COMP 3100 – Web Programming

Project - Iteration 3

Winter 2022

Project Document

Part 1:

Update from iteration 2: The major functionalities mentioned in iteration 1 and 2, were implemented this time in the front end by using HTML, CSS, JAVASCRIPT and JQUERY. Implementation was done using the REST architecture with the MVC-like framework. In iteration 2 the Model and Controller's were implemented, in this iteration the View codes were added and implemented, so the website can be fully interactive and functional dynamically in both back-end and front-end.

To implement front-end/client-side HTML was used to create the layout of the website. CSS was used for styling the HTML components and finally by using jQuery and JavaScript and with the help of AJAX library the front end interacted with the backend in a dynamic manner, where the requests/changes done in the front end was received/documented in the back end dynamically.

As mentioned in iteration 1 and iteration 2, the major functionalities were covered and implemented. One change was done in the routes/paths of the request calls, initially we planned to use the address of the tenants or landlords to search/update/delete, but we changed the routes/paths of the request calls from address to name. The reason this change was done, is because not all the time we were able to retrieve the information's by using the given address by using the 2 API's MapQuest and PositionStack. Many times, even though everything else was working, there was no respond sent to the client from the server. Therefore, we decided it would be more efficient if we use names instead of addresses. We also divided the Controller into 2 parts, 1 file dedicated for the landlord and 1 dedicated for the tenant, in iteration 2 initially there was only 1 Controller file. Functionalities:

register/create landlord or tenant profiles,

#Update the profiles, landlord or tenant

#For a tenant to search for all the rental address's listed,

#For a landlord to see all the preferred locations listed by a tenant,

#For a tenant to search with a specific preferred address (filter search) using the landlord's name

For a landlord to search with a specific preferred address that was listed by a tenant (filter search) using the tenant's name

For a landlord or tenant to delete their profiles from the database – were mentioned in the first and second iteration and were implemented in this iteration.

There are currently no functionalities that were added unplanned. No major functionalities were removed, but the chat, feedback/report functionalities were mentioned and in iteration 1 and iteration 2, but we could not make it work. We tried to include the chat functionality within the website, but unfortunately could not make it work.

Part 2:

Views:

To complete our MVC framework we created a folder named View within our project directory. The main goal of this folder View was to include the front-end codes that would interact with the back-end. This folder includes 3 HTML files, one for the homepage of the page, called index.html, 1 for the landlords, and 1 for the tenants. We also have a CSS file called style.css to style the HTML elements. Then we have the methods which were implemented to interact the back-end with the HTML elements and the front end. These methods were implemented using the JQuery library and javascript. we have a total of 8 view methods. To get all the tenants and landlords(2 GET methods), to get a specific landlord or a tenant (2 GET methods), to update a landlord or a tenant(2 PUT methods), and to delete a landlord or a tenant(2 DELETE methods). All these methods were implemented separately in their respective files.

In the add.js we implemented a method that, once a landlord/client clicks the add button after filling the necessary information will tell the server using the AJAX library that a user clicked, and with the proper path/routes the server will add these info/profile to the database. We will notification both from in the front end and back end. This will happen dynamically.

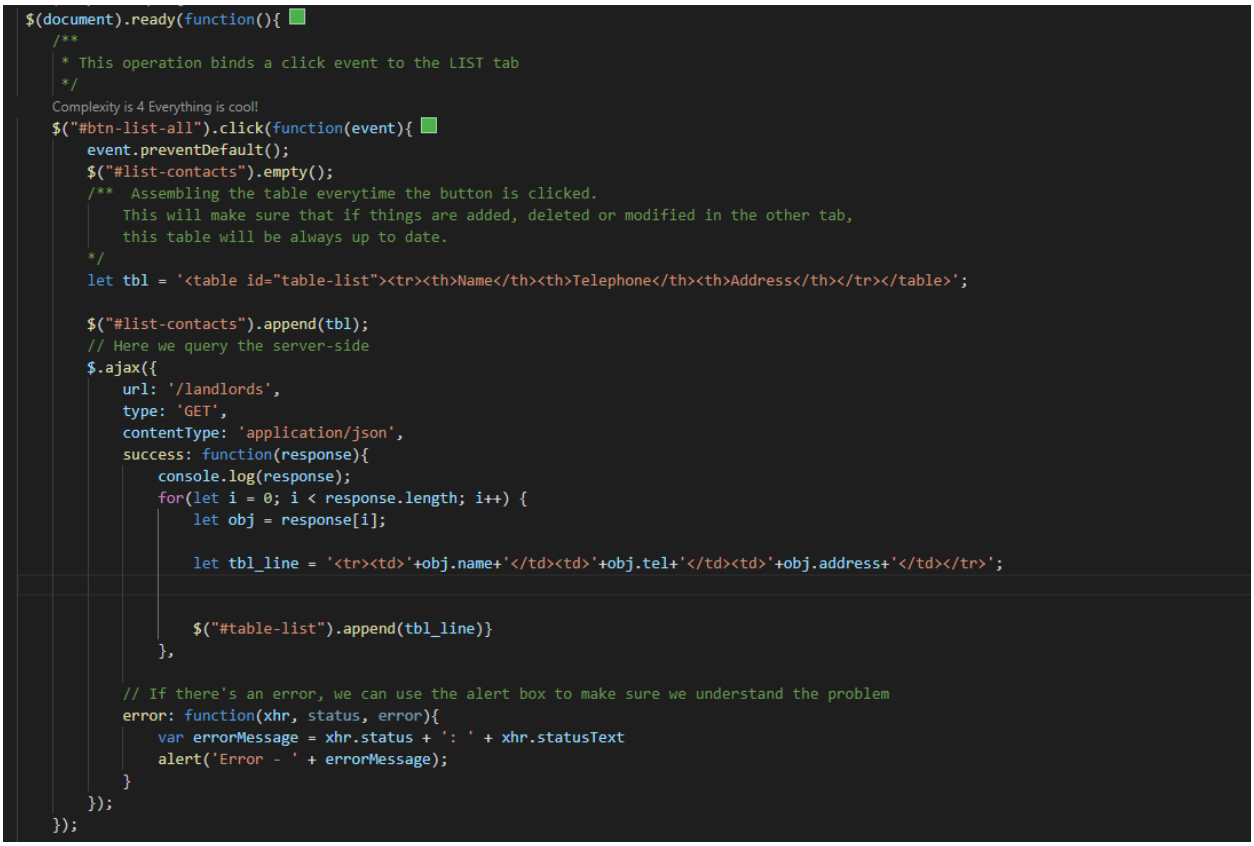
```
Complexity is 3 Everything is cool!
$("#add-contact-btn").click(function(event){
    event.preventDefault();
    let contact = assembleContact();
    $.ajax({
        url: '/landlords',
        type: 'POST',
        contentType: 'application/json',
        data: JSON.stringify(contact),
        success: function(response){
            // We can print in the front-end console to verify
            // what is coming back from the server side
            console.log(JSON.stringify(response));
            $("#add-out").text(response);
        },
        //We can use the alert box to show if there's an error in the server-side
        error: function(xhr, status, error){
            var errorMessage = xhr.status + ': ' + xhr.statusText
            alert('Error - ' + errorMessage);
        }
    });
});
```

The addTenant.js file does the similar thing for tenants.

```
Complexity is 3 Everything is cool!
$("#add-contact-btn").click(function(event){
    event.preventDefault();
    let contact = assembleContact();
    $.ajax({
        url: '/tenants',
        type: 'POST',
        contentType: 'application/json',
        data: JSON.stringify(contact),
        success: function(response){
            // We can print in the front-end console to verify
            // what is coming back from the server side
            console.log(JSON.stringify(response));
            $("#add-out").text(response);
        },
        //We can use the alert box to show if there's an error in the server-side
        error: function(xhr, status, error){
            var errorMessage = xhr.status + ': ' + xhr.statusText
            alert('Error - ' + errorMessage);
        }
    });
});
```

The list.js file has the list_all method that lists all the profiles that were created by the landlords. Like the get method, here as well, with the help of jquery and javascript we bonded the List tab

of the landlord web page to this method. AJAX library sends the GET request to the appropriate path for the server to receive, and the server responds by sending all the listed profiles in the client side.



```
$(document).ready(function(){
    /**
     * This operation binds a click event to the LIST tab
     */
    Complexity is 4 Everything is cool!
    $("#btn-list-all").click(function(event){
        event.preventDefault();
        $("#list-contacts").empty();
        /**  Assembling the table everytime the button is clicked.
            This will make sure that if things are added, deleted or modified in the other tab,
            this table will be always up to date.
        */
        let tbl = '<table id="table-list"><tr><th>Name</th><th>Telephone</th><th>Address</th></tr></table>';

        $("#list-contacts").append(tbl);
        // Here we query the server-side
        $.ajax({
            url: '/landlords',
            type: 'GET',
            contentType: 'application/json',
            success: function(response){
                console.log(response);
                for(let i = 0; i < response.length; i++) {
                    let obj = response[i];

                    let tbl_line = '<tr><td>'+obj.name+'</td><td>'+obj.tel+'</td><td>'+obj.address+'</td></tr>';


                    $("#table-list").append(tbl_line)}
                },

            // If there's an error, we can use the alert box to make sure we understand the problem
            error: function(xhr, status, error){
                var errorMessage = xhr.status + ': ' + xhr.statusText
                alert('Error - ' + errorMessage);
            }
        });
    });
```

listsTenants.js does the similar job for tenants.

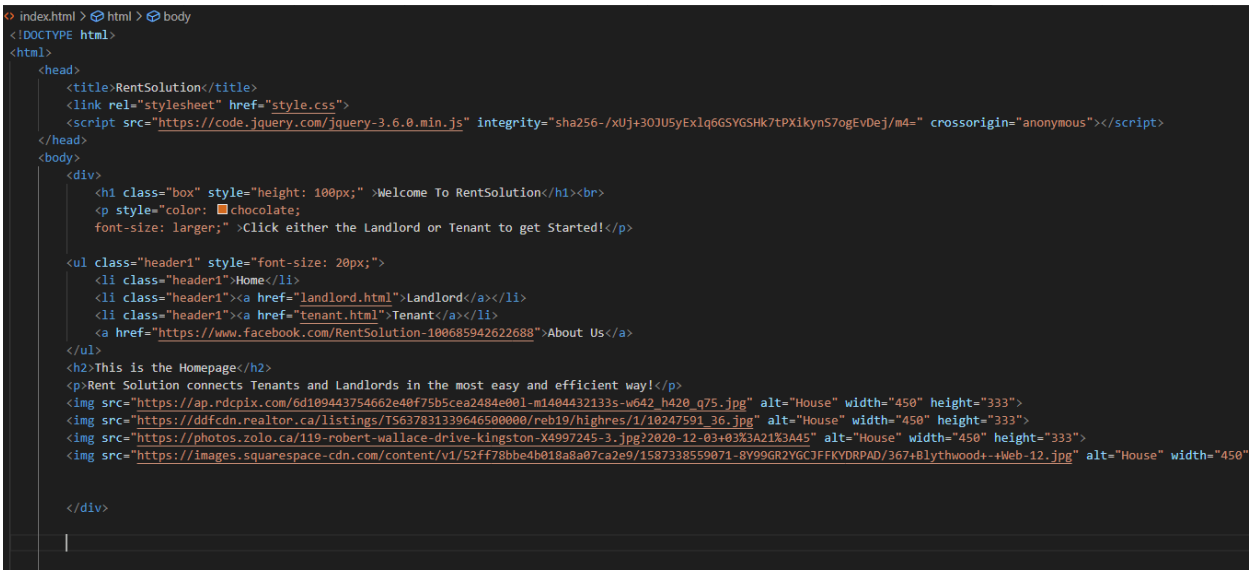
In the find.js we followed the class lectures to implement all 3 functions, get a specific landlord/tenant, update and delete in the same file. Following the same REST architecture, we bonded these methods with the respective buttons, with jquery and javascript. With the AJAX library and with the correct paths/routes the client will send requests and the server will respond accordingly and send the response, again this will happen at the same time, dynamically.

```
Complexity is 3 Everything is cool!
$("#btn-update-contact").click(function(event){
    event.preventDefault();
    let contact_name = $("#find-name-search").val();
    let contact = assembleContact();
    $.ajax({
        url: '/landlords/'+contact_name,
        type: 'PUT',
        data: JSON.stringify(contact),
        contentType: 'application/json',
        success: function(response){
            console.log(response);
            $("#update-delete-out").text(response.msg);
        },
        error: function(xhr, status, error){
            var errorMessage = xhr.status + ': ' + xhr.statusText
            alert('Error - ' + errorMessage);
        }
    });
});
/**
 * This function will bind an event to the delete button
 */
Complexity is 3 Everything is cool!
$("#btn-delete-contact").click(function(event){
    event.preventDefault();
    let contact_name = $("#find-name-search").val();
    $.ajax({
        url: '/landlords/'+contact_name,
        type: 'DELETE',
        contentType: 'application/json',
        success: function(response){
            // console.log(JSON.stringify(response));
            console.log(response);
            $("#update-delete-out").text(response.msg);
            // We clear the fields after the data is deleted
            fillFindContainer(null);
        },
```

Without these functions there would not be any connections between front end and back end, these functions are what makes the requests through the appropriate libraries to the back end, and the back end responds accordingly. Therefore, for the website to work, these functions are vital, otherwise the website would be static and would not work.

Part 3:

Apart from using all the above-mentioned functionalities our project also uses HTML and CSS elements, which are the core building blocks of the front-end side. The major HTML elements used are, header, div, buttons, forms, paragraphs, unordered lists, listed elements, images, scripts, input/output, labels etc. These elements were styled as per needed, with the help of CSS classes and properties. We created CSS classes and bonded them with the HTML elements, with the HTML elements properties. These were the necessary elements that we needed to assemble our client side. CSS elements included flex box which was used to display the HTML components in an order, the styling elements also include background color, border, padding, margin, etc.



```
index.html > html > body
<!DOCTYPE html>
<html>
    <head>
        <title>RentSolution</title>
        <link rel="stylesheet" href="style.css">
        <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
    </head>
    <body>
        <div>
            <h1 class="box" style="height: 100px;" >Welcome To RentSolution</h1><br>
            <p style="color: chocolate;
            font-size: larger;" >Click either the Landlord or Tenant to get Started!</p>

        <ul class="header1" style="font-size: 20px;">
            <li class="header1">Home</li>
            <li class="header1"><a href="landlord.html">Landlord</a></li>
            <li class="header1"><a href="tenant.html">Tenant</a></li>
            <a href="https://www.facebook.com/RentSolution-100685942622688">About Us</a>
        </ul>
        <h2>This is the Homepage</h2>
        <p>Rent Solution connects Tenants and Landlords in the most easy and efficient way!</p>
        <img src="https://ap.rdcpix.com/6d109443754662e40f75b5cea2484e00l-m1404432133s-w642_h420_q75.jpg" alt="House" width="450" height="333">
        <img src="https://ddfcdn.realtor.ca/listings/TS637831339646500000/reb19/highres/1/10247591_36.jpg" alt="House" width="450" height="333">
        <img src="https://photos.zolo.ca/119-robert-wallace-drive-kingston-X4997245-3.jpg?2020-12-03+03%3A21%3A45" alt="House" width="450" height="333">
        <img src="https://images.squarespace-cdn.com/content/v1/52ff78bbe4b018a8a07ca2e9/1587338559071-8Y99GR2YGCJFFKYDRPAD/367+Blythwood+-+Web-12.jpg" alt="House" width="450"


        </div>
```

```
1
2   .header1{
3       display: inline-block;
4   }
5
6   .box{
7       background-color: coral;
8       border: 20px;
9       padding:  50px;
10      margin: 0px;
11  }
12
13  .box2{
14      background-color: cornflowerblue;
15      border: 20px;
16      padding:  50px;
17      margin: 0px;
18  }
19
20  .box3{
21      background-color: darkolivegreen;
22      border: 20px;
23      padding:  50px;
24      margin: 0px;
25  }
26
27
28  input {
29      display: flex;
30
31      padding: 12px 20px;
32      margin: 8px 8px;
33  }
34
35  .tab {
36      display: flex;
37      overflow: hidden;
38      border: 1px solid #ccc;
39      background-color: skyblue;
```

Part 4:

As mentioned in the above sections, that the interaction between the front-end and back-end happens in real time, or dynamically. The server will inform right away once a client/user clicked any buttons to make any changes in the front-end.