CS 4610 Project Report

Easy Find Doctors – A web system to help choose a health provider based on your personal needs and make an appointment easily.

November 29, 2021

Shaojing Tian

Table of Contents

Introduction

Easy Find Doctors with its URL http://54.164.36.159/EasyFindDoctors/Login.jsp is a web application aimed at serving as a one-stop web platform for users to search for health providers based on their customized needs and preferences such as doctor's specialty, gender, insurance network, and location, and make/cancel/reschedule an appointment with the doctor. In that case, it helps patient save time and effort of browsing multiple websites to find a doctor, such as clinic's website to search based on specialty and location, and insurance company's website to find out if the doctor is within the network of the patient's insurance type. With a doctor being selected from a filtered list according to patient's needs, patient can make an appointment from the doctor's database of available date and time. Patient can also access to his/her own database to view profile and upcoming appointments and cancel or reschedule an appointment if needed.

System Overview

The project was implemented through a three-tier web application design for its front-end presentation, server-side business logic, and the database. For the front-end presentation, an EC2 instance with Apache Web Server running is utilized to deploy static web content created through HTML, CSS and JavaScript. The content of the front-end is presented to the clients and receive user requests from the web interaction. Another EC2 instance with Apache Tomcat running is used to process the dynamic requests that trigger calls to the server-side Java Servlet to manipulate the database and return responded data back to the front-end user. The third tier is the database that is stored in MySQL in the EC2 instance with Apache Web Server running. The data can be retrieved by queries and maintained by updates through the connection between Java servlets and MySQL.

Design Overview

The main functionality of the web application is to allow users to filter doctors based on their selection of a doctor's specialty, gender, location and insurance network which is stored in a doctor dataset. With a particular doctor being selected by user, the available data and time slot

information is to be retrieved and shown to user from the doctor appointment dataset. Once an appointment is scheduled/cancel/rescheduled by user, the date and time information is linked and updated with the patient's profile and upcoming appointments dataset, and also with the doctor appointment dataset. User authentication and profile retrieval can be undertaken through access to the user dataset.

Related Works

Clinics and hospitals have their own websites for user to browse health providers, such as www.batonrougeclinic.com and www.ololrmc.com. However, most of the search are done by condition/specialty and location/zipcode, which makes a patient has to call or go the insurance company website to find out if a doctor is within the patient's insurance network. Insurance network is very important especially for those who are looking for surgery or inpatient health care services. The Easy Find Doctors aims at providing a one-stop platform for patient to easily find a doctor based on their needs and preferences.

Other criteria of reviews, ratings, and a doctor's years of experience is also important when it comes to selection of doctors. But due to limitation and time constraint of this project, we are still looking to a feasible way to collect all the information and compile them into the structured database. It leads the project in the direction for future implementation.

System Architectural Design

Chosen System Architecture

The system architecture of the project adopts the three-tier web application design. Two EC2 instances were utilized with one to run Apache Web Server hosting static web contents and the other to run Tomcat Application Server processing dynamic user requests. Apache Web Server and Tomcat Server were connected through mod_jk so that requests to host website are routed to Tomcat server worker. JSP files written with HTML, CSS, Bootstrap and Java codes allows for the dynamic content injection into static web contents. Java Servlets were used as the back-end server to get parameters from user's input, then retrieve corresponding data after

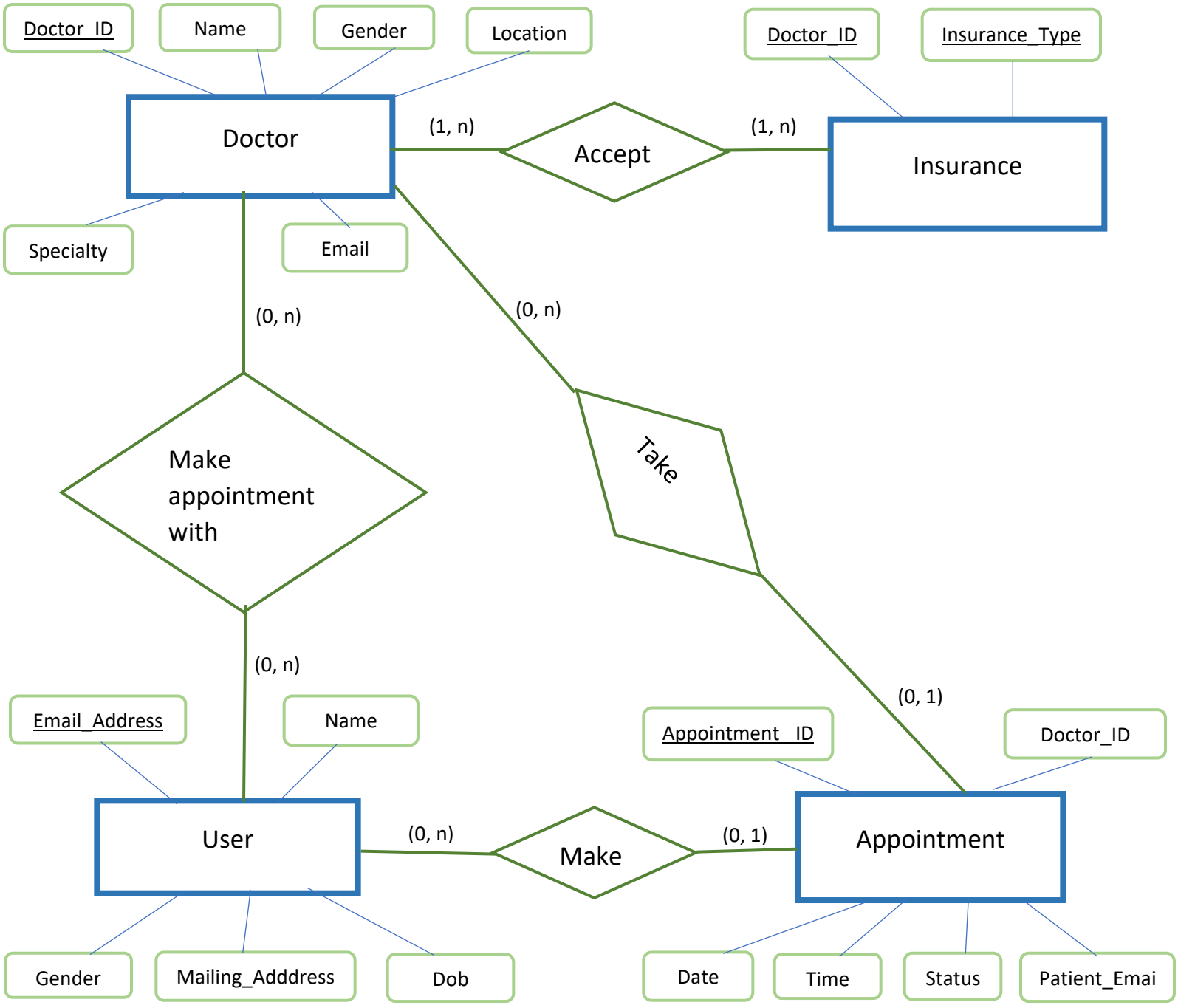
connected with the database, and inject the retrieved data to the next dispatched page. That realized dynamic interaction between front-end clients and the server-side. Images and other static contents were stored in Apache Web server. The third tier of MySQL database was connected with Java Servlets using Java Database Connectivity (JDBC).

Interface HttpSession is an important functionality in Java Servlet which provides a way to identify a user across more than one page request or visit and store information about that user. The servlet container uses this interface to create a session between a client and a server. The session persists for a certain period of time across more than one connection or page request from the user.

JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates the core functionality common to JSP applications. The operation of looping implemented by JSTL was substantially used in this project to display search results of doctors and list of patient's upcoming appointments.

MySQL was utilized for storage and manipulation of data because of the advantage of Relational Database Management System and its integration with the servlets using JDBC. The design of ER model and rational data tables are shown below.

ER model


Doctor_ID
Name
Gender
Location
Doctor
Specialty
Email
(1, n)
Accept
(1, n)
Doctor_ID
Insurance_Type
Insurance
(0, n)
Make appointment with
(0, n)
Take
(0, 1)
Email_Address
Name
User
Gender
Mailing_Adddress
Dob
(0, n)
Make
(0, 1)
Appointment_ID
Doctor_ID
Appointment
Date
Time
Status
Patient_Emai

Relational Tables

Relational table: Doctor

The Doctor table store basic data about doctors. The primary key is Doctor_ID which is used to uniquely identify a doctor.

doctor_ID	int
name	varchar(50)
email	varchar(50)
phone	char(12)
specialty	varchar(20)
location	varchar(50)
gender	char(1)

Relational table: User

The User table store basic data about registered users. The primary key is user's registered Email address which is used to uniquely identify a user.

email_address	varchar(50)
password	varchar(50)
name	varchar(50)
dob	date
gender	char(1)
mailing_address	varchar(100)

Relational table: Doctor Accepted Insurance

The Doctor Accepted Insurance used to store all types of insurances accepted by each doctor. The primary key is the composition of Doctor ID and an Insurance type.

doctor_ID	int
insurance_type	varchar(100)

Relational table: Appointment

The Appointment table is used to store all date and time slots for each doctor, including both booked and vacant appointments. The primary key is appointment_ID assigned to each date time slot for each doctor. Booked_status is field with Boolean value, if the status is "Y", the patient_email field should be set with the email address with which the patient booked an appointment.

appointment_ID	int
doctor_ID	int
date	date
time	varchar(10)
booked_status	char(1)
patient_email	varchar(50)

Detailed Description of Components

The following section is a detailed description of how each of the functions in the Easy Find Doctors http://54.164.36.159/EasyFindDoctors/Login.jsp work with it Java Servlet in Tomcat Server, and how the Servlet processes the requested information from user, interacts with database, and injects the dynamic information to the next dispatched page.

Sign In

The login Servlet get parameters from the user's input of Email address and password, and use the UserDAO class method the decide if the user's Email and password are both existent in the patient database. If they are, the servlet will create a HttpSession on the server for the client. The user's all information are set as attributes to the user's session, the those information can be retrieved in following pages in needed. Then the user will be directed to his/her home page where one can navigate to other website features.

Create Account

The Register Servlet get parameters of all user's input information of Email, full name, date of birth, gender, mailing, address, and password from the Create Account page. The Servlet connects with the patient database, and find if the Email address already exist in the database, if it is, a message will pop up to the user asking to enter a different Email address. Otherwise, the servlet creates a HttpSession to the user, and directs the user to the home page.

Search Doctors

The Search Doctor Servlet get parameters of user's choice of doctor's specialty, accepted insurance, location and gender and filters the doctors whose criteria satisfy the user's input of choices by connecting with the doctor's database and execute the SQL query. Iterate through the doctors, and put the id, name, email, and phone of each doctor to each HashMap. Then add the list of mapped information of the doctors (id, name, email, phone) to an Attribute to session. Then forward it to the destination page where the list of search results will be displayed with each doctor having a select button.

Show Appointment Date and Time

Once a user clicked the "select" button to choose a particular doctor, The Doctor To Appointment Servlet get the parameter of the doctor ID stored as a value in the "select" button. Then retrieve the doctor's name from the doctor database with the ID, and name will be displayed on the making appointment page where the user is directed to on the next step.

Make Appointment

The Make Appointment Servlet get parameters of the user's selected date and time from the making appointment web page. Check on the appointment database whether the booking status of the data and time slot with the certain doctor is already booked, if it is a message will pop up to ask the user to choose another date or time. Otherwise, update the appointment database with the status_booked field set to "Y" and the patient_email field to the user's Email address, and direct the user to the appointment booked confirmation page.

Get Appointment

Once the user click the “view upcoming appointment” button, The Get Appointment Servlet get parameter of the user’s Email address from the Session object, then from the appointment database, acquire the appointment ID, doctor ID, the doctor’s name, appointment date, and appointment time with the user’s Email address that an appointment is booked with. If there is any result, that means there is an appointment booked with this Email address. Then put those collected data to an upcoming appointment HashMap, add all mapped information to a list if there are more than one appointment result. Then direct the user to the upcoming appointment display web page.

Ask Confirmation to Cancel Appointment

As the user click the “cancel” button from one of the upcoming appointments, the Cancel Appointment Servlet gets attribute of the appointment ID from the session object, and refer to the appointment ID to retrieve the doctor’s name, appointment date and time from database, and display those information to the user and ask him/her to confirm the cancellation on the next page.

Cancel Appointment

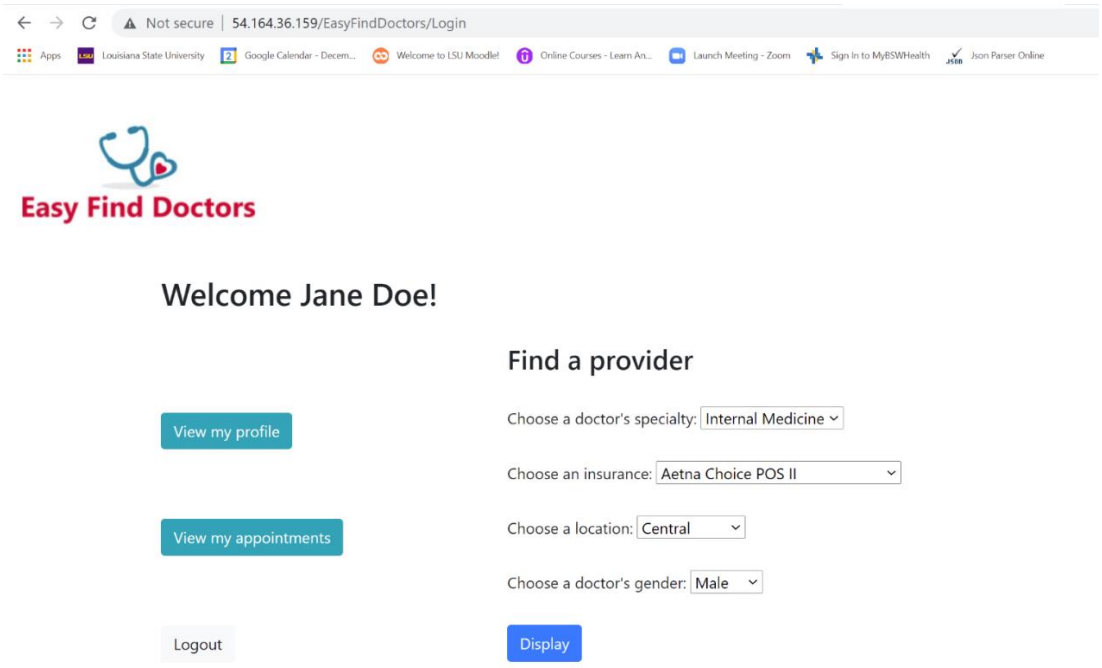
Once the user confirms the cancellation, the Cancel Appointment Servlet refers to the appointment ID to update the status_booked field to “N” and the patient_email field to “Null” in the appointment database, and directs the user to the appointment cancellation message page.

Log Out

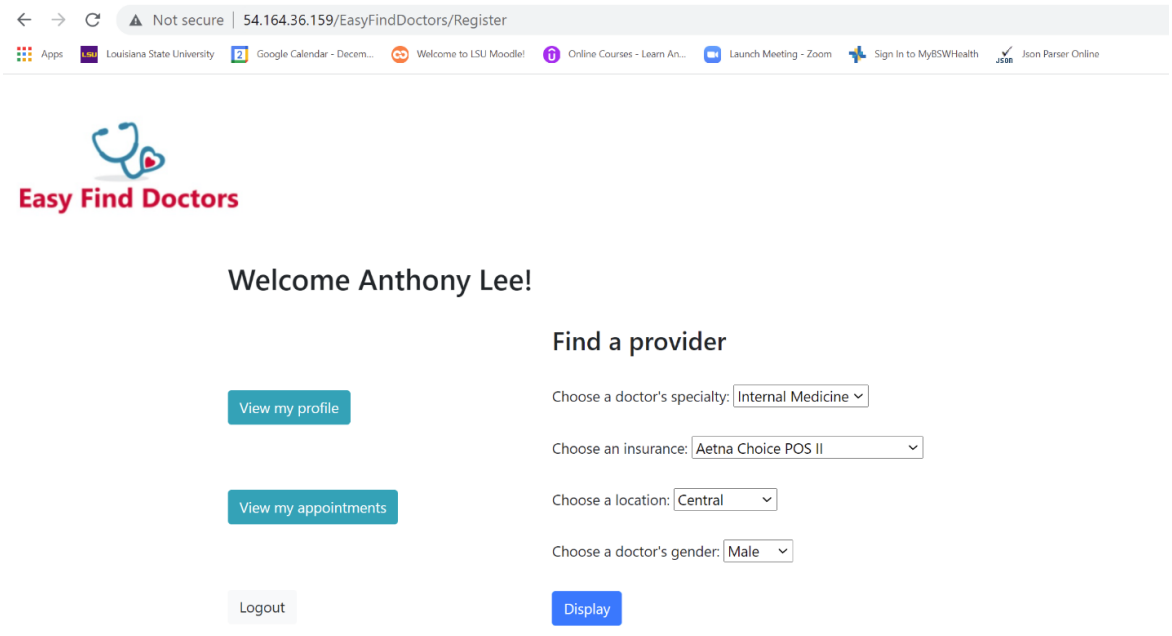
Once the user clicks the “Log out” button, invalidate the session object assigned to the user, and direct the user back to the Login page.

Screenshots of the response of each servlet

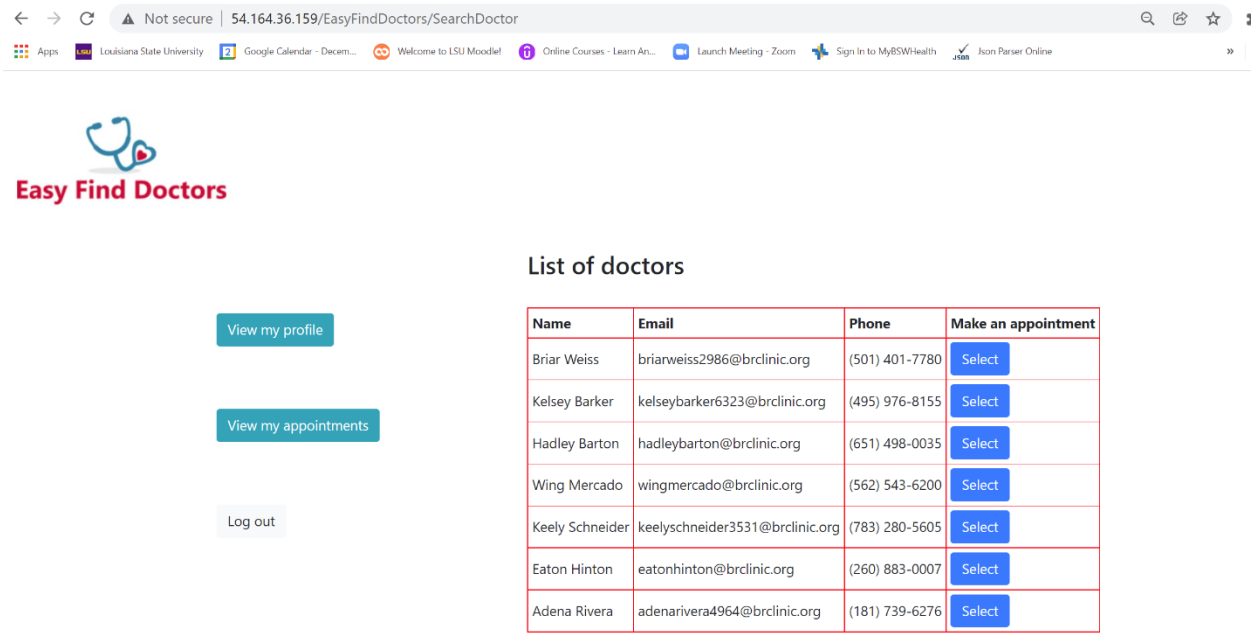
Sign In



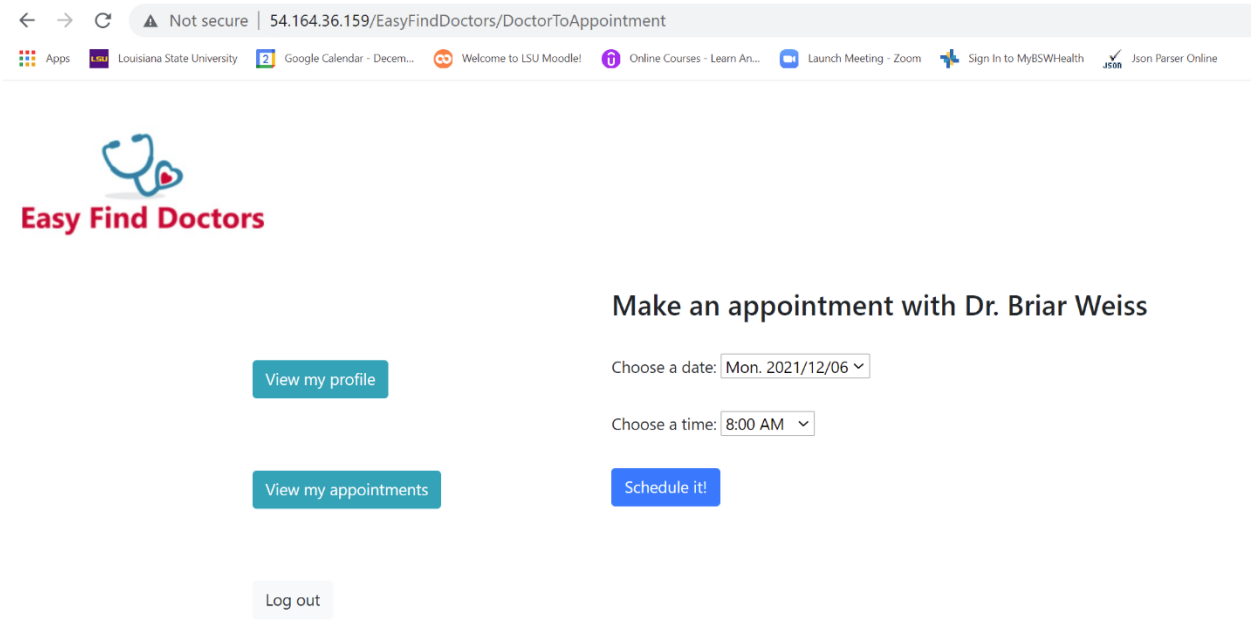
Create Account



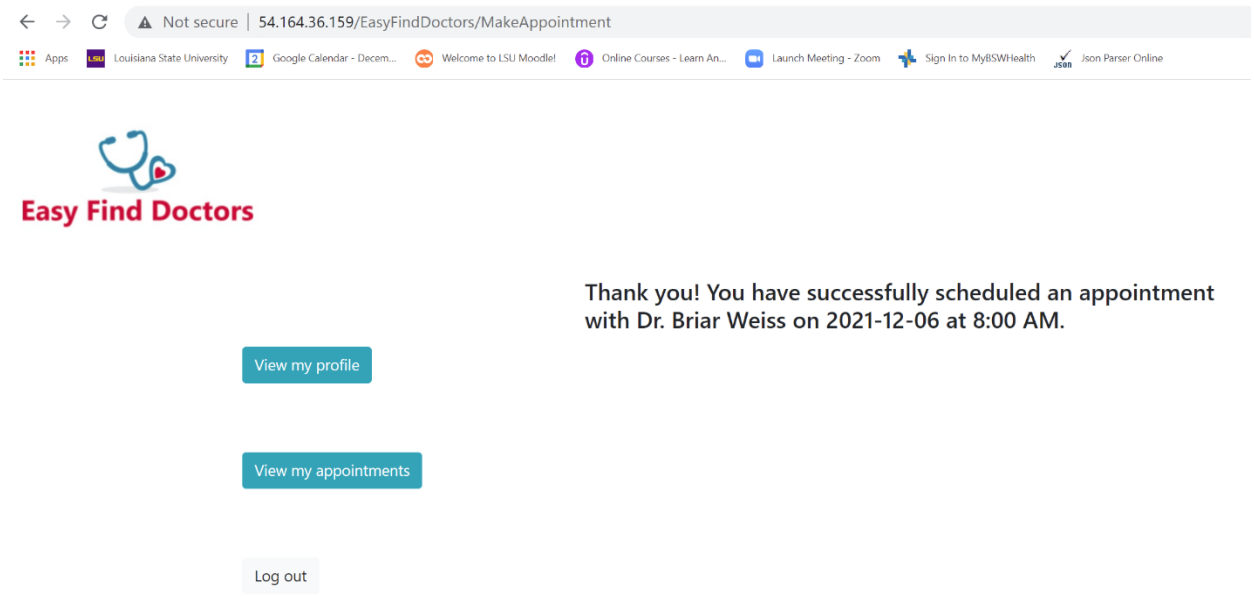
Search Doctors



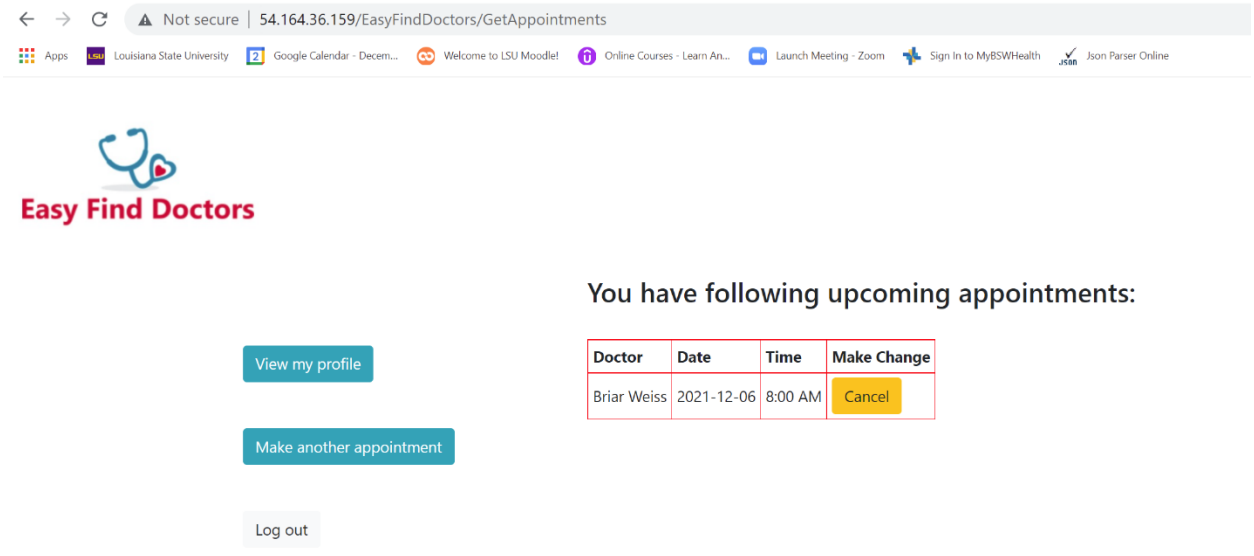
Show Appointment Date and Time



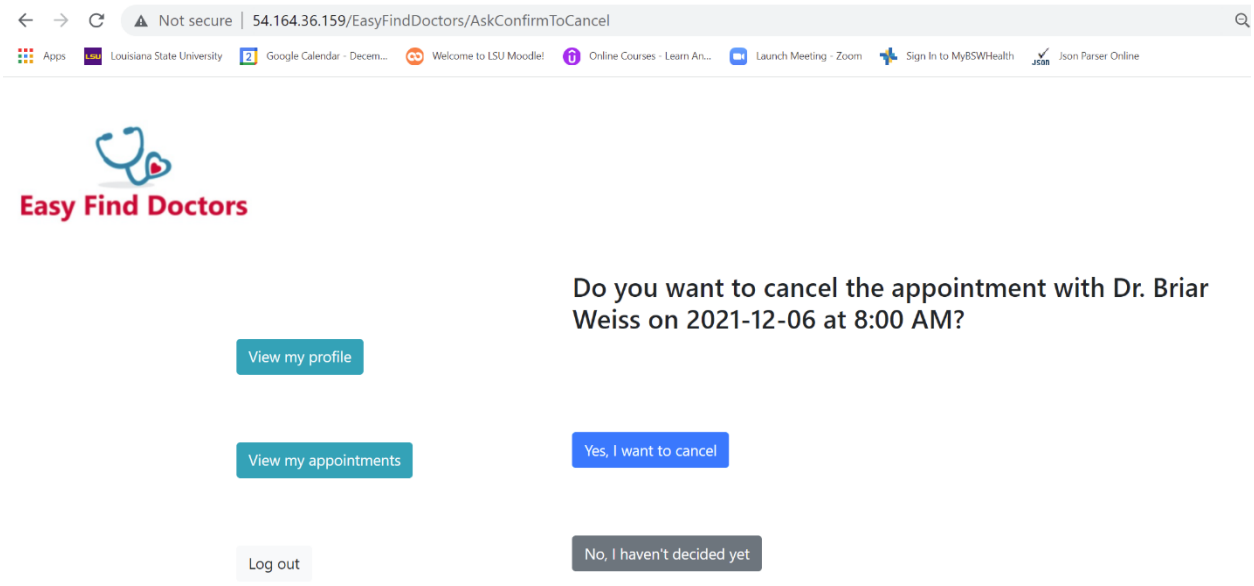
Make Appointment



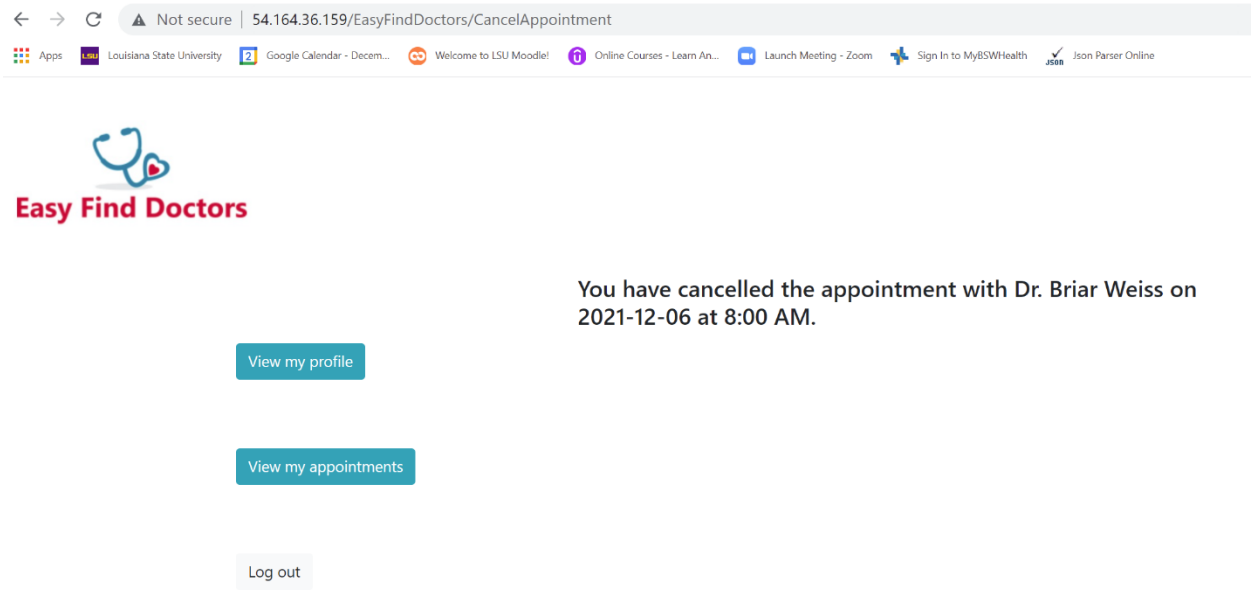
Get Appointment



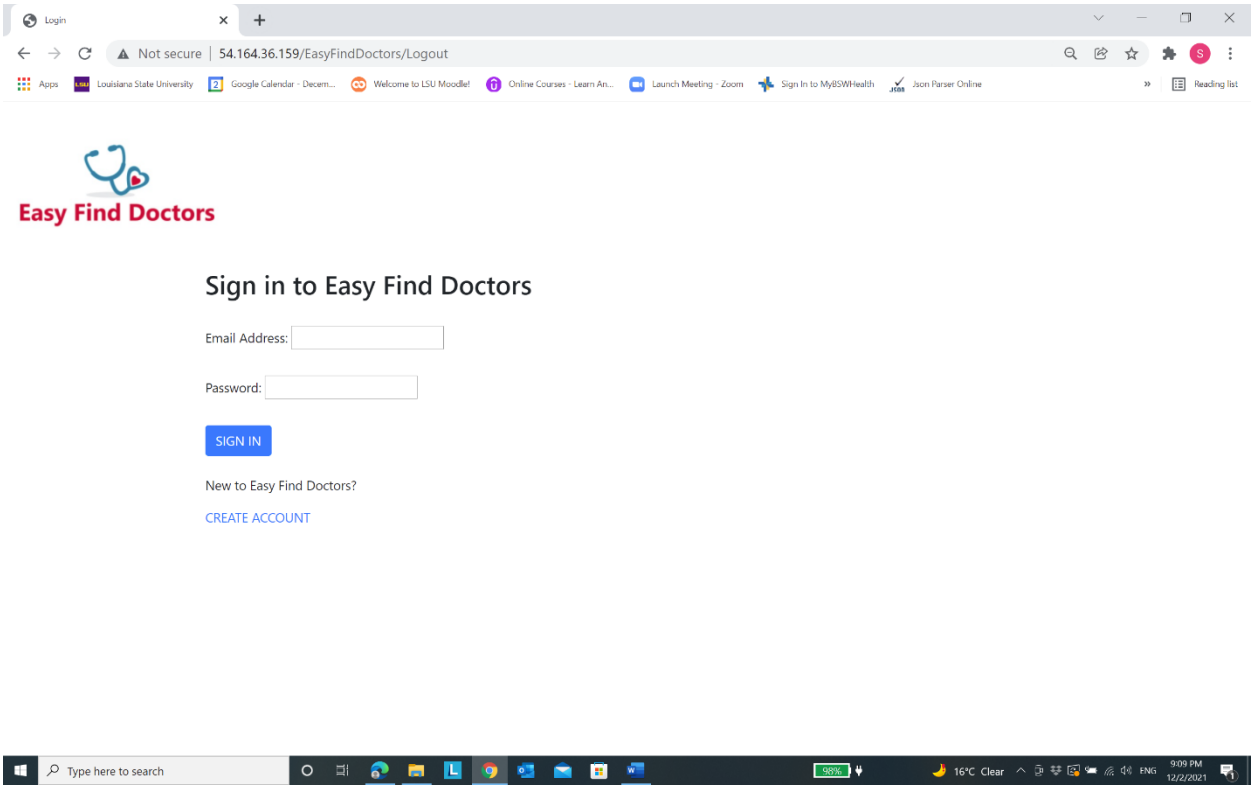
Ask Confirmation to Cancel Appointment



Cancel Appointment



Log Out



Conclusion

Easy Find Doctors http://54.164.36.159/EasyFindDoctors/Login.jsp is a three-tier web application that provides a platform for users to easily search a doctor according to their customized needs and preferences, and make/cancel/reschedule an appointment with doctors. The website can run free of bugs and responsively with a standard WIFI speed. At this point of the project, it is using auto-generated datasets for the database due to the limited accessibility to real doctors data. We are looking to collect real data from health care resources if there is a feasible way to achieve it. For future improvement, doctors' reviews, ratings, and years of experience are to be included in the database, and serve as important criteria for users to select a doctor.