TELECAREPLUS: BRIDGING THE GAP BETWEEN PROVIDERS AND PATIENTS THROUGH TELEMEDICINE

PARISA GHANBARI

A project report submitted in conformity with the requirements for the degree of Master's of Science in Information Technology

Department of Mathematical and Physical Sciences
Faculty of Graduate Studies
Concordia University of Edmonton

TELECAREPLUS: BRIDGING THE GAP BETWEEN PROVIDERS AND PATIENTS THROUGH TELEMEDICINE

PARISA GHANBARI

Approved:

Rossitza Marinova, Ph.D.

Supervisor Date

Committee Member Name, Ph.D.

Committee Member Date

Patrick Kamau, Ph.D.

Dean of Graduate Studies Date

Abstract

Telemedicine has attracted a lot of attention recently because of its promise to increase healthcare access at low costs and with few restrictions. The telemedicine platform TelecarePLUS, which is proposed in this paper, enables healthcare service providers to provide patients with the highest calibre care remotely. TelecarePLUS can give patients quick access to healthcare regardless of their location or physical restrictions, especially in the wake of COVID-19, where social distance has been the norm. With capabilities including secure audio/video/text consultations, patient registration, electronic medical record management, prescription management, and patient monitoring, it is a one-stop shop for healthcare. The platform is simple to use and simple to access, with a key focus on safeguarding data privacy and preserving data integrity and confidentiality. TelecarePLUS has the potential to completely transform the digital healthcare sector given the rise of various diseases and labour shortages by bringing healthcare services to those who might not have easy access to them, such as those who live in remote locations or have mobility issues.

Keywords: telemedicine; TelecarePLUS; electronic medical records; digital healthcare

Acknowledgement

I like to sincerely express my gratitude to Dr. Rossitza Marinova for her invaluable support, advice and patience during this journey.

Dr. Marinova's supervision inspired me throughout the project to overcome all the challenges and aim for the excellence in my work.

I am immensely grateful for her belief in me, which helped me to strive for the best results and know my capabilities and talents better.

I would also like to show my appreciation for my teammates in the TelecarePLUS project. Their dedication and professionalism, their exceptional skills and cooperative spirit was the reason we were able to deliver such innovative project in a short amount of time.

In the end, I am profoundly thankful for all the support and love received from my family and friends. For sure without their sacrifices and constant encouragement, this project would not have been possible.

Contents

List of Tables

List of Figures

Listings

1 Introduction

Despite of ground breaking advancements in the field of healthcare, many individuals still face challenges in accessing high quality healthcare services.The rise for remote healthcare services delivery has further been highlighted by the global crisis caused by the COVID-19 pandemic. This is where telemedicine comes into the picture. It refers to the use of telecommunication technologies like audio, video and text to automate the process of consultation and provide healthcare services to those in need in a seamless way. Moreover, there are some pre-existing platforms in the market that have advanced themselves further by including the aspects of robotics and virtual reality to deliver healthcare [1].

Despite of telemedicine standing up as a promising solution to the challenges imposed, existing similar applications still face the challenges like technical difficulties, limited to appointment types and security concerns. Traditional measures of delivering healthcare, though having many challenges, have developed a sense of trust and adaptability among its end users. Telemedicine, through the use of technology, still needs to develop the trust. However, it has shown some promising signs, especially after the outburst of the pandemic.

Therefore, it is a critical need to develop a new application that has the potential to address these challenges and also improve outcomes [2]. This is where TelecarePLUS, a telelmedicine web application that offers health services to patients anywhere and anytime comes into the picture.

My role in this project was to design a seamless and beautiful user experience (UI), suitable for all devices and visually appealing for all users. Utilizing the latest technologies [3] and best practices in front-end development, I was able to harness the power of React JS (18.2.0) and JSX (JavaScript XML) to create a responsive and beautiful website. React JS library, Axios was used to send data securely and safely to back-end without exposing any sensitive information to the end user. One of the most important aspects of the website that had to be achieved was the responsive design which was successfully implemented using CSS 3. Other libraries like Aos was used in order to help make the website look fabulous and elegant.

In short, the website provides a user friendly environment for patients to easily register themselves as a new or existing patient and share their health problems and other required information. On successful verification, the patient can then choose their desired doctor, date and slot and then proceed to book an appointment.

Many features and functions were implemented in the project which some of them will be explained in this paper.

For sure in the near future, working harder and dedicating more time on this project will make the website one of the most revolutionizing platforms in healthcare.

1.1 Problem Statement

The scenario of healthcare services delivery before the introduction of telemedicine was different. The services were provided through the in-person interaction between patients and healthcare service providers. Patients requiring medical attention had to either physically visit hospitals or spend a good amount of money to afford similar facilities at home. Consultation between doctor and patient had to be done face to face.

This method came with numerous challenges for both the patients and providers. Patients had to schedule their consultation well in advance which resulted in long waiting times. Patients with mobility limitations and severe illness faced travel distance as a major hindrance. Moreover, people belonging to remote areas had minimum or no access to specialist opinions.

On the other side, healthcare service providers faced challenges related to high traffic of patients during peak intervals. This influx resulted in ineffective resource management and improper staff allocation. Manual communication between different provider bodies resulted in disorganized care and diagnosis. Ineffective way of managing patient medical records saw some errors, putting patient's data to the risk of being handled insecurely.

To address these challenges, TelecarePLUS emerges as a one-stop healthcare platform. The platform consists of features like secure audio/video/text consultations, patient registration, medical records management, prescription and report management and patient monitoring. The platform achieves its goals through user friendly and interactive designs and also has a main focus of protecting the integrity and confidentiality of patient's data.

1.2 Organization of this report

This project report is organized as follows:

- Chapter 2: The objectives of creating the TelecarePLUS platform are explained and multiple questions are posed to guide the project's completion.
- Chapter 3: This chapter presents the literature review and The significance of telemedicine.It is explained how the telehealth system is driven by the growth of the technology and constant demand of patients after COVID-19.
- Chapter 4: This chapter outlines and discusses several key factors that were necessary to consider when commencing the TelecarePLUS project.
- Chapter 5: In this chapter, the technical aspects and technology stack are discussed. The use of different technologies are explained and highlighted along with sample codes from the project.
- Chapter 6: The achievements of TelecarePLUS project is concluded in this

chapter. The significance of utilizing the best technologies and a user-friendly design is emphasized and also the future possibilities and additions are mentioned.

- Chapter 7: It focuses on the contribution of the project and report.
- Chapter 8: The purpose of this chapter is to showcase visual support for the discussed project. These pictures are appended to provide a better understanding and additional context for the readers.

2 Objectives / Research Questions

Our main goal in this project was to create an innovative and state of the art web application to provide online consultation and healthcare services for the patients in the name of the telemedicine platform called TelecarePLUS. To commence the project a first, we assigned different roles to each member and then we set forth some primary questions for each assigned responsibility that had to be answered in order to complete the project. The mentioned questions for my role in the web application (CCH) are as follows:

- What are the best technologies to leverage that will satisfy the need to implement a modern, intuitive and user friendly web application, compatible with patients' needs and desires [4]?
- How to successfully and efficiently integrate essential APIs to fetch data from back-end and provide the necessary information for the patients to discern and choose from [10]?
- How to facilitate secure and smooth data transfer between front-end and back-end to protect patients' information and provide the essential data for the back-end developer [10] [11].
- How to optimize the code and utilize JSX, a JavaScript extension to do so [12] [13]?
- What libraries and frameworks can be used to efficiently increase the quality of the project and save time at the same time [14]?
- How to improve the user experience and provide the best UI and a visually captivating design on all the possible platforms [4].
- How to write the code in such a way that can be modified and updated easily in the future?

Having all these questions in mind, we tried our best to deliver a web application that not only meets the needs of our users but also displays our capabilities in coding and web development.

Addressing these questions, we surpassed our own standards and abilities in order to achieve the goals of this project in a short amount of time.

Also all the required steps for the design and then implementation were documented and all possible challenges were noted as we were working on new technologies.

The whole process was planned similar to Fig. 1 and followed with commitment.

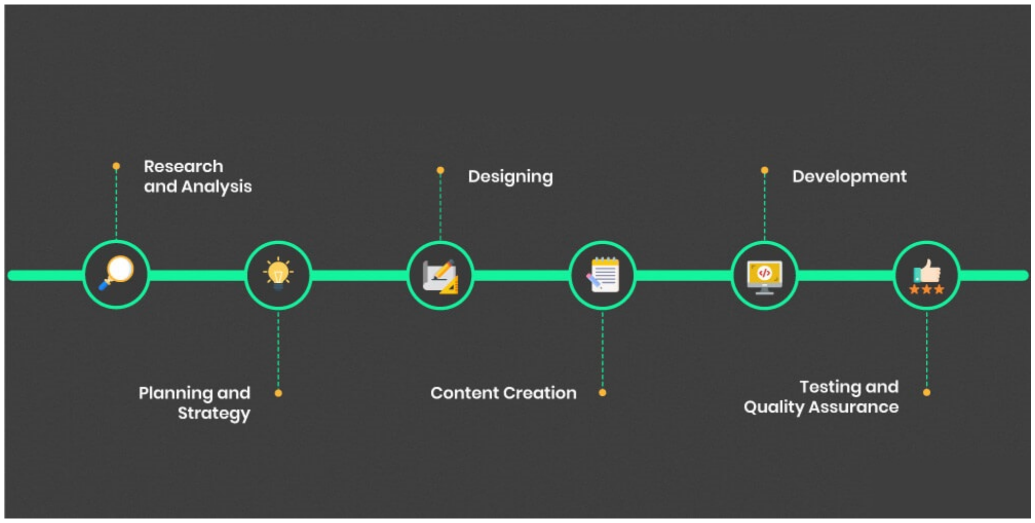


Figure 1: Steps to build the web application
[5]

3 Literature review (and theoretical framework)

Based on the article [6] the importance of telemedicine lies in the rapid growth of the technology and needs of patients for new ways to receive health services. After the Covid-19 pandemic, the significance of having such platform was felt and understood more than anytime. Many applications and web applications were developed after the pandemic, and yet, the need for a better and more efficient platform is visible.

Talking about telemedicine and telehealth systems, there are some factors that can be considered. First factor is such platform removes the barriers set between healthcare, doctors and patients. Patients do not need to be physically present to be assisted or examined. This both makes receiving healthcare more convenient and also decreases the chance of spreading contagious viruses in the society as it happened with Covid-19.

In the article written by "HealthTech" Magazine in May 2020 [7] telemedicine has offered a lifeline during the most difficult times of the pandemic and could have permanent implications on the market. It is likely to become a permanent fixture in healthcare delivery in the near future. A survey by IT vendor Sykes states that more than 60 percent of the patients are willing to try telemedicine due to the pandemic.

Kaiser Permanent, Oakland California based, which nearly operates 40 hospitals started conducting more than 90 percent of mental health visits virtually starting the pandemic period, which would have taken years to implement under normal circumstances.

Sourced from the report (Market Analysis Report) [9], the telemedicine market is estimated to grow in annual growth of 18.16% from 2023 to 2023, see Fig. 2.

More people are demanding the telehealth services and the quality of service is expected to increase rapidly. In May 2021, Walmart Inc. inherited MeMD-a telehealth provider which enables Walmart to offer virtual care access across U.S.

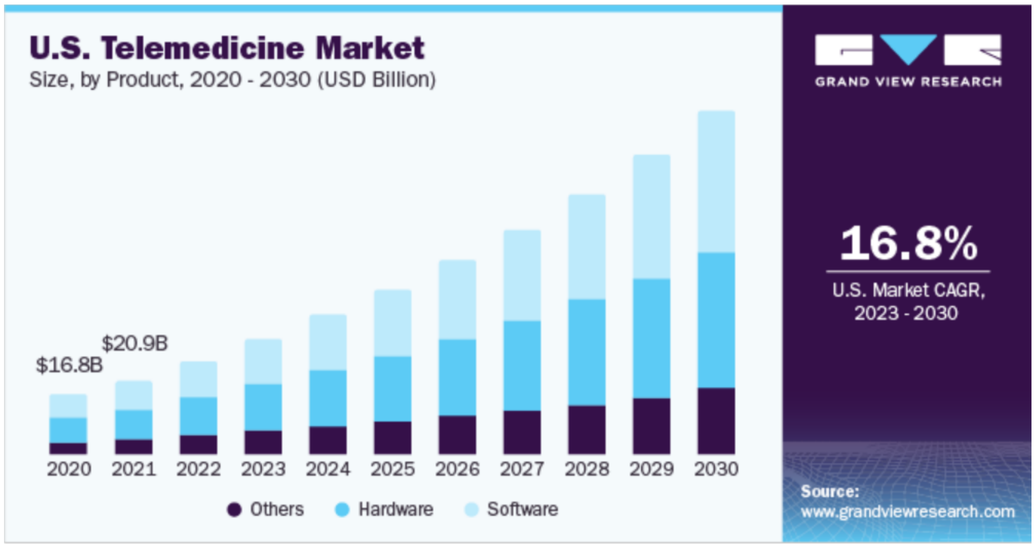


Figure 2: U.S telemedicine market
[5]

So based on the provided information, it can be concluded that the telemedicine and telehealth market is highly in demand and many opportunities [8] lie in this field as the demand is increasing exponentially.

4 Project Design

Commencing a project as a front-end developer, a few factors must be considered:

- Implementing user-friendly, visually appealing, and trustworthy interface is necessary [15]. The colors chosen in the website should promote care and trust and the design of the buttons, navigation bar and ... should be straight forward so all users in any age can simply go through their procedure.

In order to achieve this goal we used already created telemedicine web application as a sample to start with. The comments of the users and their feedback was also another useful source to understand the needs of our end users better. So the design had to follow the same successful pattern of the so called web applications while trying to improve the quality by considering the end user's feedback.

Another important factor as mentioned before, was making the pages responsive [16]. It is a modern and necessary attribute that each web application must have, however, for our project the necessity was even higher as the end users are patients and they may be in need of our services anywhere and anytime so they have to have the ability to successfully check out the website and complete their requests with no issues on any device.

Another thing to consider is if a website is not responsive it will not have a good SEO ranking [16]. If our web application is not visible for the users we will not have a chance to fulfill our project and be able to improve in the future as we are not receiving enough feedback from the user side.

It is true that the sole idea of the project is to be able to assist patients at home, however, any project has economical aspects as well as it needs to survive as a business [17]. So if we do not get enough users or they leave the website for poor design or incompatibility with their device, that's a huge negative point for the website and its SEO ranking [16] and it is considered as a failure.

As shown in Fig. 3 and Fig. 4, responsiveness was the main goal of the design part of our project.

Both pictures have been taken from TelecarePLUS web application.

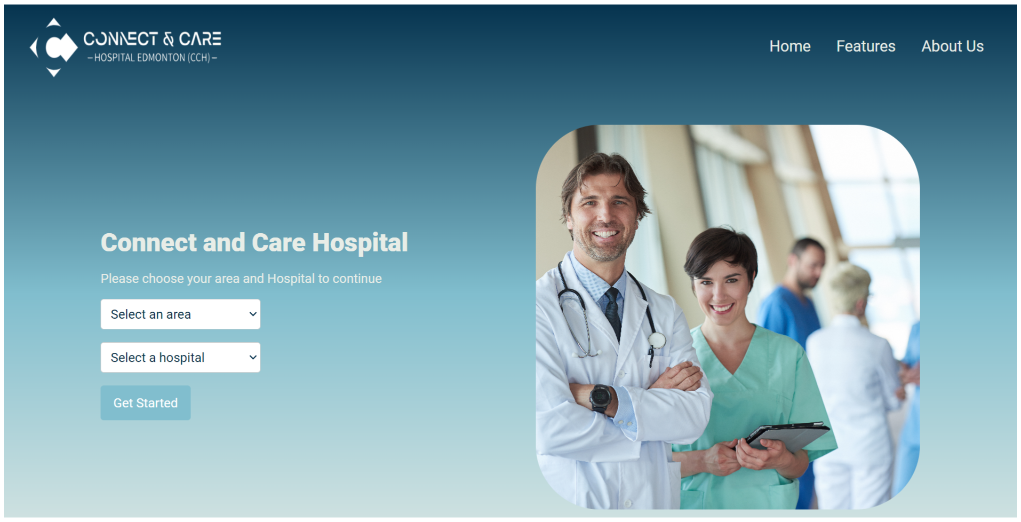

CONNECT & CARE
– HOSPITAL EDMONTON (CCH) –
Home
Features
About Us
Connect and Care Hospital
Please choose your area and Hospital to continue
Select an area
Select a hospital
Get Started

Figure 3: Homepage of our website in 1400 px width

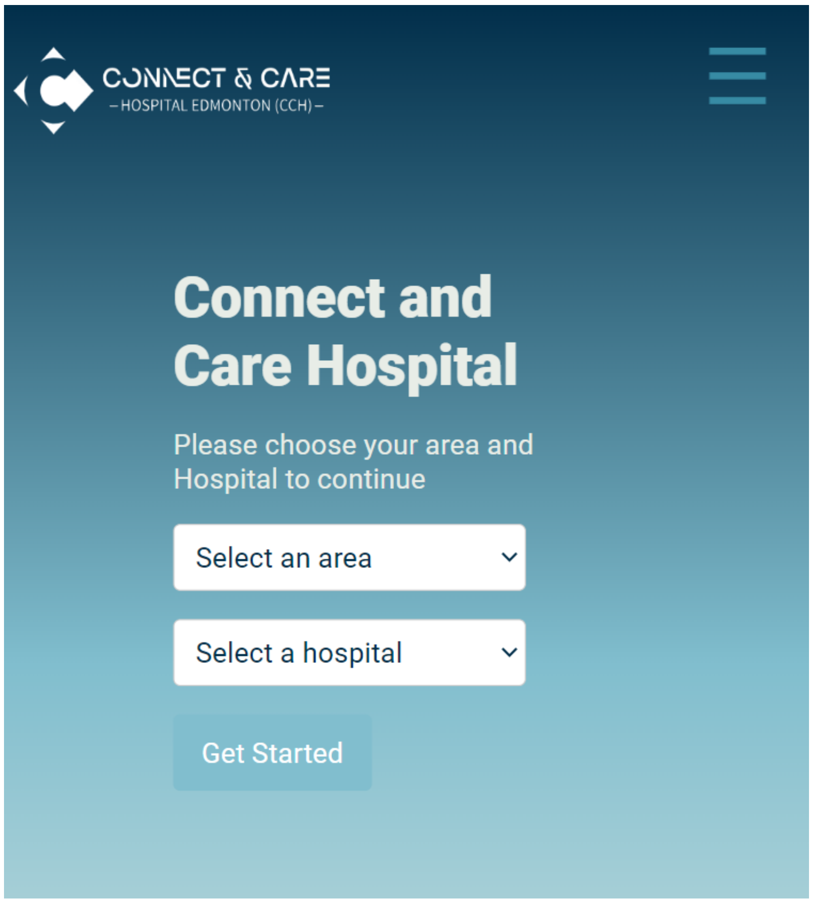

CONNECT & CARE
– HOSPITAL EDMONTON (CCH) –
Connect and Care Hospital
Please choose your area and Hospital to continue
Select an area
Select a hospital
Get Started

Figure 4: Homepage of our website in 560 px width

5 Project Implementation and Results

The front-end design and back-end functionality of the TelecarePLUS website were brought to life during the implementation phase by the integration of numerous technologies and components. The primary front-end technologies and environments used in this project are given in Table 1.

React.JS (version 18.2.0) and different React and JavaScript(ES6) libraries	Making the website dynamic and adding functionalities
JSX (JavaScript XML)	Writing the HTML code inside the React.JS file and optimizing the HTML code
HTML5	Creating a base to demonstrate the whole project
CSS3	Designing the web pages and adding enabling a responsive design
Chrome DevTools	an environment to see and debug the code
Visual Studio Code	The environment used to develope the code

Table 1: Tech stack for TelecarePLUS

Apart from the main HTML file that is a necessary part of any website the whole code for the HTML structure was written inside the React.JS file codes using JSX technology [12]. JSX is not a necessary part of React development but it enhances the functionality of the code and the debugging and coding process. It makes it easier to combine the react code with HTML elements [13] and also allows react to show warning messages and error regarding the HTML tags and their elements. The production procedure using JSX was much simpler and organized.

In the below mentioned example in Listing 1, we are combining HTML elements with React code and variables. This a code for a drop down menu that users can pick their genders from. All the elements must be wrapped inside a <div></div> element or <></> in order to work. This piece of code is returned as the result of a function named "GenderList" which is later called in the main JSX code of the file.

```

  return (
    <div>
```

```
        <label htmlFor="gender">Gender:</label>
        <select
          value={selectedGender}
          id="gender"
          onChange={handleGenderChange}
        >
          <option value="" disabled hidden>
            Your gender
          </option>
          {genderOptions.map((option) => (
            <option key={option.value} value={option.value}>
              {option.label}
            </option>
          ))}
        </select>
      </div>
    );

```

Listing 1: JXS example

The code mentioned below in Listing 2 adds functions and returns the JSX lines to the main JSX code.

```

<div className={Form_css.input_box}>
              <GenderList />
            </div>

```

Listing 2: Adding functions and returned JSX lines to the main JSX code

Also, more complicated parts of the React and JavaScript code can be embedded inside JSX and vice versa, making it completely easy and convenient to work with, implementing everything in the same file as shown below in Listing 3:

```

const guardianCountryOptions = guardianCountries.map((
   guardianCountry) => ({
    value: guardianCountry.idd.root,
    label: (
      <div>
        <img
          src={guardianCountry.flags.svg}
          alt={guardianCountry.name.common}
          style={{ width: "20px", height: "15px", marginRight: "5px"
    }}
        />
        {guardianCountry.idd.root}
        {guardianCountry.idd.suffixes &&
          guardianCountry.idd.suffixes.length > 0 && (
            <span> {guardianCountry.idd.suffixes[0]}</span>
          )}
      </div>
    ),
    data: guardianCountry, // Store the entire country data in the
   option
  }));

```

Listing 3: Embedded JXS code

In the above example, we're directly assigning the data fetched from an API inside our HTML elements.

The graphical design of the project was mostly done using pure CSS3 and the latest CSS properties such as Grid. Also React was used to add dynamic visuals in the design. Some examples for how CSS styling was done is mentioned below in Listing 4. The code also explains how we implemented the responsiveness behavior of the web application.

```
.nav_list {
  margin-right: 2rem;
}
.nav_list ul {
  display: flex;
  list-style: none;
}
.burger_icon {
  display: none;
  padding: 1rem 2rem;
}
.burger_icon span {
  display: block;
  width: 2rem;
  border: #e8ede7 solid 2px;
  margin: 10px;
}
@media (max-width: 780px) {
  .nav_list {
    display: none;
  }
  .burger_icon {
    display: block;
  }
  }

```

Listing 4: CSS example

In the screens in size more than 780px we normally have our main navigation bar named as (nav_list) and the burger icon for the navigation menu is not displayed. However, as soon as the screen width becomes less than 780 px the design is changed. The element with the (nav_list) class stops displaying and the burger icon will show as a replacement for the navigation bar. The difference was visible in reference to Fig. 3 and Fig. 4.

CSS code is written separately in a different file and imported in the React file. The normal CSS files are applied to all React files and in order to make them specifically for a certain React file we add the key word "module" in the CSS file name. Assigning CSS classes (The syntax) for "module" CSS files are also different. In the TelecarePlus project, I created one CSS file applying globally CSS properties and styles for all files (for example footer and header which are consistent in most pages) and one specific CSS file for each React.JS file or page in the web application to apply specific styles according to elements and contents on the page.

The difference between assigning the global CSS file and specific CSS file as an example in the TelecarePLUS project comes as follows in Listing 5:

```

import Features_css from "../CSS/features.module.css"; \\ The CSS
   file created specifically for the "Features" page
import "../CSS/allFiles.css"; \\The global CSS file

\\add the module CSS code

<div className={Features_css.explanation}>
              <h2>Book an Appointment</h2>
            </div>

\\add the global CSS code

<div className="info">
              <p> 2023 cch.me Inc. All rights reserved.</p>
              <p>Emergency/Help Call</p>
              <p>780*********</p>
            </div>

```

Listing 5: Adding global and module CSS files

As mentioned, React was also in play adding visuals to the website. One of the React libraries used to make it happen was AOS [18]. AOS is a library used to animate elements on the screen [19], for example adding scroll on motion visuals to the website. Just like any library it has to be installed and then added to the code. The library is first activated inside the code using useEffect hook [20]. useEffect is basically a hook in React.JS which is used to perform side effects in the component [21]. The code is shown below in Listing 6:

```

useEffect(() => {
    Aos.init({ duration: 2000 }); // Activate the animation effect
  }, []);

```

Listing 6: Initiating AOS

Then the predefined code in the library can be used to add animations to the elements [18] like the example below in Listing 7.

```
        <div data-aos="fade-up" className={Features_css.img6_box}>
          <div>
            <h2>Consult your doctor online and receive E-prescription</h2>
          </div>
          <div>
            <img src={img6} alt="Icons from flatIcon" />
          </div>

```

Listing 7: Utilizing AOS inside JSX elements

The main parts of the project however, had to be coded using React.JS. Connecting front-end code to back-end, calling APIs, navigating through different pages and exporting the React code in HTML file and etc are all done by React.JS.

Working with this technology was a bit challenging for me as I had zero experience using React. However, working with JavaScript had given me the ability to understand how React works better as a front-end developer.

Some parts of the code are mentioned below as example in Listing 8. We have one of the main React file where we upload all the pages and export them all in the indext.js file later to be uploaded in the HTML file later. Any page that is added inside the project must be imported in this file (App.js) and in the router. Navigating through different pages in React is possible through using Router which has to be imported in the project. Line 14 is adding the "HomePage" as the primary page of the website and line 20 is creating a error page for all the wrong pages that users enter inside URL manually. It is an efficient and organized way of navigation possible thanks to React.JS.

```
1  import React from "react";
2  import { Route, Routes, Navigate } from "react-router-dom";
3  import { HomePage } from "./Pages/HomePage";
4  import { Features } from "./Pages/Features";
5  import { Form } from "./Pages/Form";
6  import { Otp } from "./Pages/Otp";
7  import { DoctorPage } from "./Pages/DoctorPage";
8  import { Error } from "./Pages/Error";
9
10 export function App() {
11   return (
12     <Routes>
13       <Route path="/" element={<Navigate to="/homepage" replace />} />
14       <Route path="/homepage" element={<HomePage />} />
15       <Route path="/features" element={<Features />} />
16       <Route path="/form" element={<Form />} />
17       <Route path="/otp" element={<Otp />} />
18       <Route path="/doctorpage" element={<DoctorPage />} />
```

```
      <Route path="/*" element={<Error />} />
    </Routes>
  );
}
export default App;

```

Listing 8: Router in React.JS

In the example mentioned below in Listing 9, we are handling the input received from users in the form. So we're adding some restrictions and conditions for the users as well as setting the final input as the value that later will be sent to the back-end (the back-end code is not mentioned here and only the values are received and displayed in the console.log).

To write these functions, useState has been used which is a React.JS hook [22] used to add states to functional components [23]. Each time the user adds anything to the input element written in JSX code the React functions receive the input and change the state of the value based on the input change. The values later are handled in "handleSubmit"when the user tries to submit the form. That is where they will receive the final validation errors to fill the form properly. (Just a part of the whole code is added in Listing 9 in order to explain the functionality of the code).

```

 const [age, setAge] = useState("");
 const [monthInput, setMonthInput] = useState("");

    const handleAgeChange = (event) => {
    const value = Number(event.target.value);
    // value within valid range
    if (isNaN(value) || value < 0 || value > 120) {
      setAge("");
    } else {
      setAge(value.toString());
    }

  };
  const handleMonthChange = (event) => {
    const month = parseInt(event.target.value);
    if (isNaN(month) || month < 1 || month > 12) {
      setMonthInput("");
    } else {
      setMonthInput(month.toString());
    }
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    console.log("Age:", age);
    console.log("Month Input:", monthInput);
    }

  return (
  <div className={Form_css.input_box}>
              <label htmlFor="number-input">Age:</label>
              <input
                type="number"
                id="number-input"
                placeholder="Your age"
                value={age}
                min={1}
                max={120}
                onChange={handleAgeChange}
              />
            </div>
            <div className={Form_css.input_box}>
              <label htmlFor="month-input">Month:</label>
              <input
                type="number"
                id="month-input"
                placeholder="Enter month"
                value={monthInput}
                min={1}
                max={12}
                onChange={handleMonthChange}
              />
            </div>
);
```

Listing 9: Handling the input received from user

In the code snippet mentioned below in Listing 10, we are fetching some data about doctors in the "doctorPage" from the API created and provided by the back-end developer. The response is received as JSON format and then assigned to "items" in the end. result is what has been returned from the API and rows is the list inside the API that has the desired data we want to show in our page, that is why the items is defined as a list in the beginning. as explained inside the code using comments, we are handling the errors in the early state of receiving the data in order to prevent swallowing exceptions from actual bugs in components. The Axios library [24] is used to handle fetching and posting the API data. as it is displayed in the function "handleButtonClick" (the button used to book the appointment), this library is used to post the chosen (by the user) data to back-end , which in this care is the doctor-id from API. This function activates whenever the user clicks on the booking button. It has been considered that ion every part of the process, from receiving to sending the data, that errors might happens, so error handling code has been written in order to protect the web application from crashing.

Lastly just a specific part of the JSX code is added in this document to show how this functions are called inside the JSX elements. From line 54 to 58 we can see how different data is retrieved from API (saved in the list "items" and retrieved individually as "item") and then in line 59 to 64 we have the button element added to activate the process of sending the data to server based on user's choice. This function is basically activated in line 60 and the function has been explained in the previous paragraph.

```
 1
 2 import axios from "axios";
 3
 4   const [error, setError] = useState(null);
 5   const [isLoaded, setIsLoaded] = useState(false);
 6   const [items, setItems] = useState([]);
 7
 8      useEffect(() => {
 9     fetch(
10       "https://12vwe151nh.execute-api.ap-south-1.amazonaws.com/dev/
   onlinedoctor?account_id=demo_account"
11     )
12       .then((res) => res.json())
13       .then(
14         (result) => {
15           setIsLoaded(true);
16           setItems(result.rows);
17         },
18         // explanation: it's important to handle errors here
19         // instead of a catch() block so that we don't swallow
20         // exceptions from actual bugs in components.
21         (error) => {
22           setIsLoaded(true);
23           setError(error);
24         }
25       );
26   }, []);
27   const handleButtonClick = (item) => {
28     const dataToSend = {
29       //sending data to the server
30       DoctorId: item.id,
31     };
32     axios
33       .post("/api/bookAppointment", dataToSend)
34       .then((response) => {
35         // handling he response in backend
36         console.log("Appointment booked successfully!");
37       })
38       .catch((error) => {
39         // if backend requests fail
40         console.error("Error booking appointment:", error);
41       });
42   };
43
44   if (error) {
45     return <div>Error: {error.message}</div>;
46   } else if (!isLoaded) {
47     return <div>Loading...</div>;
48   } else {

  return (
  <div className={DoctorPage_css.doc_container}>
              {items.map((item) => (
    <div className={DoctorPage_css.back_box}>
                      <h3>Doctor Name: {item.doctor_name}</h3>
                      <p>Specialty: {item.specialty}</p>
                      <p>Medical Reg No: {item.medicalreg_no}</p>
                      <p>Consultation Fee: {item.consultation_fee}
                      </p>
                      <button
                        onClick={() => handleButtonClick(item)}
                        className={DoctorPage_css.book_btn}>

                        Book an appointment
                      </button>

        </div>
    </div>
    );
```

Listing 10: Fetching and sending data from API

In the code example mentioned below in Listing 11, line 14, a token is assigned, which is later used to access the API which contains detailed information about doctors' availability. Also in lines 16 to 21 various date methods are used to extract year, month and date from the "date" object.

The "fetchDataForSelectedDate" function fetches appointment slots for the chosen date (by the user) from the API. The authorization key is used here to access this information.

Later in "generateNextSevenDays " function the next seven days are determined on the basis on current date and then showed in the page using jsx.

At the end the approval of this appointment is displayed to patients after the date submission.

There is also the option to cancel the whole process and return to doctor page using "useNavigate" in line 79.

For each and every function error handling function have been written to prevent any crash in the web application.

```
1
2 export function SlotSelection() {
3   const [error, setError] = useState(null);
4   const [isLoaded, setIsLoaded] = useState(false);
5   const [items, setItems] = useState([]);
6   const [selectedDate, setSelectedDate] = useState(null);
7
8   const location = useLocation();
```

```
const id = location.state.id;
const name = location.state.name;
const exp = location.state.exp;

const tokenKey =
  "Bearer <Authorization token>";

const getFormattedDate = (date) => {
  const year = date.getFullYear();
  const month = String(date.getMonth() + 1).padStart(2, "0");
  const day = String(date.getDate()).padStart(2, "0");
  return `${year}-${month}-${day}`;
};

const fetchDataForSelectedDate = (date) => {
  setIsLoaded(false);
  setSelectedDate(date);

  const formattedDate = getFormattedDate(date);
  const slotAPIurl = `https://betaemrapi.hlthclub.in/appointment/
 onlineappointment/getslots/${id}?date=${formattedDate}&account_id
 =demo_account&get_instant_slots=false`;

  fetch(slotAPIurl, {
    headers: {
      Authorization: tokenKey,
    },
  })
    .then((res) => {
      if (!res.ok) {
        throw new Error("Network response was not ok");
      }
      return res.json();
    })
    .then(
      (result) => {
        // Assuming the API response contains afternoon, evening,
 and morning slots
        setItems({
          afternoon: result.afternoon || [],
          evening: result.evening || [],
          morning: result.morning || [],
        });
        setIsLoaded(true);
      },
      (error) => {
        setIsLoaded(true);
        setError(error);
      }
    );
};
console.log(items);
useEffect(() => {
  fetchDataForSelectedDate(new Date()); // Fetch initial data for
 today's date
}, []); // Empty dependency array means this effect runs only once
 , on mount

const handleDateSelection = (date) => {
  fetchDataForSelectedDate(date);
};

const generateNextSevenDays = () => {
  const days = [];
  const today = new Date();
  for (let i = 0; i < 7; i++) {
    const nextDay = new Date(today);
    nextDay.setDate(today.getDate() + i);
    days.push(nextDay);
  }
  return days;
};
const handleFormSubmit = () => {
  window.alert("Your appointment has been booked.");
};
const navigate = useNavigate();
const handleCancelButton = () => {
  try {
    navigate("/doctorpage");
  } catch (error) {
    navigate("/error", { state: { message: "Failed to load the
 page" } });
  }
};

const daysArray = generateNextSevenDays();
if (error) {
  return <div>Error: {error.message}</div>;
} else if (!isLoaded) {
  return <div>Loading...</div>;
} else {
  return (
    <div>
      <header className="otp_header">
        <div className="otp_icon">
          <img src={cchLogo} alt="Connect and Care Hospital Logo"
 />
        </div>
      </header>
      <main className={SlotSelection_css.main_slot_selection}>
        <article className={SlotSelection_css.doc_box}>
          <div className={SlotSelection_css.img_doc}>
            <img src={img7} alt="Doctor Avatar created by AI" />
          </div>
          <div className={SlotSelection_css.doc_info}>
            <h3>Name : {name}</h3>
            <h4>Specialty : Internal Surgeon</h4>
            <h4>Years of Experience : {exp}</h4>
          </div>
        </article>
        <div>
          <h1>Slot Selection</h1>
          <div className={SlotSelection_css.days_box}>
            {daysArray.map((date) => (
              <button
                className={SlotSelection_css.day_btn}
                key={date.getTime()}
                onClick={() => handleDateSelection(date)}
              >
                {date.toLocaleDateString("en-US", {
                  weekday: "long",
                  month: "short",
                  day: "numeric",
                })}
              </button>
            ))}
          </div>
          <div>
            <p>
              Doctor time:{" "}
              {selectedDate.toLocaleDateString("en-US", {
                weekday: "long",
                month: "short",
                day: "numeric",
              })}
            </p>
            {/* Check if the morning time slots array is not empty before rendering */}
            {items.morning.length > 0 && (
              <div>
                <h3>Morning</h3>
                <ul className={SlotSelection_css.li_slots}>
                  {items.morning.map((timeSlot) => (
                    <li key={timeSlot}>{timeSlot}</li>
                  ))}
                </ul>
              </div>
            )}

            {/* Check if the afternoon time slots array is not empty before rendering */}
            {items.afternoon.length > 0 && (
              <div>
                <h3>Afternoon</h3>
                <ul className={SlotSelection_css.li_slots}>
                  {items.afternoon.map((timeSlot) => (
                    <li key={timeSlot}>{timeSlot}</li>
                  ))}
                </ul>
              </div>
            )}

            {/* Check if the evening time slots array is not empty
before rendering */}
            {items.evening.length > 0 && (
              <div>
                <h3>Evening</h3>
                <ul className={SlotSelection_css.li_slots}>
                  {items.evening.map((timeSlot) => (
                    <li key={timeSlot}>{timeSlot}</li>
                  ))}
                </ul>
              </div>
            )}
          </div>
        </div>
        <div>
          <button
            className={SlotSelection_css.slot_btn}
            onClick={handleCancelButton}
          >
            Cancel
          </button>
          <button
            type="submit"
            className={SlotSelection_css.slot_btn}
            onClick={handleFormSubmit} // Call the
handleFormSubmit function on button click
          >
            Submit
          </button>
        </div>
      </main>
      <div>
        <footer className="all_footer otp_footer">
          <div className="info">
            <p> 2023 cch.me Inc. All rights reserved.</p>
            <p>Emergency/Help Call</p>
            <p>780*********</p>
          </div>
        </footer>
      </div>
    </div>
  );
}
}
```

Listing 11: Fetching and sending data from API and displaying it based on current date

These are only short lines of the code that has been worked on so far to create the TelecarePLUS web application. For sure a telemedicine website is not just a one time coding task and it has to be updated regularly based on user feedback and functionality monitoring. However, considering the time and scope of the project, it can be said the main part of the project has been done in a short amount of time.

Moreover, apart from code snippets, some screenshots of the web application can also be found in the appendix section 8. The screenshot in Fig. 5 is the first landing page of the application. This page asks the user to choose a desired area. Based on the area chosen, the user is asked to choose a hospital from the second dropdown list. The objective of this page is to let the user choose the nearest hospital to their area for consultation, so that it is convenient for the patient to visit the hospital, if needed. However, the user can proceed with anything they like.

Fig. 6 and Fig. 7 just gives an insight to the features of this web application. These pages are static and meant to provide information to the end user.

The patient registration page is shown in Fig. 8 and Fig. 9. This form is meant to ask the patient for their details like name, age, gender, mobile, email and symptoms experienced. Based on the age of the patient(if less than 18), the form also asks the user to enter their guardian's name and mobile number. Proper validations for these input are in place to ensure that the user enters the correct and valid information to proceed forward using the Submit button.

The email address entered by the user in the registration page is prompted by a random six digit OTP that the user needs to enter correctly in the page shown in Fig. 10. This is to ensure that the user is requesting an appointment using a legitimate email. The same email will also be used to send the e-prescription and other related documents to the user after the teleconsultation is over.

On successful OTP verification, the user is redirected to a doctor page as shown in Fig. 11. This page lists down all the doctors for the hospital chosen by the user in the first page. The page has doctor cards with related information listed using a flip animation. The user can choose a doctor and proceed ahead to date and time selection.

The date and time selection page as shown in Fig. 12 and Fig. 13 shows the available slots for the selected doctor in the previous page. Current date is selected by default and the user can see the list of available slots for that doctor for the next seven days. After selecting a slot, the user can submit the request and the frontend code sends the request to an API URL that books the appointment for the user after proper data validations and returns a success message as shown in Fig. 14.

6 Conclusion and Future Works

In conclusion, the TelecarepLUS project has achieved its goal in creating a seamless and elegant website, providing an environments for the patients to receive online health services. The project was planned and research questions were set at the beginning and in the end all the questions were answered and desirable result was delivered.

By ensuring that the best technologies were utilized and a user-friendly design was implemented, an intuitive and spectacular interface was designed. The success in meeting all the required objectives and overcoming the challenges along the way is a proof for the dedication of the team and using the right technologies and methods used in implementing the project.

Looking ahead for the future and enhancing the capabilities of the telecarePLUS project, multiple future works has been considered to be added in the website which are as follows:

- **Multilingual Support**: Integrating the website using multiple languages will be a valuable addition for the website. It helps improving the communication and assisting patients with different background from different regions.
- **Ease of access for visually impaired individuals**: Despite of the fact that this aspect was considered during the implementation and some features (adding extra information for text to speech features) were added to the code, additional measures will be incorporated into the project.
- **AI-based diagnosis**: Implementation of preliminary AI diagnosis can be a revolutionary asset for the telecarePLUS project, providing fast and proper medical advice, based on the patient's medical history and the symptoms they declare in the website.
- **Monitoring Devices**: Integrating the wearable devices and remote monitoring tools into the application to assist patients track their health status and sharing this information with the doctors using telecarePLUS website.
- **Use feedback and make improvements**: Use the feedback received by real patients and make improvements based on the patient's needs and requirements. Conducting a survey and involving the patients in the process are few examples of how this will be done.

7 Contribution

The goals of this project report is to establish the case for the TelecarePLUS web application, as an adaptable and favorable platform for patients. The notable contributions of TelecarePLUS are elucidated as follows:

- **Core Features:** TelecarePLUS provides multiple features that simplifies accessing healthcare and enables users to receive consultation services utilizing different platforms and devices. Throughout the implementation process, users with different conditions have been considered and additional considerations have been designed to simplify the process for them.
- **User-Friendly Interface and Responsive Design:** It is ensured that patients, in all ages are able to easily navigate through the web application and benefit from the services. TelecarePLUS is designed in a way that welcomes patients with an intuitive and trusting interface and enables them to access the platform wherever they are regardless of what device they use. Patients will have a seamless and consistent experience as they receive their desired service.
- **Secure Data Transfer:** During the implementation phase, one of the primary focuses was to establish a secure data transfer between front-end and back-end which was implemented and achieved successfully. The patient's information are communicated and transferred safely and it is safeguarded from different threats and attacks.
- **Tackling Issues with an Innovative Approach:** The primary goal of this project was to remove the obstacles posed by traditional healthcare services and provide a simple yet impeccable platform for the patients regardless of the time or location.
- **Future Work:** Looking ahead, this project report argues the different possibilities for TelecarePLUS and its impacts on the healthcare system. All the future directions and designated plans are designed in such a way that enables TelecarePLUS to expand and have a revolutionizing impact on healthcare.

8 Appendix: Screenshots

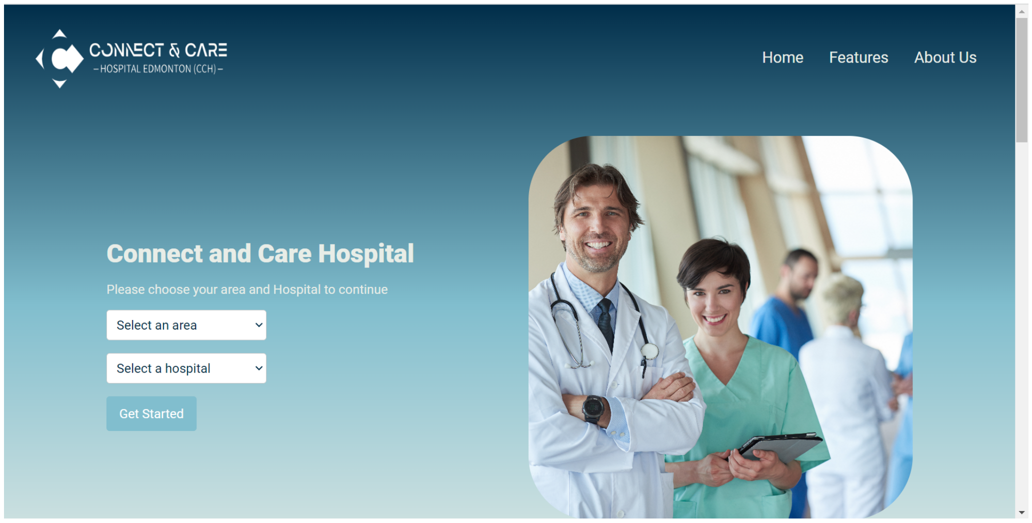

CONNECT & CARE
– HOSPITAL EDMONTON (CCH) –
Home
Features
About Us
Connect and Care Hospital
Please choose your area and Hospital to continue
Select an area
Select a hospital
Get Started

Figure 5: TelecarePLUS - Home page

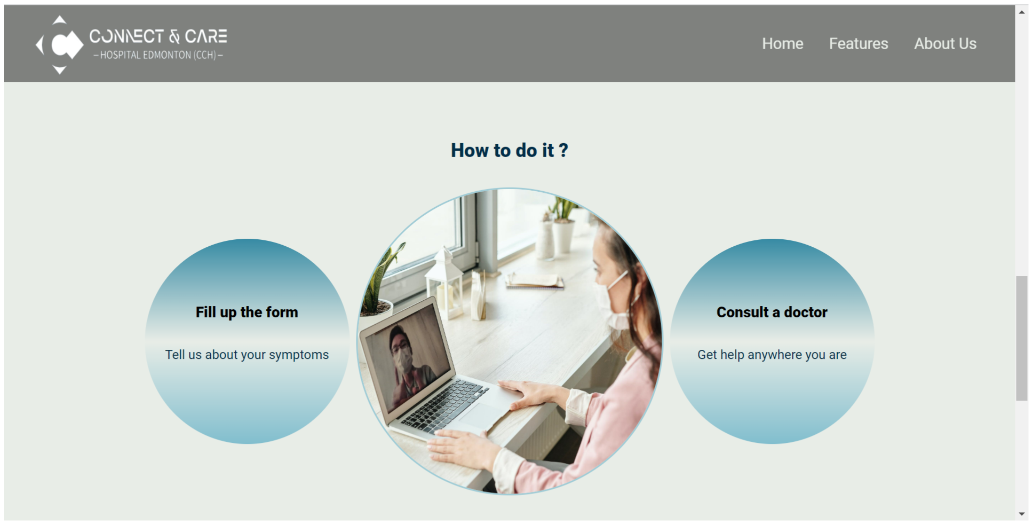
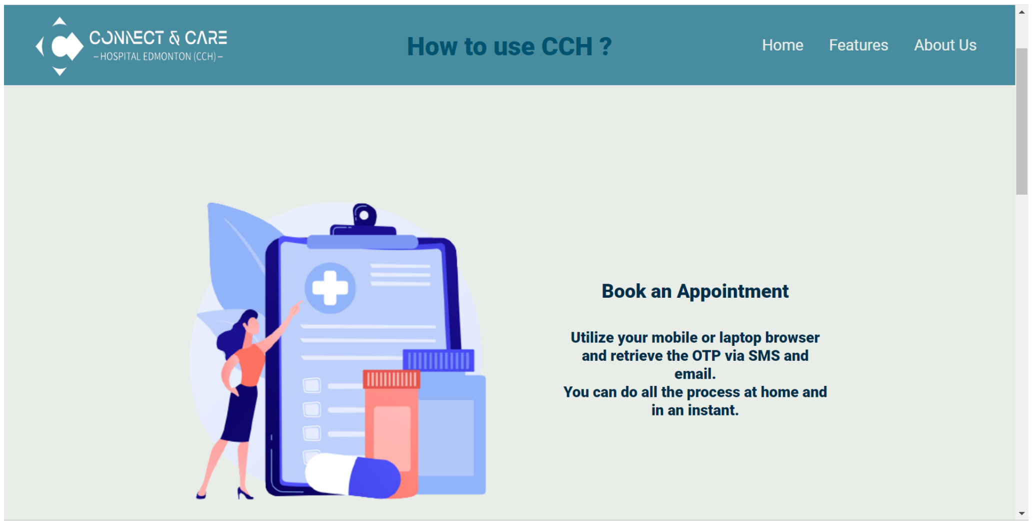


Figure 6: TelecarePLUS - Home page

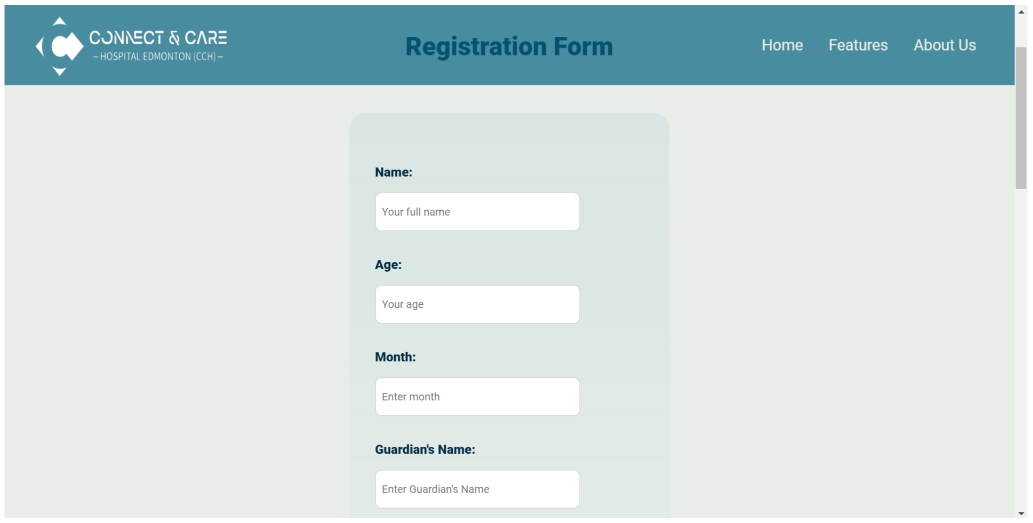


How to use CCH ?

Book an Appointment

Utilize your mobile or laptop browser and retrieve the OTP via SMS and email.
You can do all the process at home and in an instant.

Figure 7: TelecarePLUS - Features page



Registration Form

Name: Your full name

Age: Your age

Month: Enter month

Guardian's Name: Enter Guardian's Name

Figure 8: TelecarePLUS - Patient Registration Form

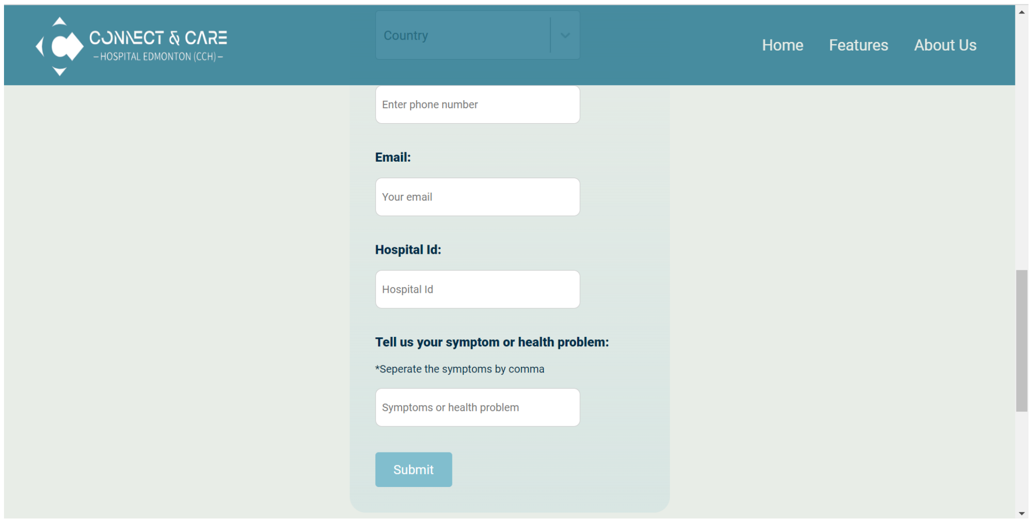

CONNECT & CARE
– HOSPITAL EDMONTON (CCH) –
Country
Home
Features
About Us
Enter phone number
Email:
Your email
Hospital Id:
Hospital Id
Tell us your symptom or health problem:
*Seperate the symptoms by comma
Symptoms or health problem
Submit

Figure 9: TelecarePLUS - Patient Registration Form

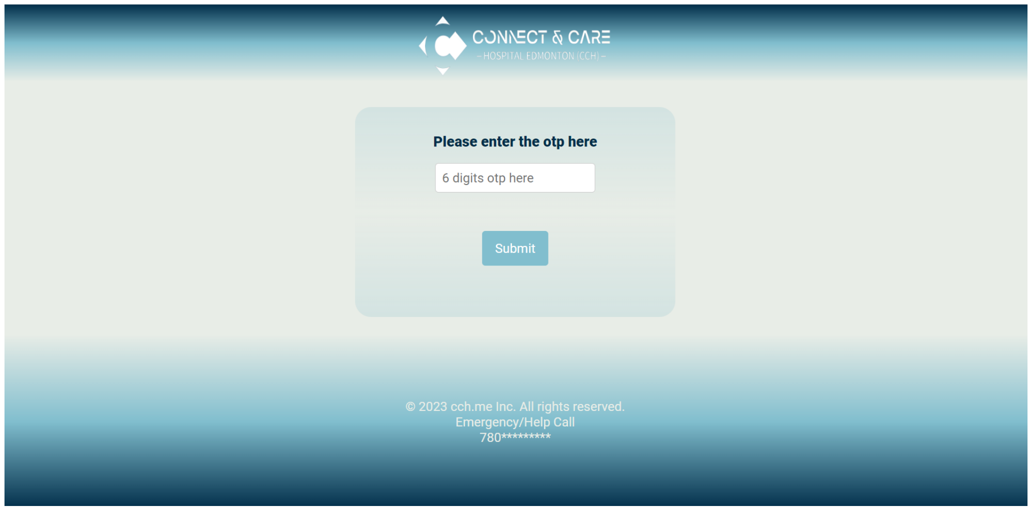

CONNECT & CARE
– HOSPITAL EDMONTON (CCH) –
Please enter the otp here
6 digits otp here
Submit
© 2023 cch.me Inc. All rights reserved.
Emergency/Help Call
780*********

Figure 10: TelecarePLUS - OTP Verification page

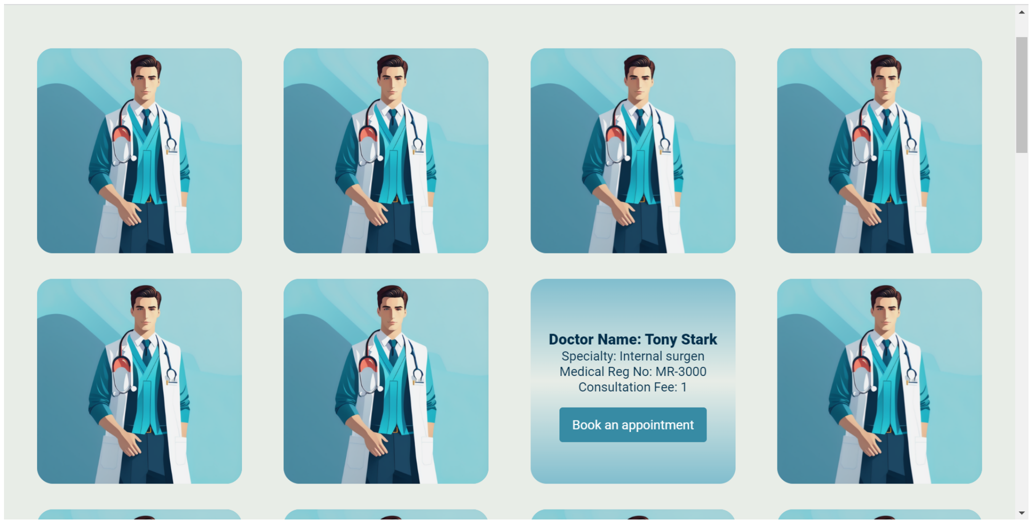
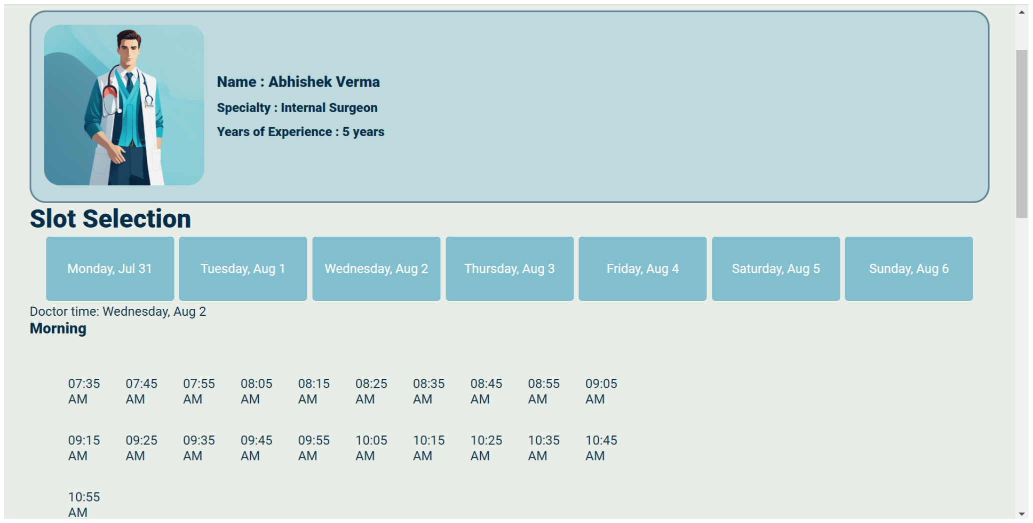


Figure 11: TelecarePLUS - Doctor Selection page


Name : Abhishek Verma

Specialty : Internal Surgeon

Years of Experience : 5 years

Slot Selection

Monday, Jul 31 | Tuesday, Aug 1 | Wednesday, Aug 2 | Thursday, Aug 3 | Friday, Aug 4 | Saturday, Aug 5 | Sunday, Aug 6

Doctor time: Wednesday, Aug 2

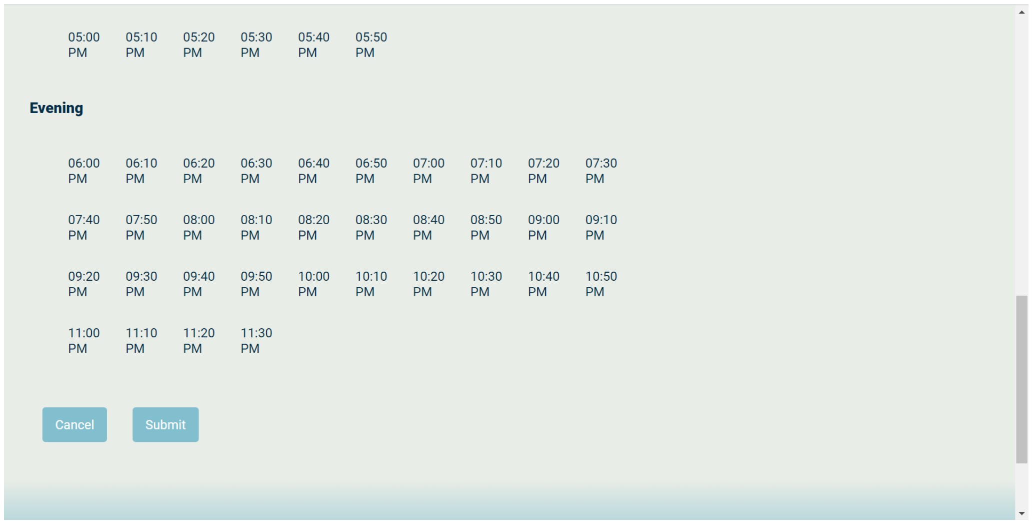
Morning

07:35 AM, 07:45 AM, 07:55 AM, 08:05 AM, 08:15 AM, 08:25 AM, 08:35 AM, 08:45 AM, 08:55 AM, 09:05 AM

09:15 AM, 09:25 AM, 09:35 AM, 09:45 AM, 09:55 AM, 10:05 AM, 10:15 AM, 10:25 AM, 10:35 AM, 10:45 AM

10:55 AM

Figure 12: TelecarePLUS - Slot Selection page


05:00 PM 05:10 PM 05:20 PM 05:30 PM 05:40 PM 05:50 PM

Evening

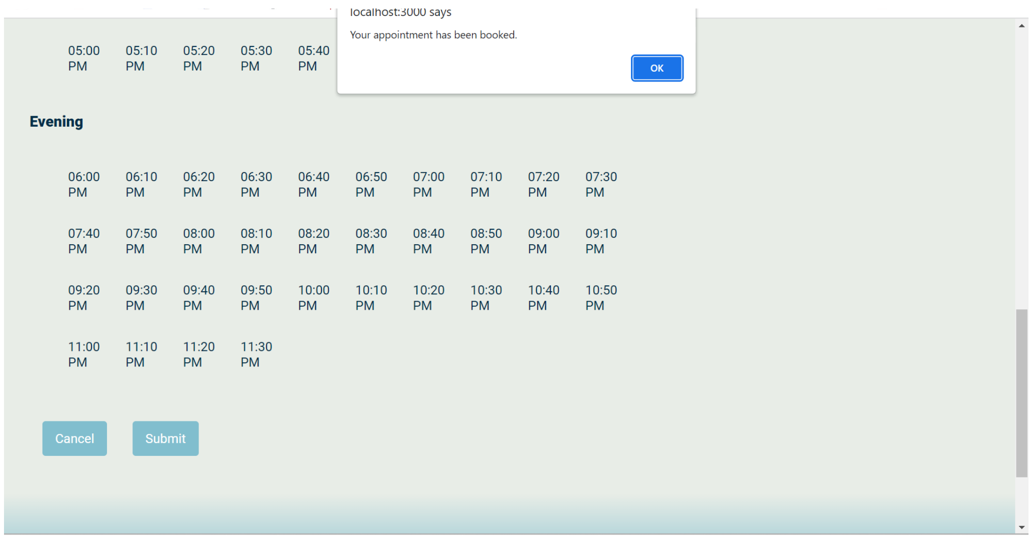
06:00 PM 06:10 PM 06:20 PM 06:30 PM 06:40 PM 06:50 PM 07:00 PM 07:10 PM 07:20 PM 07:30 PM

07:40 PM 07:50 PM 08:00 PM 08:10 PM 08:20 PM 08:30 PM 08:40 PM 08:50 PM 09:00 PM 09:10 PM

09:20 PM 09:30 PM 09:40 PM 09:50 PM 10:00 PM 10:10 PM 10:20 PM 10:30 PM 10:40 PM 10:50 PM

11:00 PM 11:10 PM 11:20 PM 11:30 PM

Figure 13: TelecarePLUS - Slot Selection page


localhost:3000 says

Your appointment has been booked.

05:00 PM 05:10 PM 05:20 PM 05:30 PM 05:40 PM

Evening

06:00 PM 06:10 PM 06:20 PM 06:30 PM 06:40 PM 06:50 PM 07:00 PM 07:10 PM 07:20 PM 07:30 PM

07:40 PM 07:50 PM 08:00 PM 08:10 PM 08:20 PM 08:30 PM 08:40 PM 08:50 PM 09:00 PM 09:10 PM

09:20 PM 09:30 PM 09:40 PM 09:50 PM 10:00 PM 10:10 PM 10:20 PM 10:30 PM 10:40 PM 10:50 PM

11:00 PM 11:10 PM 11:20 PM 11:30 PM

Figure 14: TelecarePLUS - Appointment Booking Successful

References

[1] Grigsby, J., Kaehny, M. M., Sandberg, E. J., Schlenker, R. E., & Shaughnessy, P. W. (1995). Effects and effectiveness of telemedicine. Health care financing re- view, 17(1), 115–131. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4193577/

[2] Haleem, A., Javaid, M., Singh, R. P., & Suman, R. (2021). Telemedicine for healthcare: Capabilities, features, barriers, and applications. *Sensors international, 2*, 100117. DOI: https://doi.org/10.1016/j.sintl.2021.100117

[3] Top Frontend Technologies and their usage. By Anusha SP. https://www.knowledgehut.com/blog/web-development/front-end-technologies-list

[4] Telehealth Website Design: Best Principles to Consider. https://www.syntacticsinc.com/news-articles-cat/telehealth-website-design-best-principles-consider/

[5] The 7 Phases of Web Development Life Cycle. monocubed. https://www.monocubed.com/blog/web-development-life-cycle/

[6] Successful development of telemedicine systems-seven core principles. Peter Yellowlees. https://journals.sagepub.com/doi/abs/10.1258/1357633971931192

[7] 6 Reasons Telehealth Is Now More Important Than Ever by Craig Guillot. https://healthtechmagazine.net/article/2020/05/6-reasons-telehealth-now-more-important-ever

[8] Benefits of Telemedicine. Brian William Hasselfeld, M.D. https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/benefits-of-telemedicine

[9] Telemedicine Market Size, Share & Trends Analysis Report By Component (Products, Services), By End-user (Patients, Providers), By Application, By Modality, By Delivery Mode, By Facility, And By Segment Forecasts, 2023 - 2030. https://www.grandviewresearch.com/industry-analysis/telemedicine-industry

[10] AJAX and APIs official documentation. https://legacy.reactjs.org/docs/faq-ajax.html

[11] How To Use Axios With React. Reed Barger. https://www.freecodecamp.org/news/how-to-use-axios-with-react/

[12] Writing Markup with JSX. https://react.dev/learn/writing-markup-with-jsx

[13] React JSX. w3schools. https://www.w3schools.com/react/react$_j$sx.asp

[14] Optimizing Performance in React official documentation. `https://legacy.reactjs.org/docs/optimizing-performance.html`

[15] Telemedicine Website Design: Top 5 Tips for Telehealth Product Design in 2023 by Maria Shapovalova. `https://www.voypost.com/blog/telemedicine-website-design`

[16] Leaders in Healthcare, Website Design and Marketing. Website Design, E-Commerce, and Digital Marketing `https://www.solution21.com/blog-Importance-of-responsive-web-design-for-medical/`

[17] Why Your Medical Practice Website Needs A Mobile Responsive Design BY SUYOGYA TRIVEDI. `https://practicetechsolutions.com/blog/medical-practice-website-needs-a-mobile-responsive-design/`

[18] AOS. Animate On Scroll Library. Michał Sajnóg. `https://michalsnik.github.io/aos/`

[19] AOS - Animate on scroll library. Michał Sajnóg. `https://www.npmjs.com/package/aos`

[20] React useEffect Hooks. w3schools. `https://www.w3schools.com/react/` react$_u$seeffect.asp

[21] Components and Props official documentation. `https://legacy.reactjs.org/docs/components-and-props.html`

[22] Using the State Hook official documentation. `https://legacy.reactjs.org/docs/hooks-state.html`

[23] useState in React: A complete guide. Esteban Herrera. `https://blog.logrocket.com/guide-usestate-react/`

[24] axios-http.com. Axios. Promise based HTTP client for the browser and node.js. `https://axios-http.com/docs/intro`