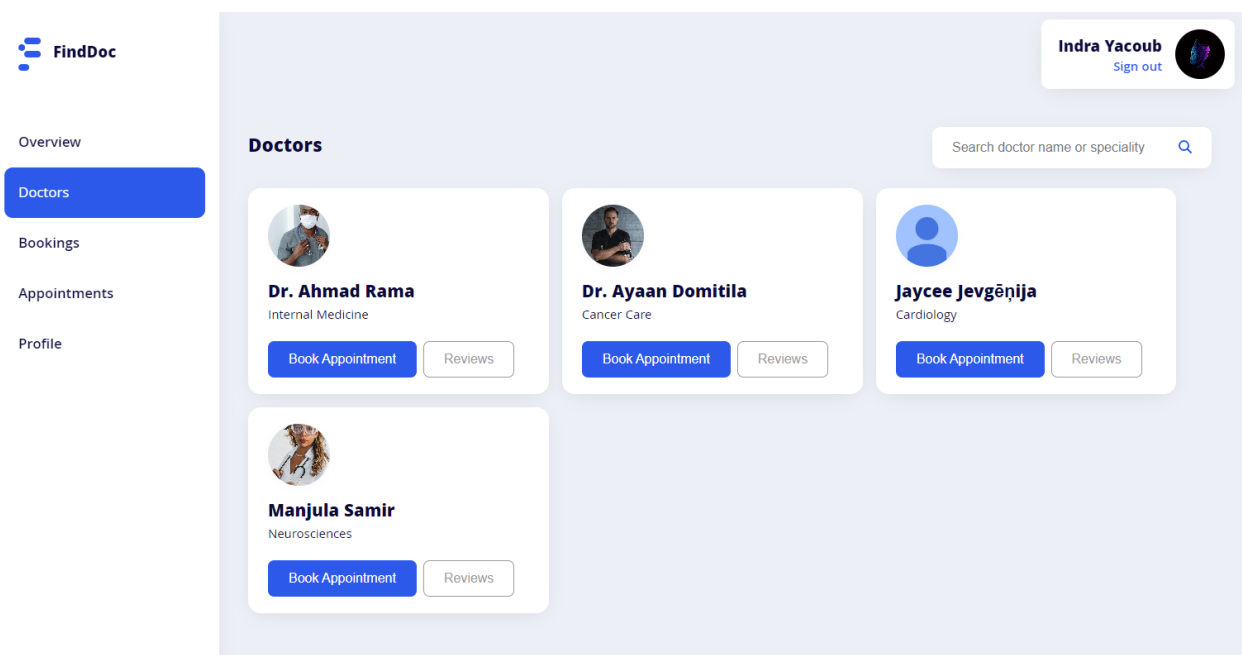


When the page first loads, the patient's bookings and appointments are retrieved from the database and displayed in two containers 'Recent Bookings', and 'Scheduled Appointments' container.

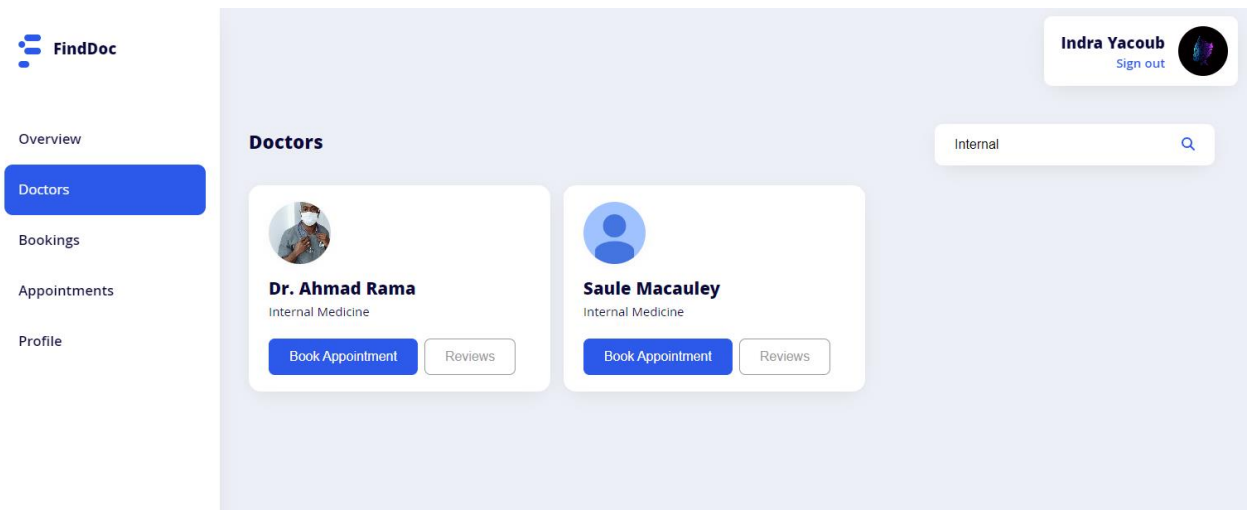
If an appointment has started, a button that allows the user to join the appointment chat session will be shown to the right side of the specified appointment row.

4.2.2. Doctors

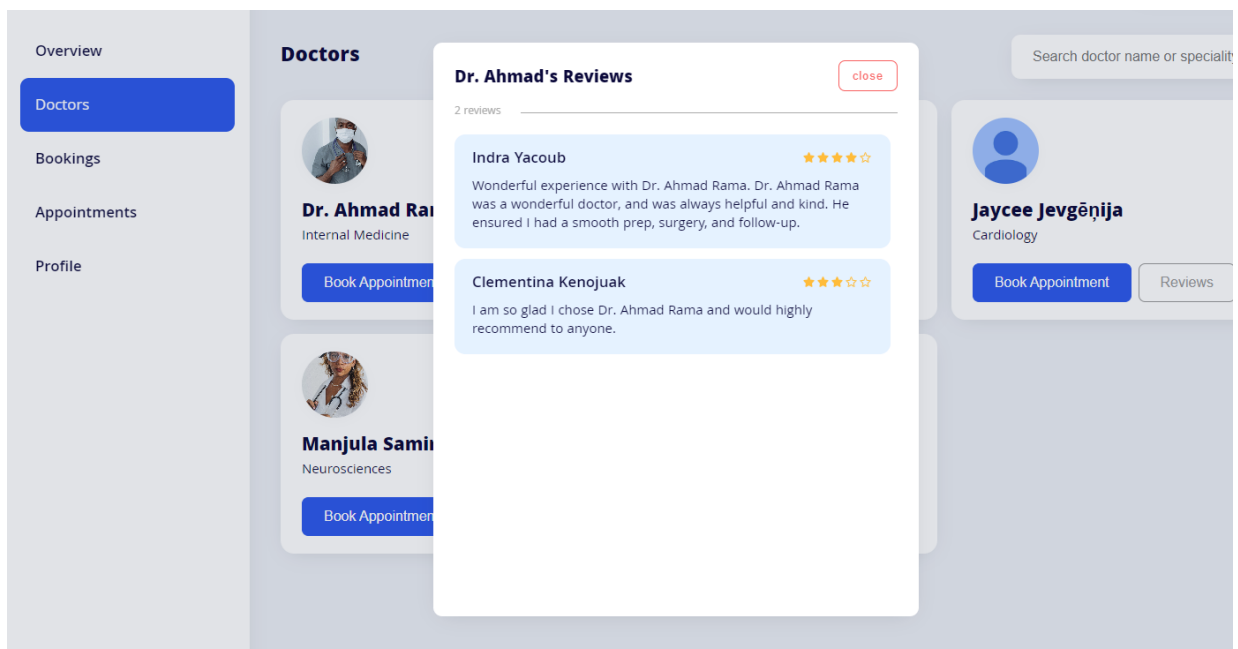
All the registered doctors who completed their account setup will be displayed in this page.



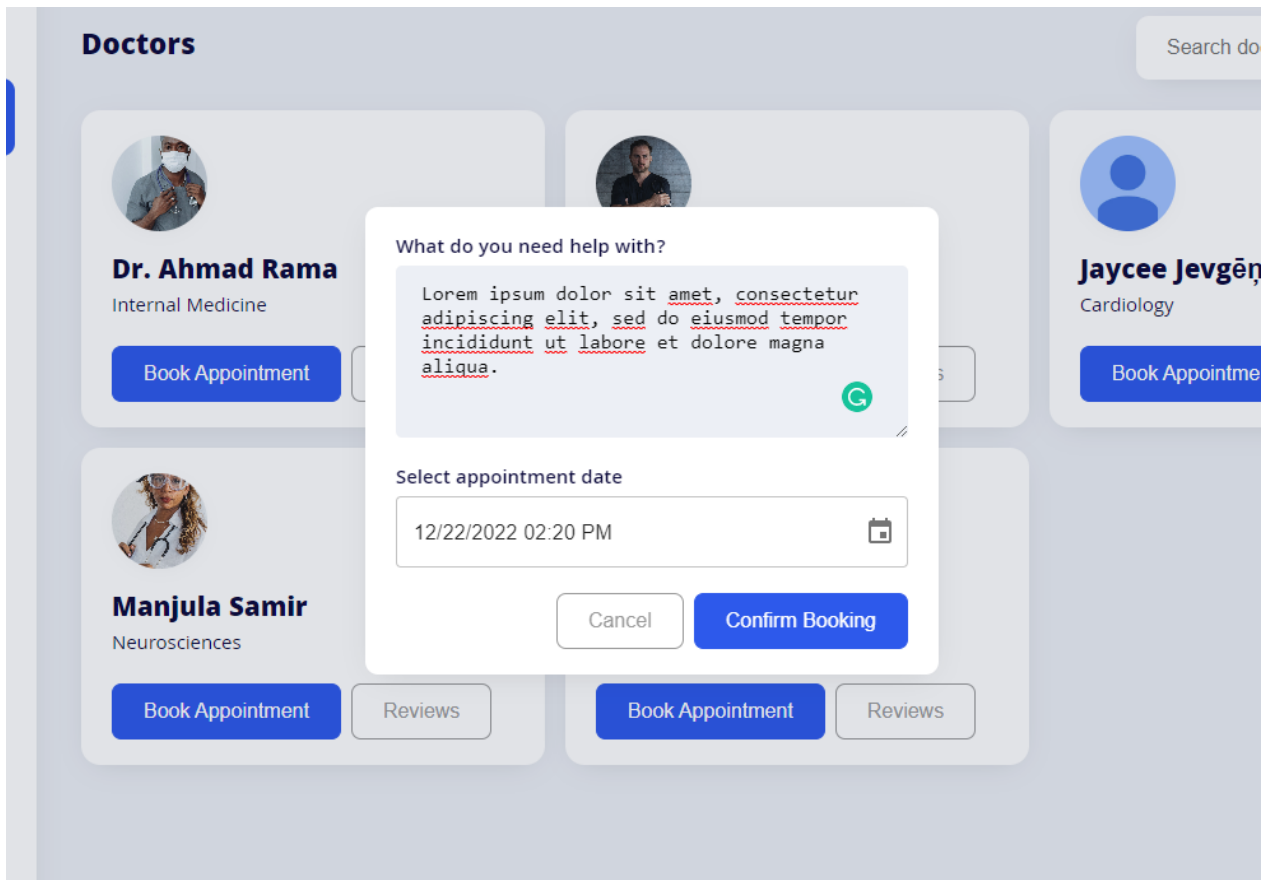
Users can search for a specific doctor using the search field on the top side of the page. The users can search for a doctor using the doctor's name or specialty, the results will show the doctors that username or specialty as the input in the search field.



A patient can check how good a doctor's service is by checking their reviews.



To book an appointment, the patient has to click the 'Book Appointment' button, that will display a booking tab. The tab will include a booking form with two required fields: appointment reason, and appointment date. The form will also include two buttons: 'Confirm Booking', and a 'Cancel' button.

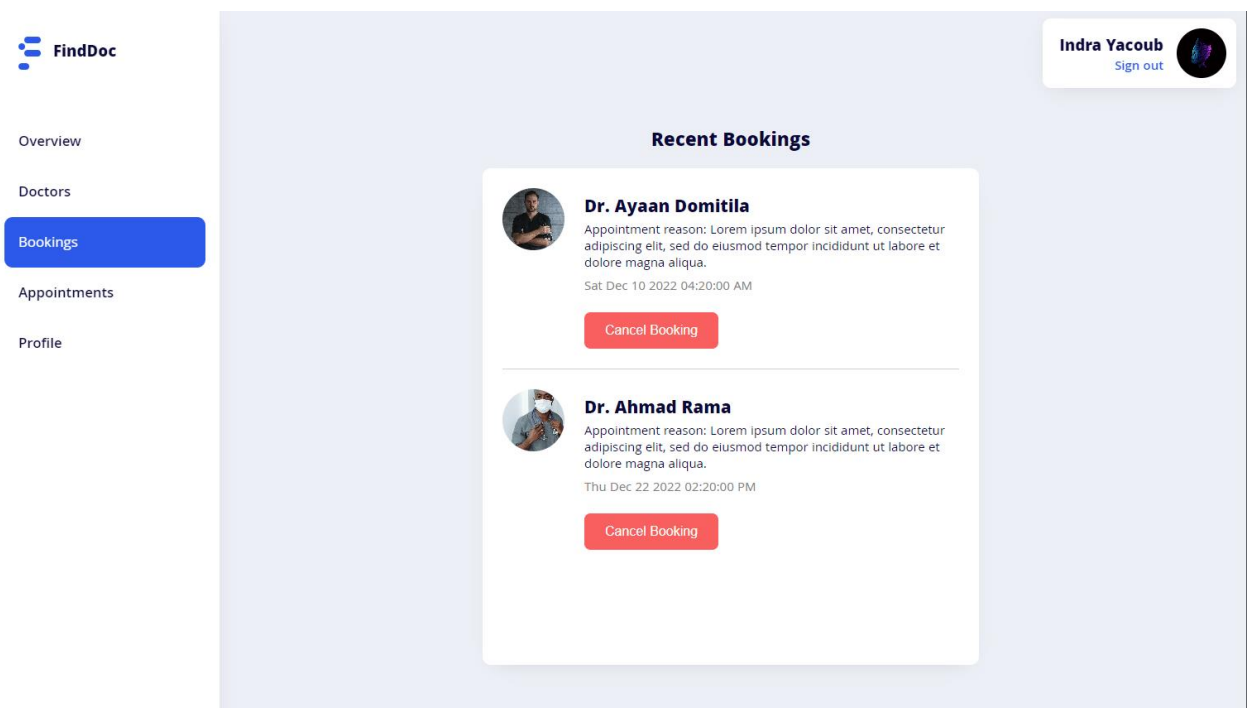


After the patient fills in the required fields, the patient has to click 'Confirm Booking' which will make a post request to the backend to add the appointment to the database if the appointment date is valid.

The booked appointment will be stored in the appointment model in the database, and the 'confirmed' field of this appointment will be set to false until the booked doctor confirms the booking.

4.2.3. Bookings

The bookings page will display a list of all the bookings made by the patient that are not confirmed. Once an appointment booking is confirmed by the booked doctor, the appointment will be removed from the bookings page and will be displayed in the appointments page instead.

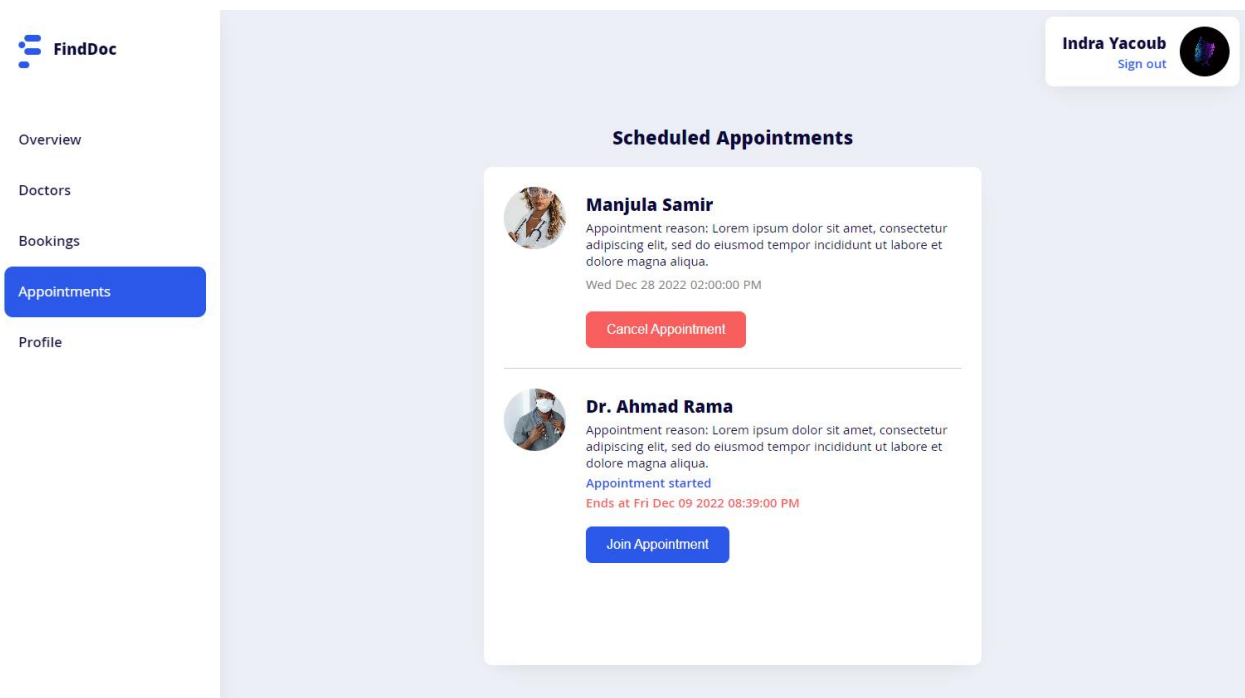


Each booking row shows the booked doctor's username, appointment reason, appointment date, and a 'Cancel Booking' button.

When the 'Cancel Booking' button is clicked, a confirmation tab will be displayed for the patient to confirm the booking cancellation, after the patient confirms, a delete request will be made to the backend to delete the booking from the appointment model in the database.

4.2.4. Appointments

The appointments page will display a list of all the booked appointments made by the patient that are confirmed by the booked doctors.

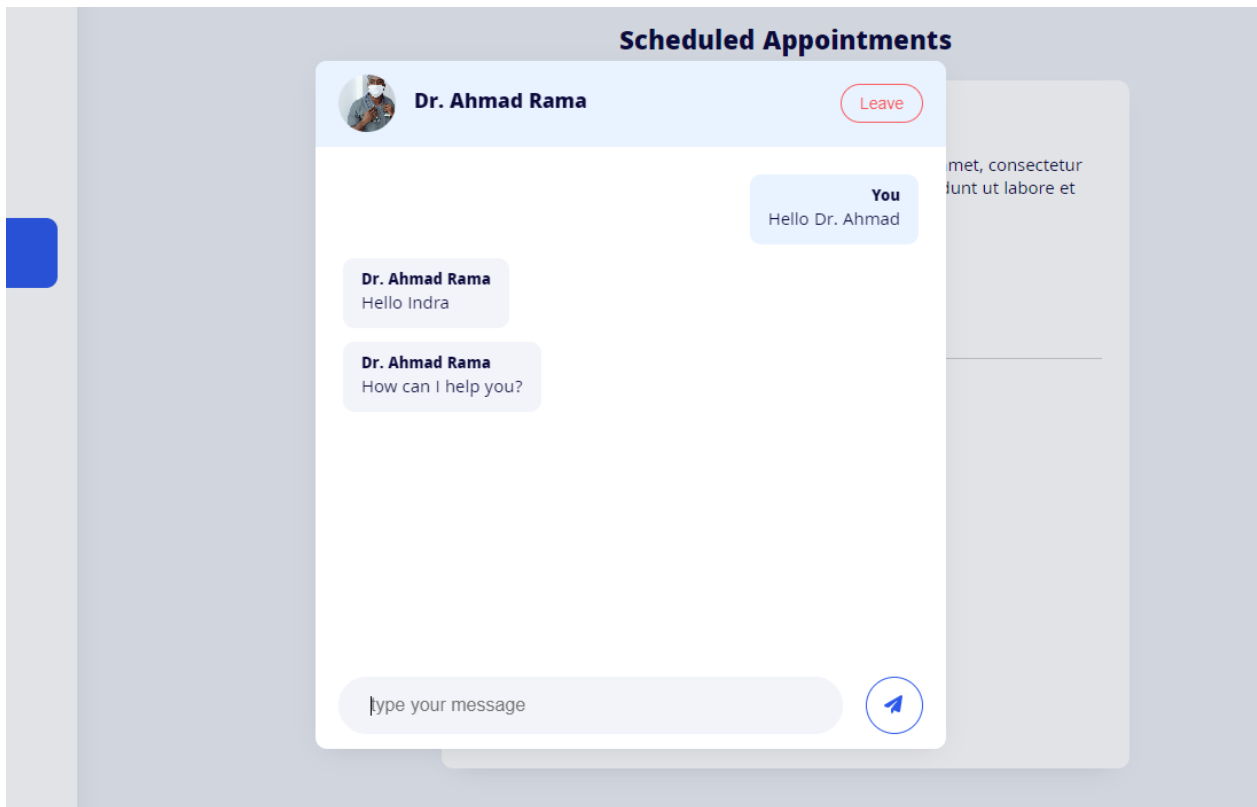


Each appointment row shows the booked doctor's username, appointment reason, appointment date, and either a 'Cancel Appointment', 'Join Appointment', or 'Delete Appointment' button depending on the appointment status.

- If the appointment is yet to start, the appointment date will be shown along with the 'Cancel Appointment' button.
- If the appointment started, the appointment status will be shown as 'Appointment started' along with the appointment ending time, and a 'Join Appointment' button.
- If the appointment finished, the appointment status will be shown as 'Session finished' along with a 'Delete Appointment' button.

Both the 'Cancel Appointment' and 'Delete Appointment' buttons would delete the appointment from the appointment model in the database once clicked and confirmed.

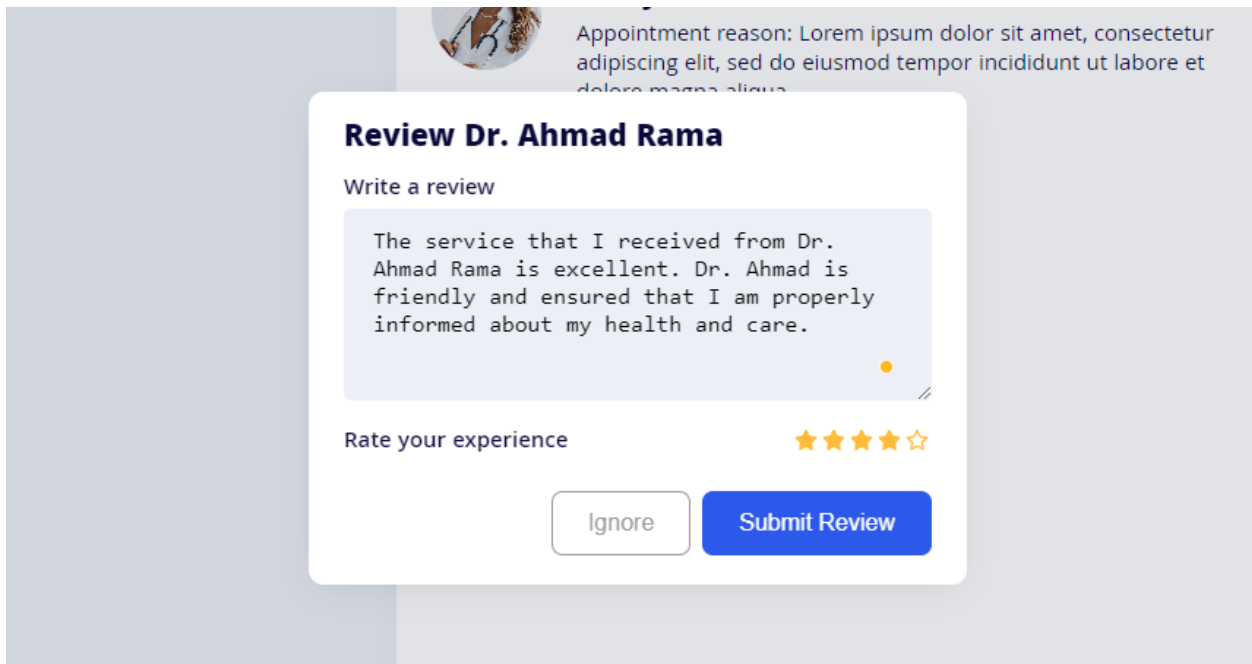
As for the 'Join Appointment' button, when it's clicked, the appointment chat tab is displayed and the patient joins the appointment chat session with appointed doctor.



The appointment session lasts for one hour after the chosen appointment time.

The chatting feature is implemented using [PubNub](#).

Patients cannot join the appointment session again after leaving the chat. When the patient clicks the 'Leave' button, the patient will be prompted to confirm since they cannot join again. After the patient confirms and leaves the chat session, they will be prompted to review the doctor.



The screenshot shows a web interface for reviewing a doctor. At the top, there is a circular profile picture of a doctor and a text field for the appointment reason containing placeholder text. Below this is a modal form titled "Review Dr. Ahmad Rama". The form has a section "Write a review" with a text area containing a sample review. Below the text area is a "Rate your experience" section with five star icons, the first four of which are filled. At the bottom of the form are two buttons: "Ignore" and "Submit Review".

Appointment reason: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Review Dr. Ahmad Rama

Write a review

The service that I received from Dr. Ahmad Rama is excellent. Dr. Ahmad is friendly and ensured that I am properly informed about my health and care.

Rate your experience ★★★★★

The patient can review the doctor and rate their experience by filling in the review message field and selecting a rating, then submitting the review by clicking the 'Submit Review' button. After the patient clicks the submit button, a post request will be made to the backend to store the doctor's review by the patient in the review model in the database.

After the patient submits the review or clicks the 'Ignore' button, a delete request is automatically made to delete the appointment from the appointments model in the database.

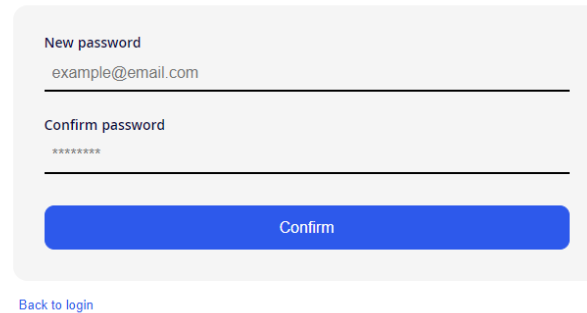
4.2.5. Profile

The patients profile page displays the patient's account information and allows them to update their profile picture, username, and their password.

The screenshot shows the 'FindDoc' application interface. On the left is a sidebar with navigation links: 'Overview', 'Doctors', 'Bookings', 'Appointments', and 'Profile' (which is highlighted with a blue background). The main content area is titled 'Profile'. At the top right of this area, there is a user header for 'Indra Yacoub' with a 'Sign out' link and a profile picture icon. The central 'Profile' section contains a 'Profile picture' field with a circular image placeholder and a 'Choose File' button next to the text 'No file chosen'. Below this is a 'Username' field containing the text 'Indra Yacoub'. At the bottom of the profile section, there is a link 'Get password reset link' and a blue 'Update' button.

The patient can update their profile by modifying the field they want to update then clicking the 'Update' button. When the 'Update' button is clicked, a tab will be shown prompting the patient to enter their password to confirm the profile update.

Change Password



The form is titled 'Change Password' and is contained within a light gray rounded rectangle. It has two input fields: 'New password' with the placeholder text 'example@email.com' and 'Confirm password' with placeholder text '*****'. Below the fields is a blue 'Confirm' button. At the bottom left of the form is a link 'Back to login'.

New password
example@email.com

Confirm password

Confirm

[Back to login](#)

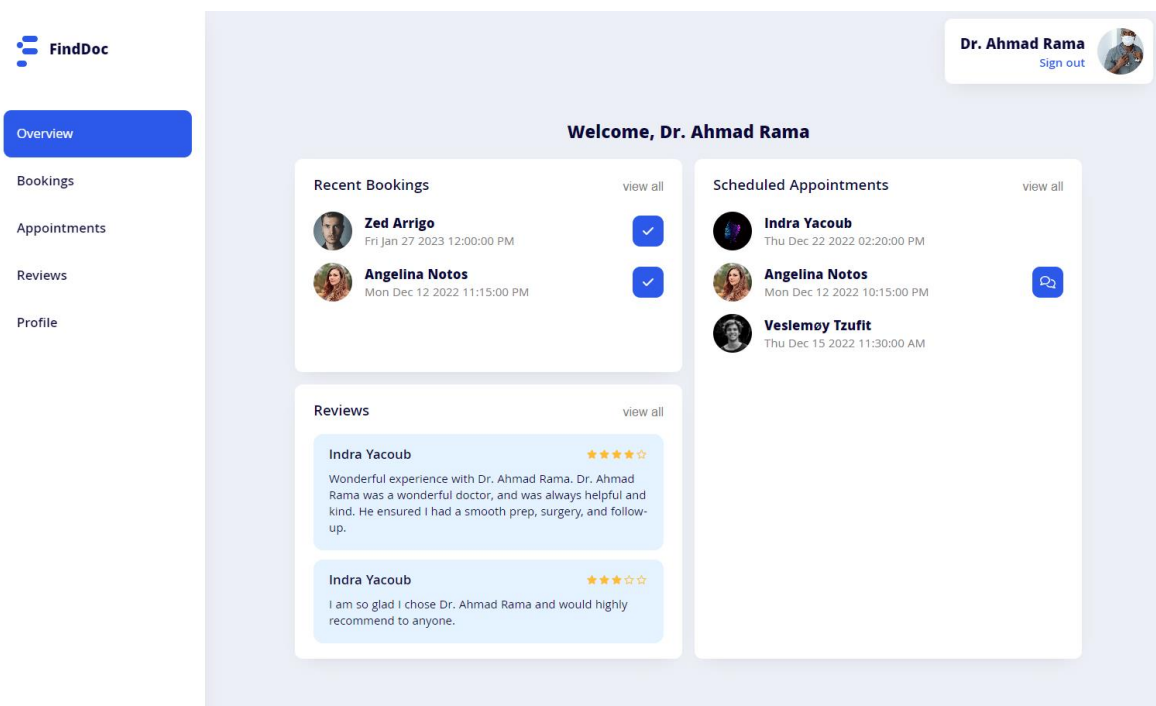
Once the user enters the new password and confirms it, if the passwords match and are valid, a post request is made to the backend to update the user's password in the user model in the database.

4.3. Doctor features

When users registered as a 'Doctor' role login to the application, the user is redirected to the doctor's dashboard. The doctor's dashboard includes 5 pages: Overview, Bookings, Appointments, Review, and Profile page.

4.3.1. Overview

The overview page is the first page that the doctor joins after logging in. This page shows an overview of the bookings made for the doctor, scheduled appointments, and the doctor's reviews by patients.



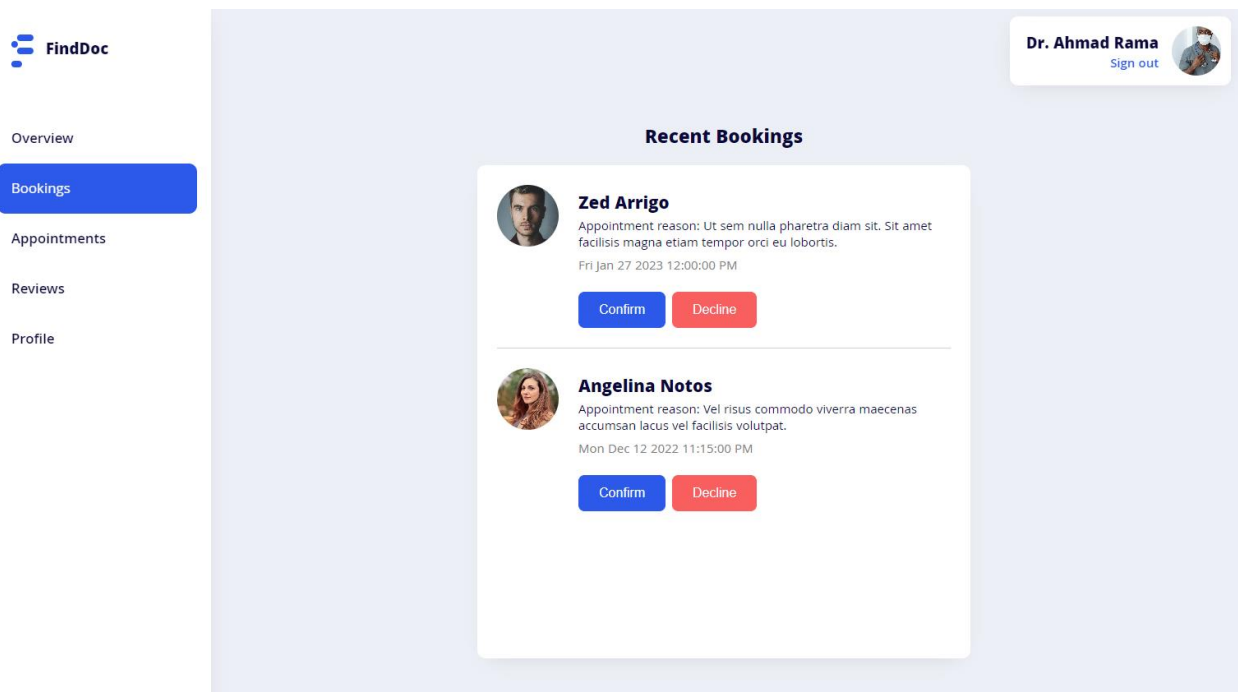
When the page first loads, the doctor's bookings, appointments, and reviews are retrieved from the database and displayed in three containers 'Recent Bookings', 'Reviews', and 'Scheduled Appointments' container.

A button that allows the doctor to confirm the appointment booked by a patient will be shown to the right side of each appointment booking row.

If an appointment has started, a button that allows the user to join the appointment chat session will be shown to the right side of the specified appointment row.

4.3.2. Bookings

The bookings page will display a list of all the bookings made by patients that are not confirmed by the doctor yet. Once an appointment booking is confirmed by the doctor, the appointment will be removed from the bookings page and will be displayed in the appointments page instead.



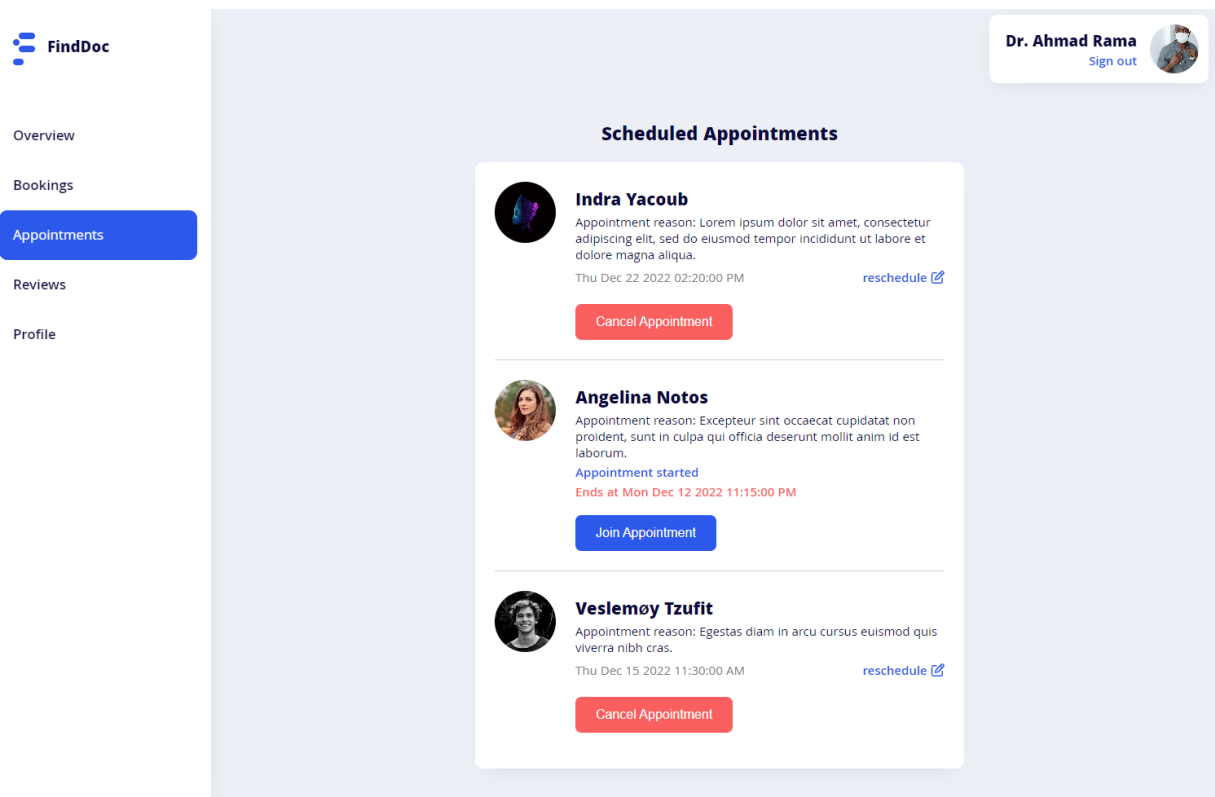
Each booking row shows the username of the patient who made the booking, appointment reason, appointment date, and two buttons ‘Confirm’ and ‘Decline’.

When the ‘Confirm’ button is clicked, a post request will be made to the backend to set the ‘confirmed’ field to true in the appointment model for this booking in the database, and the confirmed booking will be removed from the bookings page and displayed in the appointments page.

When the ‘Decline’ button is clicked, a confirmation tab will be displayed for the patient to confirm the booking declination, after the doctor confirms, a delete request will be made to the backend to delete the booking from the appointment model in the database.

4.3.3. Appointments

The appointments page will display a list of all the bookings made by patients for the doctor that are confirmed.



Each appointment row shows the username of the patient who made the booking, appointment reason, appointment date, and either a ‘Cancel Appointment’, ‘Join Appointment’, or ‘Delete Appointment’ button depending on the appointment status. If the appointment is yet to start, also a ‘reschedule’ button is displayed.

- If the appointment is yet to start, the appointment date will be shown along with a ‘reschedule’ button and a ‘Cancel Appointment’ button.
- If the appointment started, the appointment status will be shown as ‘Appointment started’ along with the appointment ending time, and a ‘Join Appointment’ button.
- If the appointment finished, the appointment status will be shown as ‘Session finished’ along with a ‘Delete Appointment’ button.

Both the 'Cancel Appointment' and 'Delete Appointment' buttons would delete the appointment from the appointment model in the database once clicked and confirmed.

The 'reschedule' button allows the doctor to reschedule the appointment date, which would make a post request to the backend to update the date in the appointment model for the specified appointment in the database.

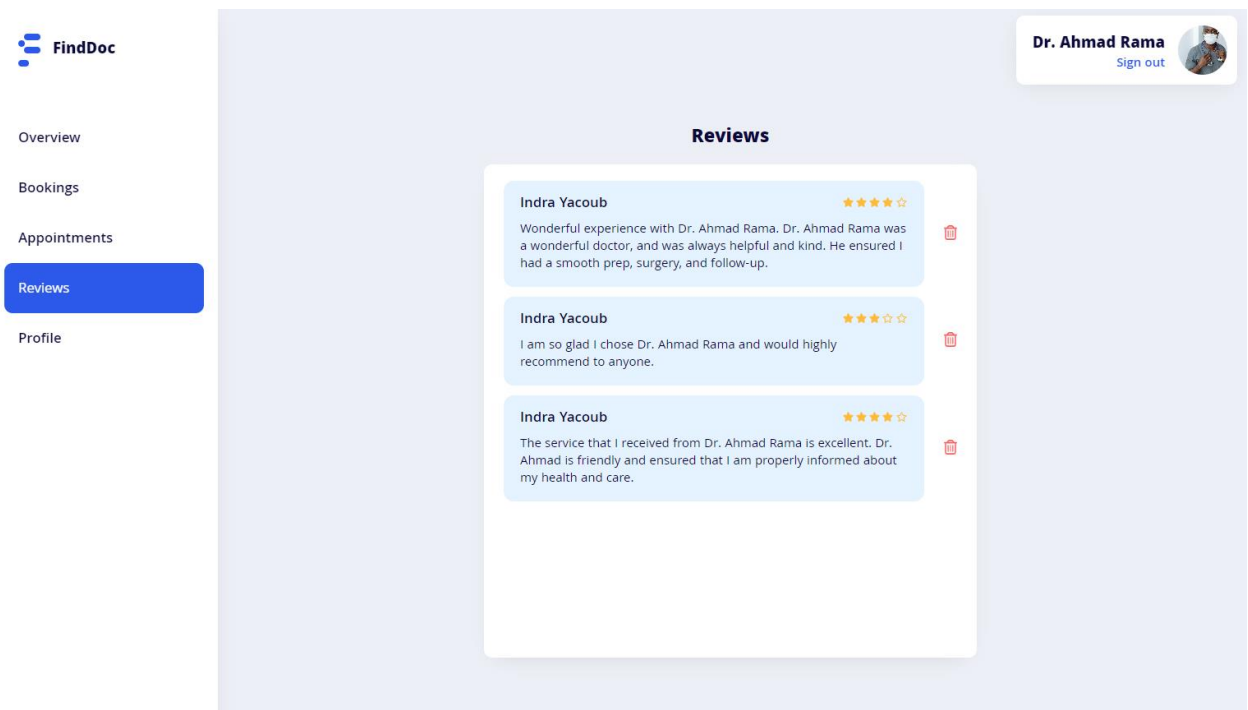
As for the 'Join Appointment' button, when it's clicked, the appointment chat tab is displayed and the patient joins the appointment chat session with appointed doctor.

The appointment chatting session works the same way for the doctor like it works for the patient, but unlike the patient user, the doctor can leave the chat tab and join any time before the session ends, Whereas the patient can only join once.

Also after the doctor leaves the chat session, the review prompt is not displayed for the doctor the same way it works for a patient.

4.3.4. Reviews

The reviews page displays a list of all the reviews submitted by patients for the doctor.



Each review row shows the username of the patient that submitted the review, the review message, rating out of 5, and a button to delete the review.

When the delete button is clicked, a confirmation tab is displayed asking the doctor to confirm before deleting the review. Once deletion is confirmed, a delete request is made to the backend to delete the doctor's review from the review model in the database.

4.3.5. Profile

The doctors profile page displays the doctor's account information and allows them to update their profile picture, username, specialty, and their password.

The screenshot shows the 'FindDoc' application interface. On the left is a sidebar with navigation links: Overview, Bookings, Appointments, Reviews, and Profile (which is highlighted in blue). The main content area is titled 'Profile' and contains a form for a doctor's profile. At the top right of the main area, there is a user header for 'Dr. Ahmad Rama' with a 'Sign out' link and a profile picture. The profile form includes a 'Profile picture' section with a circular image of a doctor and a 'Choose File' button. Below this are text input fields for 'Username' (containing 'Dr. Ahmad Rama') and 'Specialty' (containing 'Internal Medicine'). A link 'Get password reset link' is located below the specialty field. An 'Update' button is positioned at the bottom right of the form.

The profile page works the same way for the doctor as the patient, but the doctor's profile page includes an additional field 'Specialty', that allows doctors to update their specialty.

When the specialty field is updated, a profile update post request is made to the backend which will update the specialty field that is linked to the doctor's id in the specialty model in the database.

5. Conclusion

In conclusion, web-based appointment scheduling is more patient-focused than conventional appointment methods and offers many benefits because of increased accessibility. Many practices have seen improvements after implementing web-based appointment systems, including decreased no-show rates, reduced staff workload, shorter wait times, and higher patient satisfaction.

Although these improvements imply web-based appointment systems might result in favorable outcomes, this claim should be supported by more advanced study designs. The positive results could be caused by additional factors, or by a combination of additional factors and web-based appointment systems.

The slow adoption of online appointment scheduling is because of factors involving both patients and doctors. Web-based scheduling is not widely used by doctors for a number of reasons, including cost, flexibility, safety, and integrity. Patients' past experiences using computers and the internet, as well as their preferences, are the primary reasons for their reluctance to adopt web-based appointment scheduling.

The results of this thesis suggest that web-based scheduling interventions can improve a range of patient outcomes, but more research and improvements are required.

6. Bibliography

About PubNub, The Real-time Communication Platform | PubNub. (n.d.). Retrieved from PubNub: <https://www.pubnub.com/company/>

Cloudinary Image & Video Management - Documentation. (n.d.). Retrieved from Cloudinary: <https://cloudinary.com/documentation>

Express Tutorial Part 3: Using a Database (with Mongoose). (n.d.). Retrieved from MDN: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose

How To Use React Styled Components in React Efficiently. (n.d.). Retrieved from CopyCat: <https://www.copycat.dev/blog/styled-components-react/>

HTTP | MDN. (n.d.). Retrieved from MDN: <https://developer.mozilla.org/en-US/docs/Web/HTTP>

MongoDB: The Developer Data Platform | MongoDB. (n.d.). Retrieved from MongoDB: <https://www.mongodb.com/home>

Mongoose ODM v6.8.1. (n.d.). Retrieved from Mongoose: <https://mongoosejs.com/>

Node.js | beecrowd. (n.d.). Retrieved from Node.js: <https://beecrowd.io/en/blog/node-js-2/>

Sending Emails From React With EmailJS. (n.d.). Retrieved from OpenReplay: <https://blog.openreplay.com/sending-emails-from-react-with-emailjs/>

surazova / mongoDB: MongoDB stores data in flexible, JSON-like ... (n.d.). Retrieved from Github: <https://github.com/surazova/mongoDB/>

What do client side and server side mean? | Client side vs. server side. (n.d.). Retrieved from Cloudflare: <https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/>

What is MongoDB Atlas? — MongoDB Atlas. (n.d.). Retrieved from MongoDB.

What is React.js? (Uses, Examples, & More). (n.d.). Retrieved from HubSpot:

<https://blog.hubspot.com/website/react-js>

What Is The MERN Stack? Introduction & Examples | MongoDB. (n.d.). Retrieved from

MongoDB: <https://www.mongodb.com/mern-stack>

University of Debrecen

Faculty of Informatics, Computer Science Engineering

Online Doctor Appointments Web Application (FindDoc)

Candidate name: Hesham Ibrahim

Supervisor: Dr. Adamkó Attila

Debrecen, 2022

Thesis

Candidate name: Hesham Ibrahim

Debrecen, 2022