Documentation for Hispanic Healthcare Directory:  
  
Compiling and Installing from Source

To compile the project from its source code, download the full source from https://github.com/HenrySmith3/GeorgiaHealthcareInformation. Then, open it in your IDE of choice (we used Intellij while developing, other options include myEclipse and netbeans) and set up the source as a new project. Before actually being able to deploy the project, you will need a few prerequisites. You will have to install apache and tomcat from <http://tomcat.apache.org/>. These can be installed separately, but the far easier way to handle this is to just use XAMPP, which you can download from <http://www.apachefriends.org/en/xampp.html>. XAMPP will install tomcat and apache, as well as configure everything for you. You will also need a few libraries, namely mysql:mysql-connector-java:5.1.92 and net.sourceforge.json:com.springsource.net.sf.json:2.2.22. Both of these libraries are easily found in maven, but you can google them if your IDE does not easily support maven.

You will want to set up a run configuration for tomcat from within your IDE. Below are step by step instructions for setting this up in Intellij. Instructions for other IDEs should be fairly similar:

1. You should already have the tomcat plugin. If you do not, open up Settings, go to Plugins, hit Browse Repositories, and select "Tomcat and Tomcat EE Integration".

Open up the run configurations and click edit configurations. It is the dropdown box next to the run button.

2. Hit the green plus sign to add a new configuration, then select Tomcat Server, then Local. Name it something, I just named it “tomcat.”

3. Down at the bottom it should say "warning: no artifacts marked for deployment". Hit the fix button next to it.

4. From there, it should add the "Main:war exploded" artifact to the project. Hit apply, and you should be good to go.

5. When you run it, you should have XAMPP also running. Apache and MySQL should be running from XAMPP, Tomcat should not (that's handled by Intellij).

6. From there, when you hit run, Intellij should handle deploying everything and then open your browser to localhost:8080/, which is the project’s root directory.

Installing from War

If you are just trying to install from the compiled .war file, you just have to have tomcat set up and give it the war. Setting up tomcat is still best handled using XAMPP (<http://www.apachefriends.org/en/xampp.html>). To deploy the war, simply go to tomcat’s install directory and place the .war file in the ‘webapps’ folder. From there, just start tomcat and point your browser to “localhost:8080”, and it should start up.

Tomcat Overview

We chose tomcat because it is one of the oldest and most reliable server technologies. Tomcat works by deploying java servlets for server-side applications and java servlet pages for client-side applications. It is fairly easy to get set up with Tomcat in any operating system, and it is old enough to have very good community support and documentation. For the purposes of this project, you only really need to know that Tomcat uses JSPs and servlets, and is the way that we host our server for our project. However, if you would like to learn more about Tomcat, you can check out <http://tomcat.apache.org/>.

JSPs Overview

JSPs are how Tomcat handles client side code. The basic idea behind JSPs is that anything that is valid HTML is a valid JSP. This means that any normal client side code will work within Tomcat’s framework. JSPs also provide the ability to embed java code directly into the client. There are a few special tags to allow you to start writing Java code which just look like slightly strange HTML tags. This is a big advantage because if you need to do something that can’t be handled easily through simple HTML and javascript, you can simply open a java tag with <% and write whatever code needs to be written, then close the tag with %> and continue with regular HTML and javascript. We don’t use these JSP specific tag for very much in our code. We use it to interpret the results in the hospital search results page and to programatically put the web page in the correct language. The way we handle having an English version of the web page and a Spanish version is explained later in the .properties file section.

Twitter Bootstrap

Twitter bootstrap is how we chose to style our application. We wanted a very clean, modern-looking design, but we did not want to have to put too much time into getting the style to look just right (we wanted to be able to focus as much as possible on functionality). Twitter bootstrap is a very easy to use library which does just that. It gave us a very easy to use set of web tools to create a great style for our website without having to do too much of the design by hand. Unless you are planning to redesign the site, you probably do not need to concern yourself too much with how bootstrap works. However, if you would like to learn more, bootstrap’s home page is <http://getbootstrap.com/>.

Servlet.java

The heart of our application is Servlet.java, which handles all of the requests coming into the system. The method processSearch processes the functions of the clinic search form, clinic addition form, clinic editing form and error report form separately. Depending on which page has been submitted, it calls different methods to process the request and works on the database. More details in the database section.

Handling of English and Spanish

While developing our application, we had to have the entire system be in English. Three of us don’t speak spanish at all, and the one that does speak spanish doesn’t speak it very well. So, our solution was that we would work on the application in English and then translate it to Spanish at the end. However, if we’ve already got an English version working, there’s really no reason to throw it out completely and replace it with a Spanish version. So even though we only really intend for this application to be used in Spanish, we might as well provide an English alternative as well.

The way that this is handled is with two .properties files. There is one JSP that handles the search form for the application. When it loads, that JSP looks at the current URL. If that URL contains the word “english”, then it loads english.properties and pulls all text from that file. If the URL does not contain the word “english”, then the default spanish.properties file is used. What this basically means for someone maintaining and possibly improving this project is that changing the wording of anything in the application is handled in these .properties files. Simply search for the phrase that you’re looking to replace in english.properties or spanish.properties and change it to whatever you like. Then, when you redeploy the application, the phrase is updated without actually having to change the JSP at all. This also allows for this application to potentially be put into another language like French. All that would have to be done is a french.properties file would have to be created and the part where index.jsp loads the .properties file would have to be changed to look for “french” as well as “english” in the URL.

web.xml

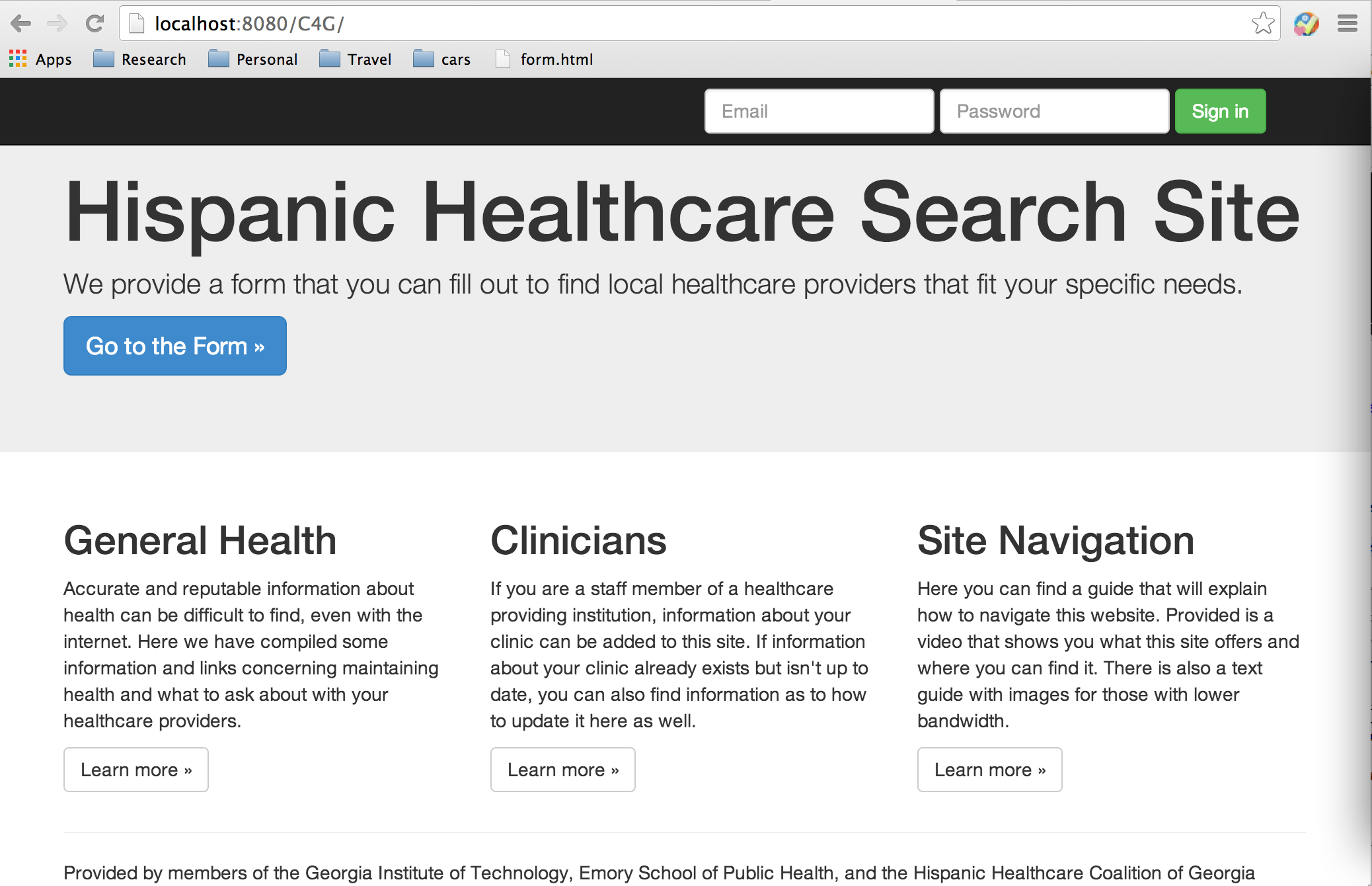
Web.xml handles of the mapping of the system. It maps JSP files to servlets, and then maps those servlets to certain URLs. There shouldn’t be any reason to have to edit this, but if so, it’s as simple as a web.xml can really be. All it has is servlet names and servlet mappings, which are the required parts of a web.xml.

Running database locally

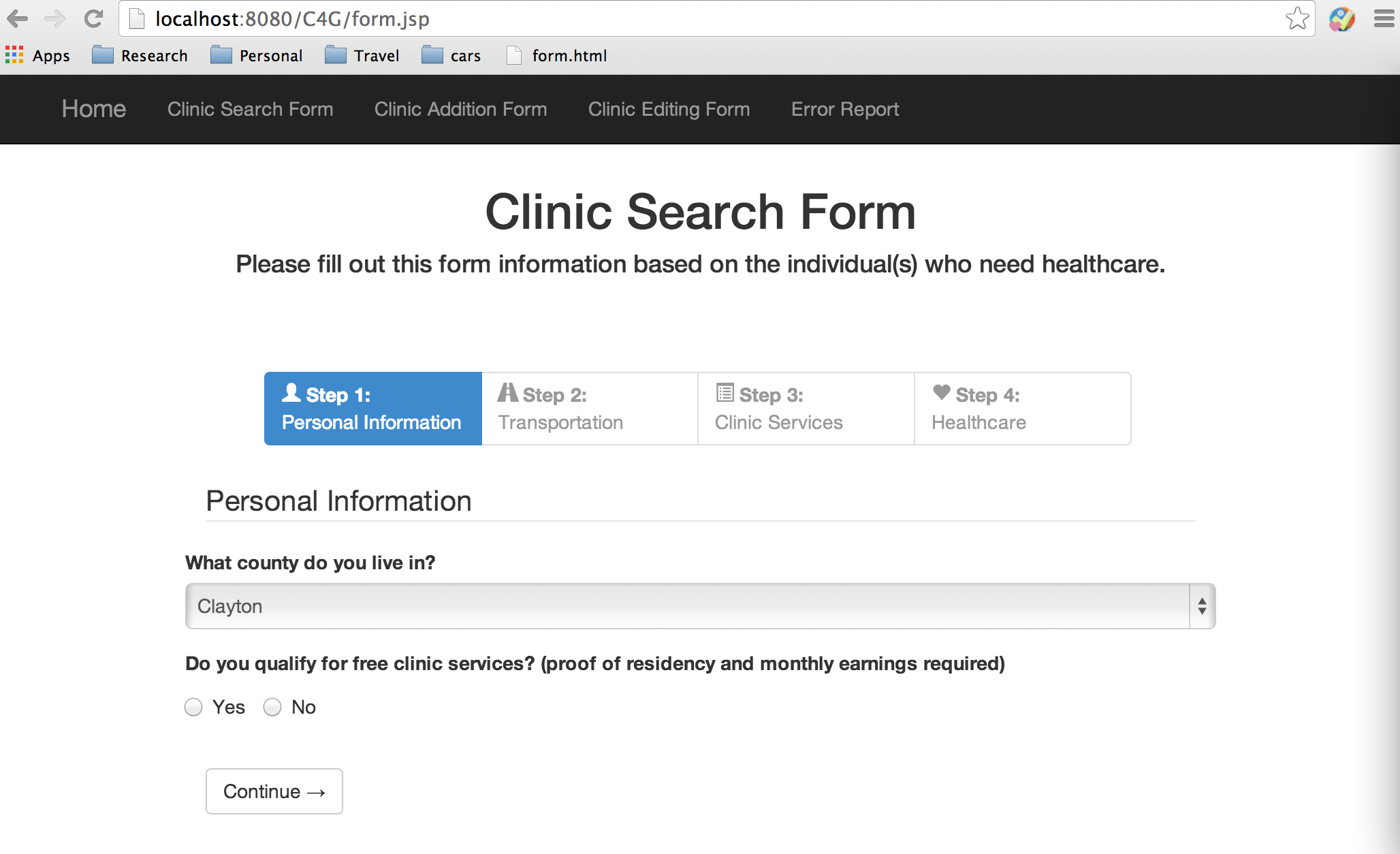
We use tomcat to run the database and put the clinic service data in phpMyAdmin. There are three tables in phpMyAdmin that contains all the collected information about the hospitals. The three tables are P1, P2, P3\_4\_5 containing the information collected from the 1st, 2nd, and 3-4-5 pages in Health Resource Guide Survey. The website contains functions including: searching for the clinic services, editing the current clinic service information, adding new clinic services, and reporting errors. The MySQL statements are in servlet.java. When the user click “Clinic Search Form”, input data, and submit the form, the data in the form will be passed to servlet, and the SQL statements in servlet will run on the database, select the hospitals that can meet the user’s requirements. For editing and adding new clinic services, the process is the same. The SQL statements in servlet can update the current information stored as well as add new tuples containing the hospital information to database. The “Error report” page is for the user or the health workers to report any bugs they meet in using the website. The errors will be added to the Bug\_Report table in database and for later retrieval.

Screenshots

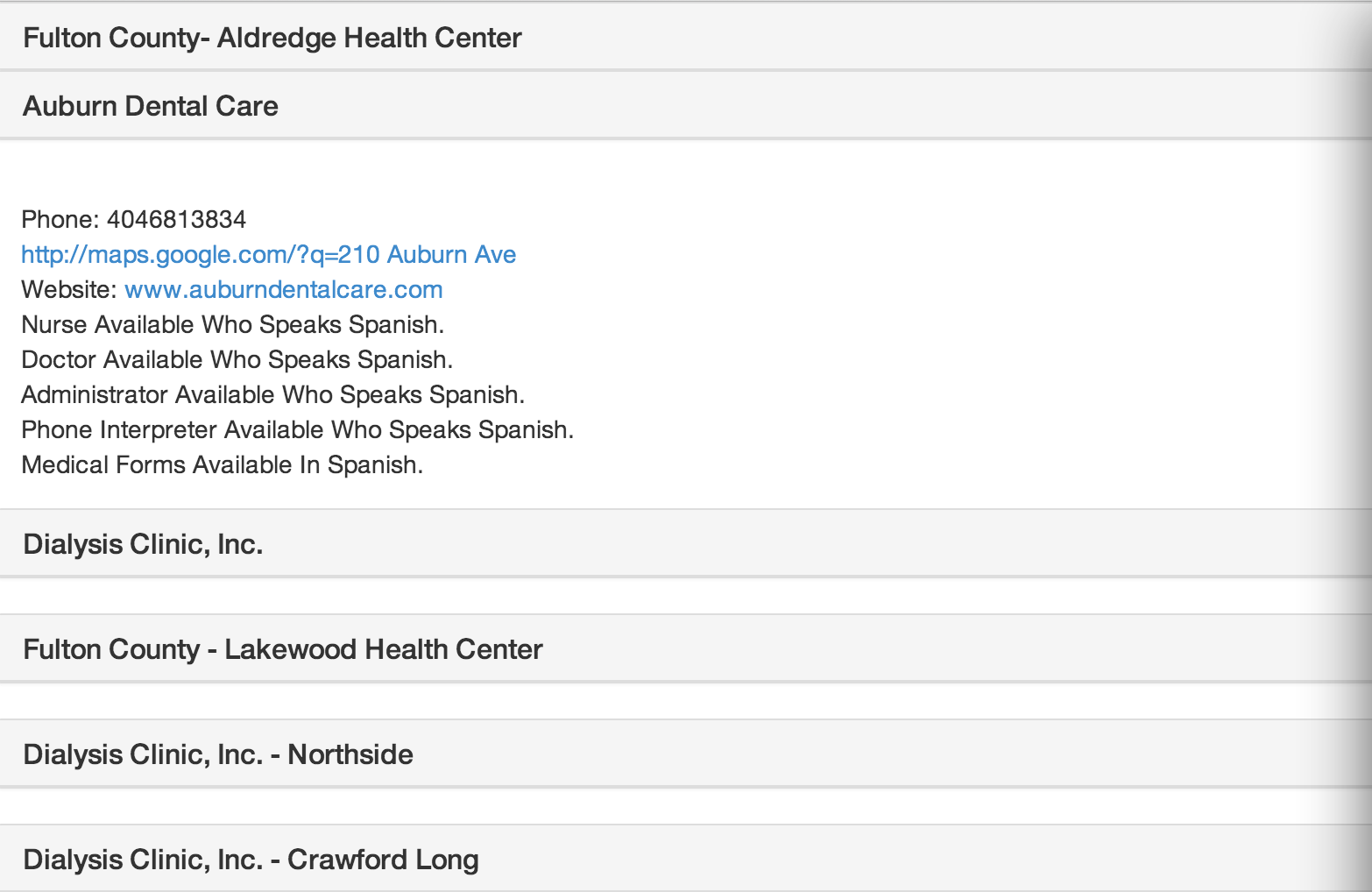
1, When the user open the web app, the welcome page will show up. The user can input email and password to sign in, or just go to the forms directly.



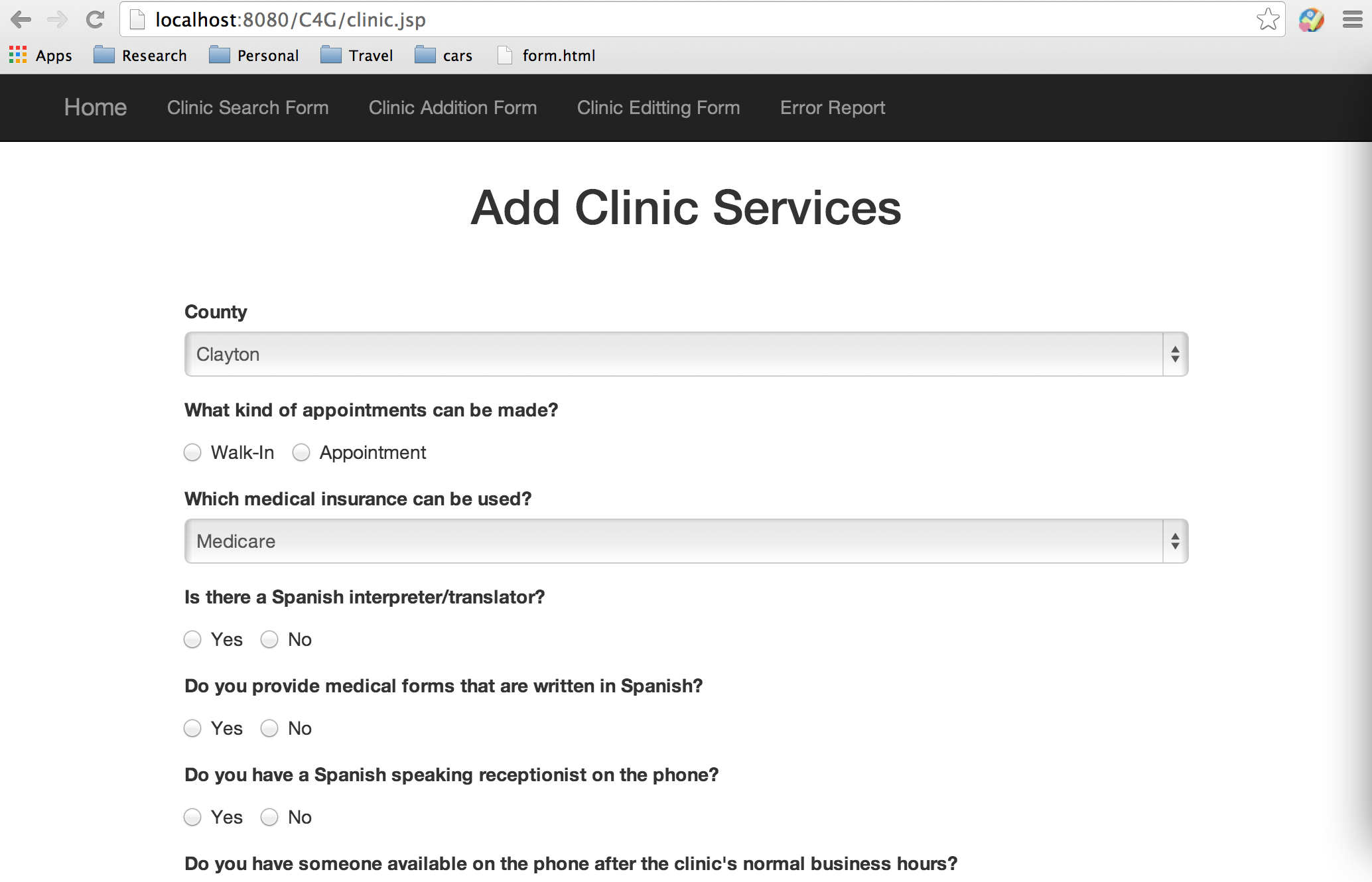
2. After clicking “Go to the Form” or logging in, the user will be able to see the clinic search page. The user can answer the questions and provide basic information about what he/she wants for a clinic service.



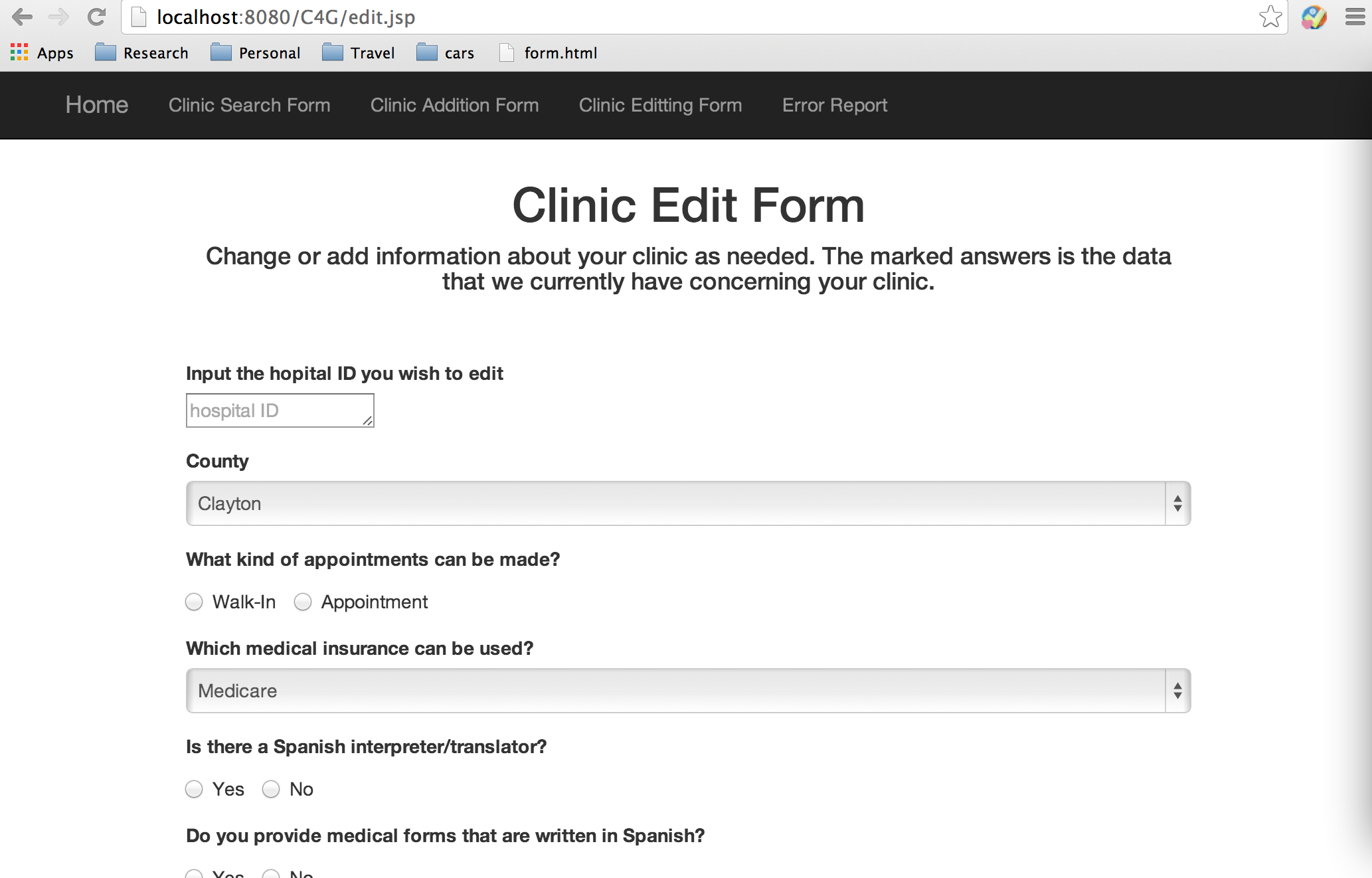
After submitting the clinic search form, the selected hospital information will show up. This is our current design for the response form. The phone number, website, address in google map, and spanish relation information will show up when the user clicks on a certain clinic service.



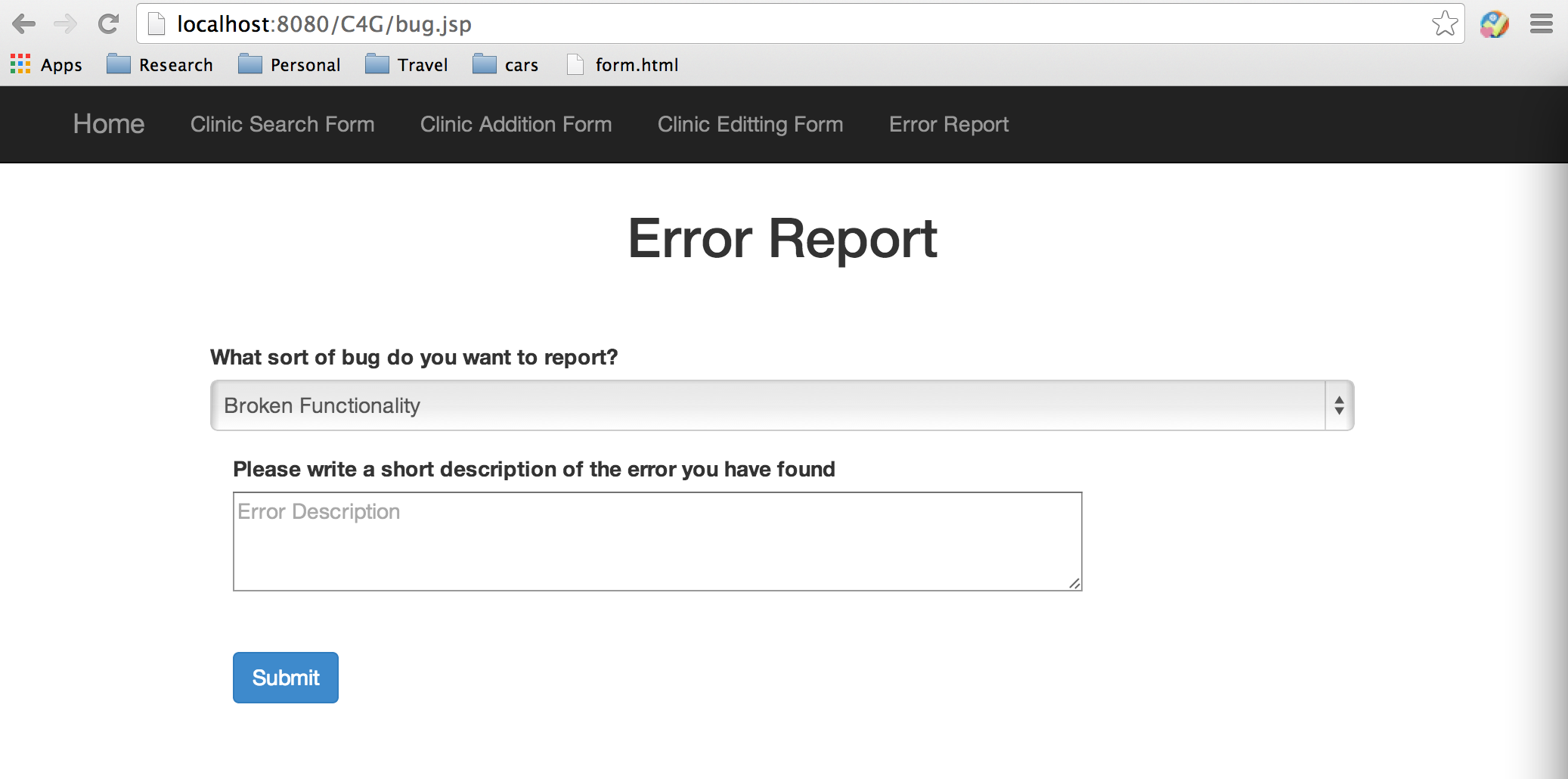
3, The user can also get access to the clinic addition form, clinic editing form and error report form. The clinic addition form is mostly designed for the healthcare workers to add new clinic service information.



4, The clinic edit form is also designed for the healthcare workers to edit the current information stored in the system. This is for the purpose of keeping the information up to date and always provide the correct information to the users.



5, The error report page is for the users, healthcare workers, as well as the system maintainers to report any bug they meet when they are using the system. They can report broken functionality, incorrect information, typos, or any other errors the system have.



User stories

1, As a Hispanic immigrant, I want to locate a medical facility near me that can provide proper health services to me and is accessible via methods of transportation available to me, so that I can seek the proper medical attention I need. I do not speak English and might not be technically savvy, I do not own a private vehicle and will rely on the information provided by the system. When I get into the system and find the clinic search form, I enter my current location, medical situation, and expected clinic service, the system provides me a list of nearby medical facilities that satisfies my criterias.

2, As a faculty member of a medical facility, I want to be edit some information regarding my facility’s services in the system, so that the information regarding the facility I work at is accurate and accessible to potential Hispanic clients. When I click clinic editing form, I input the necessary edits I need to make in the form and submit. The system accurately update the database with the correct information immediately. When I search for my facility’s services, the information are up-to-date.

3, As a maintainer of the system, I want to be able to add, update the database as well as modify the bugs collected from error report form, so that the information provided is always accurate and the system is always running in a good manner. When I found a bug in the Bug\_Report table in database showing that the information provided by a clinic service is incorrect, I check the error and notify the corresponding clinic service, ask for the correct information.