**Srikanth Gude** – sgcnm@umsystem.edu

Git Hub Link -
https://github.com/gudesrikanth/webcourse/tree/main/Webpart/ICP6

**Gopi Nelluri** – gng75@umsystem.edu

Git Hub Link -
https://github.com/gopinelluri9/demo_remote_repository/tree/main/WebPart/ICP6

# ICP – 6(REST APIs with Angular)

## Introduction:

A **REST**ful API is a type of application program interface (API) that uses HTTP requests to retrieve and use data. This data can be used with the GET, PUT, POST, and DELETE data types, which correspond to accessing, modifying, creating, and deleting resources, respectively.

To download or upload files and use various rear features, most front-end applications must interact with a server using the HTTP protocol. The **HttpClient** service class in @angular/common/http offers an HTTP client **API** for **Angular** applications. This service has mainly contained some features such as the users can send typed answer objects as a request, error handling has been simplified and Interception of requests and responses.

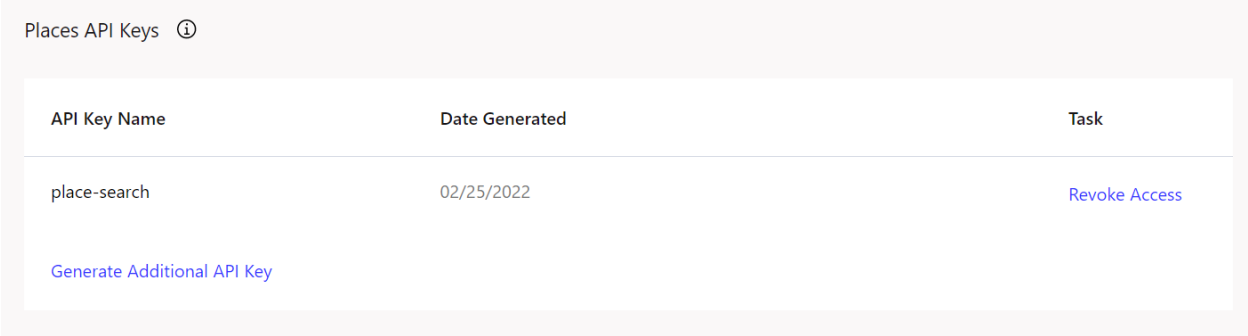
## Webpage Description:

Created a simple application with two primary functions: the user can search for nearby restaurants by clicking on the link, which takes them to Google Maps and a list of recipes; and the user can navigate to recipe preparation blogs by clicking on the link. I'm using the EDAMAM API to keep track of recipes and the Foursquare API to keep track of restaurants.

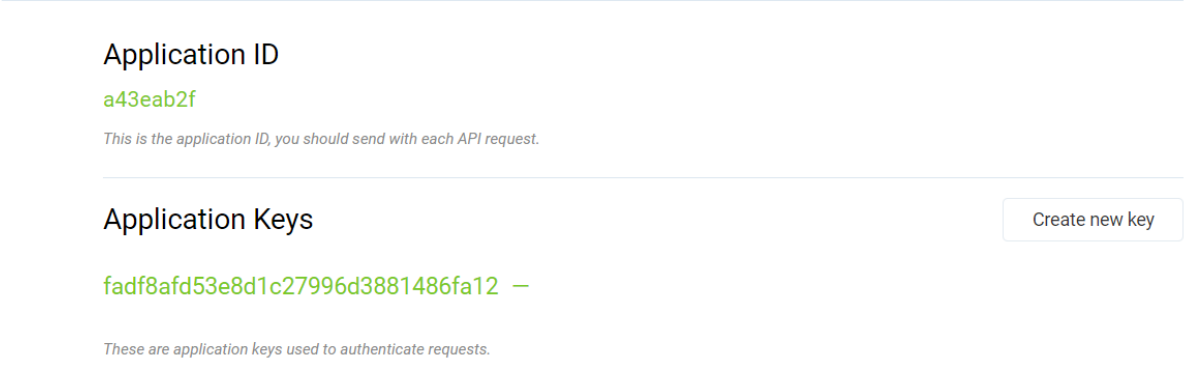
We have followed below steps to create api authorization information:

- By using this URL https://foursquare.com/ we created an account in Four Square.

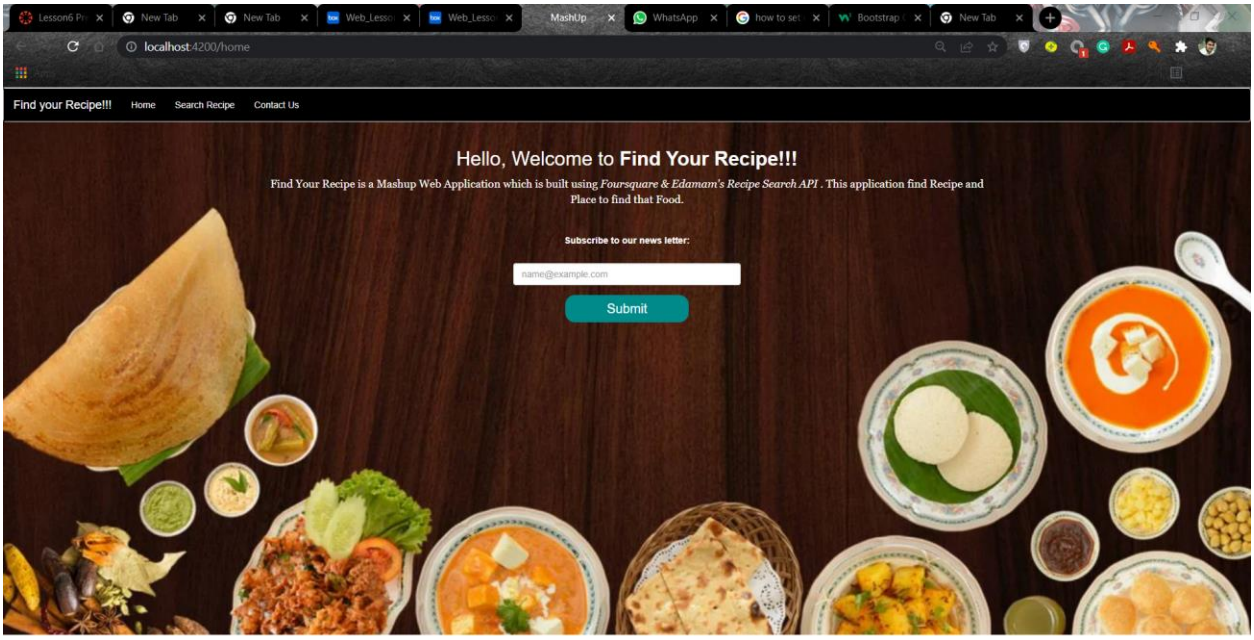
- Go to the URL https://developer.foursquare.com/docs/api/venues/search and specify the query parameters as necessary.
- For the API request that we used api key to get user requested place search.




- By using this URL https://www.edamam.com/ we created an account in EDAMAM.
- Go to the URL https://developer.edamam.com/admin/applications and specify the query parameters as necessary.
- For the API request that you make, use the application id and key.



The Website looked like below:




# Header:

## HTML Code:

```
ME.md | header.component.ts | header.component.html | home.component.spec.ts | home.component.ts | search
<nav class="navbar navbar-default background">
  <div class="container-fluid">
    <div class="navbar-header">
      <a routerLink="/" class="navbar-brand background">Find your Recipe!!!</a>
    </div>

    <div class="collapse navbar-collapse tx">
      <ul class="nav navbar-nav">
        <li><a routerLink="/home">Home</a></li>
        <li><a routerLink="/search-recipe">Search Recipe</a></li>
        <li routerLinkActive="active"><a routerLink="/shopping-list">Contact Us</a></li>
      </ul>

    </div>
  </div>
</nav>
```

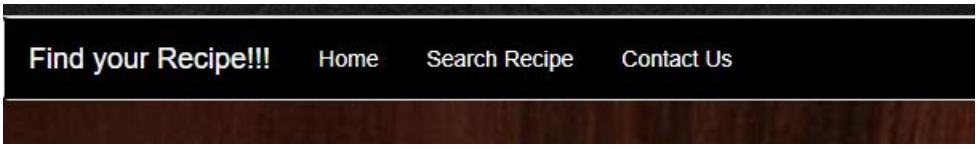
In above code snippet we have used ‘navbar’ bootstrap tag for responsive navigation of web application when the user is searching for the recipe. Moreover, we also used ‘router link’ tag for declaratively travelling to a different path such as Home, Search Recipe, Contact Us..

**TS Code:**

```
import { Component } from '@angular/core';


@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styles: [
    '.background {background:#000000; color: white}',
    'li a { color: white}',
    'ul.nav a:hover { color: #fffccc  }'
  ]
})
export class HeaderComponent {
  constructor() {}

  }
```

In above code snippet HeaderComponent class is created and used as a component and css styles are used to apply color for background.

**Output:**




In above screenshot it is clearly showing that, ‘navbar’ is used to maintain the header and responsive webpage based upon the screen size

## HOME:

### HTML Code:

```
<div class="parent-container">
  <div class="child-container">
    <h2>Hello, Welcome to <b>Find Your Recipe!!!
    </b></h2>
    <span class="desc">Find Your Recipe is a Mashup Web Application which is built using <i> Foursc
    <form>
      <div class="form-group">
        <label>Subscribe to our news letter:</label><br><br>
        <div class="form-group col-lg-4 col-lg-offset-4">
          <input type="text" class="form-control" placeholder="name@example.com"/>
        </div>
        <br>
        <button class="btn btn primary col-lg-2 col-lg-offset-5">Submit</button>
      </div>
    </form>
  </div>
</div>
```

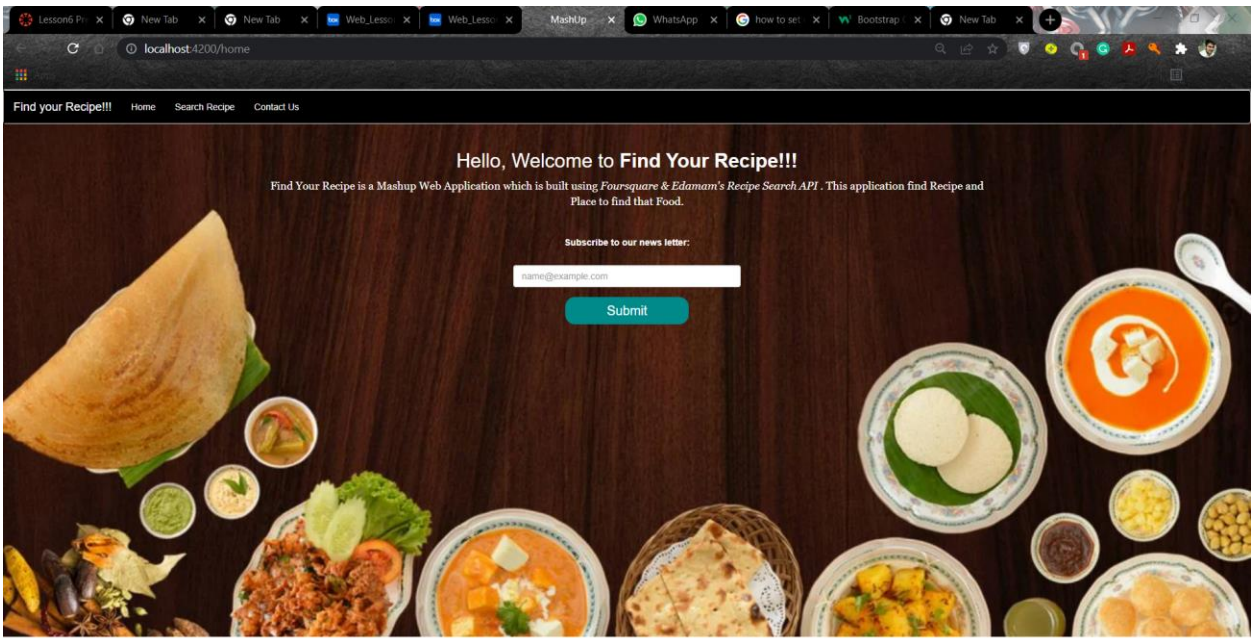
In above code snippet we used ‘form-group’ class to maintain form structure and optimum space between the optional help text, controls, labels. ‘form-control’ class will handle the user interface connection with user and server.

### TS Code:

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

In above code snippet HomeComponent class is created and implements OnInit component with all data-bound properties and sets directive input properties.

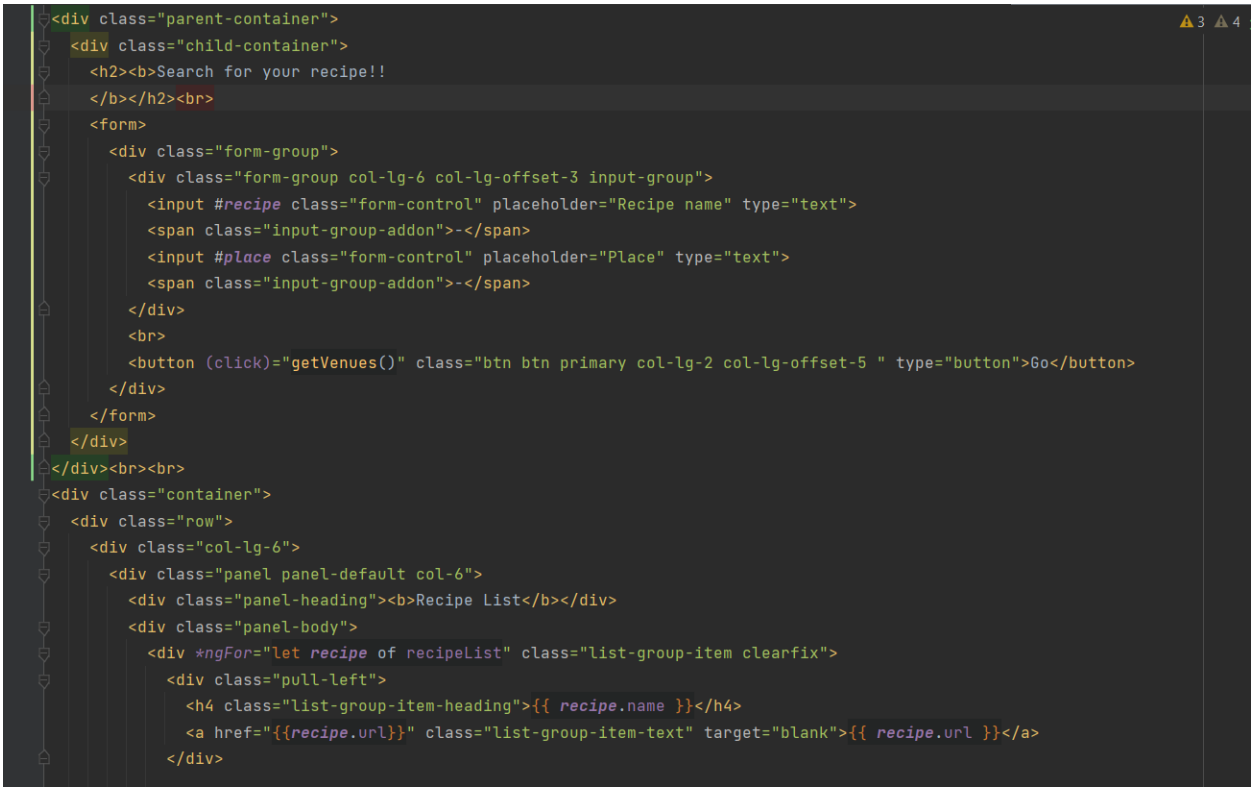
**Output:**




In above screenshot output is displayed for Home menu and it displays all text and input text area

## Search Recipe:

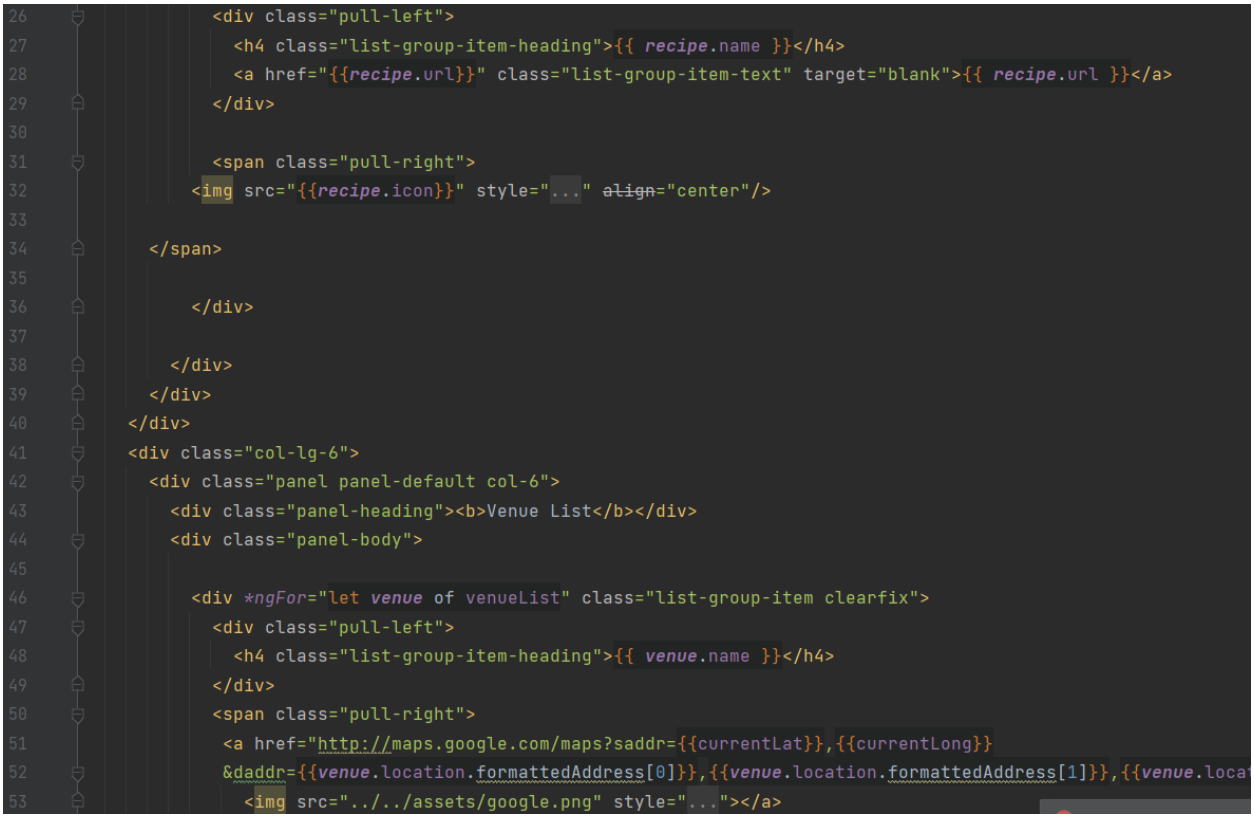
### HTML Code:



```
<div class="parent-container">
  <div class="child-container">
    <h2><b>Search for your recipe!!
    </b></h2><br>
    <form>
      <div class="form-group">
        <div class="form-group col-lg-6 col-lg-offset-3 input-group">
          <input #recipe class="form-control" placeholder="Recipe name" type="text">
          <span class="input-group-addon">-</span>
          <input #place class="form-control" placeholder="Place" type="text">
          <span class="input-group-addon">-</span>
        </div>
        <br>
        <button (click)="getVenues()" class="btn btn primary col-lg-2 col-lg-offset-5 " type="button">Go</button>
      </div>
    </form>
  </div>
</div><br><br>
<div class="container">
  <div class="row">
    <div class="col-lg-6">
      <div class="panel panel-default col-6">
        <div class="panel-heading"><b>Recipe List</b></div>
        <div class="panel-body">
          <div *ngFor="let recipe of recipeList" class="list-group-item clearfix">
            <div class="pull-left">
              <h4 class="list-group-item-heading">{{ recipe.name }}</h4>
              <a href="{{recipe.url}}" class="list-group-item-text" target="blank">{{ recipe.url }}</a>
            </div>
```

In above code snippet we used different bootstrap classes and components to create webpage as responsive and organized view, and input tag is used to enter data from user and button tag will create clickable option to get user requested information.

**HTML Code:**



```
          <div class="pull-left">
            <h4 class="list-group-item-heading">{{ recipe.name }}</h4>
            <a href="{{recipe.url}}" class="list-group-item-text" target="blank">{{ recipe.url }}</a>
          </div>

          <span class="pull-right">
        <img src="{{recipe.icon}}" style="..." align="center"/>

    </span>

        </div>

      </div>
    </div>
  </div>
  <div class="col-lg-6">
    <div class="panel panel-default col-6">
      <div class="panel-heading"><b>Venue List</b></div>
      <div class="panel-body">

        <div *ngFor="let venue of venueList" class="list-group-item clearfix">
          <div class="pull-left">
            <h4 class="list-group-item-heading">{{ venue.name }}</h4>
          </div>
          <span class="pull-right">
           <a href="http://maps.google.com/maps?saddr={{currentLat}},{{currentLong}}
           &daddr={{venue.location.formattedAddress[0]}},{{venue.location.formattedAddress[1]}},{{venue.loca
            <img src="../../assets/google.png" style="..."></a>
```

In above code snippet *ngFor is used to iterate data collection with different template at each time, and all other bootstrap classes and components will rearrange the data in list or particular view.

**TS Code:**

```
// To get Recipe Data from Edamam API
getRecipeData() {
  this.recipeList = [];
  this.http.get( url: `https://api.edamam.com/search?q=${this.recipeValue}&from=0&to=5&`
    +`app_id=a43eab2f&app_key=fadf8afd53e8d1c27996d3881486fa12`).
  subscribe( next: (data :Object ) =>
  {
    console.log("Recipe Data",data);
    const recipeData = data['hits'];
    recipeData.forEach(recipeValue => {
      this.recipeList.push({
        name : recipeValue['recipe'].label,
        url : recipeValue['recipe'].url,
        icon: recipeValue['recipe'].image
      })
    });
    console.log("Receipe Array",this.recipeList)
  })
  return this.recipeList;
}
```

In above code snippet we created getRecipeData() method to get recipe related information from website. And by using get method we sent an url with app id and key, using api connection get method will connect with server and will post the information requested by user. After receiving data from server it will store in array and by using forEach we will iterate recipe name, url, and image.

**TS Code:**

```
//To get Place Data from FourSquare API
getPlaceData(){
  this.venueList = [];
  const head_HTTP = new HttpHeaders( headers: {
    Accept: 'application/json',
    Authorization: 'fsq3wQ8U88RYgxh4D/C6WDSqvZd4mwz51ho0L6CnyhdMpf8=',
  });
  this.http.get( url: `https://api.foursquare.com/v3/places/search?query=`
    +this.recipeValue + `&near=` + this.placeValue + '&v=20220221', options: {headers: head_HTTP}).
  subscribe( next: (data :Object ) => {
    console.log("Restaurent Location data",data);
    const placeData = data['results'];
    placeData.forEach(places => {
      this.venueList.push({
        name: places.name,
        location: {
          formattedAddress: [places.location.formatted_address,
            places.location.locality, places.location.country]
        }
      });
    });
    console.log("placedata", this.venueList);
    return this.venueList;
  })
}
```

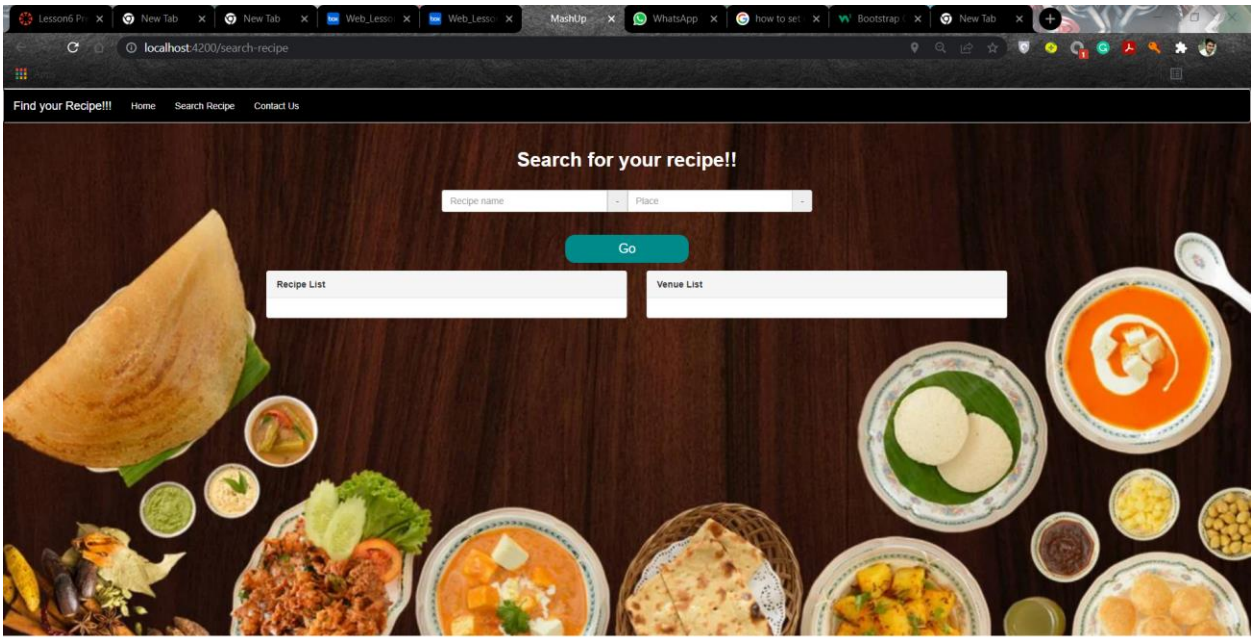
In above screenshot getPlaceData() method is created to get place related information. We created a constant HTTP Header and requested information through get method with foursquare website url with client id and secret id, after sending request api will take the information from application json file and it will check for authorization and accepts it with provided data, After succersfull authorization it will send information back with Post method to respected variables and using forEach it will iterate the data requested by user and print to the console with list view.

**TS Code:**

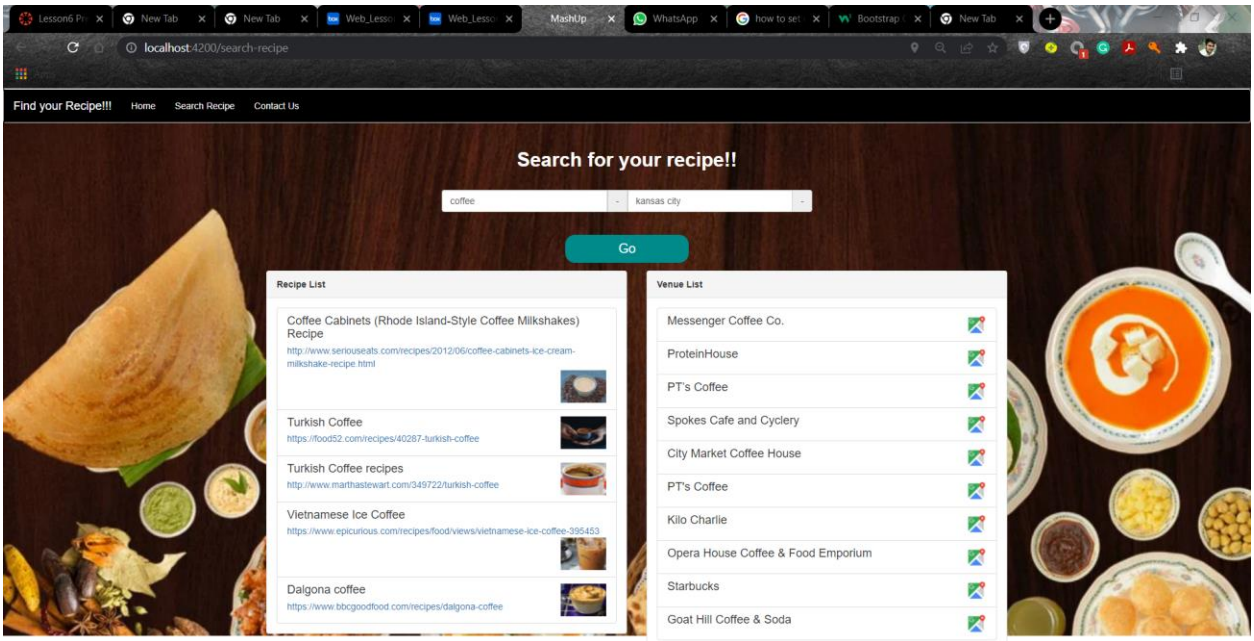
```
//To get venues method
getVenues() {
  this.recipeValue = this.recipes.nativeElement.value;
  this.placeValue = this.places.nativeElement.value;
  if (this.recipeValue !== null) {
    this.getRecipeData();
  }
  if (this.placeValue != null && this.placeValue !== '' && this.recipeValue != null &&
    this.recipeValue !== '') {
    this.getPlaceData()
  }
}
}
```

In above screenshot we created getVenues() method and called getRecipeData() and getPlaceData(), both methods will validate if conditions and return the values as requested by user.

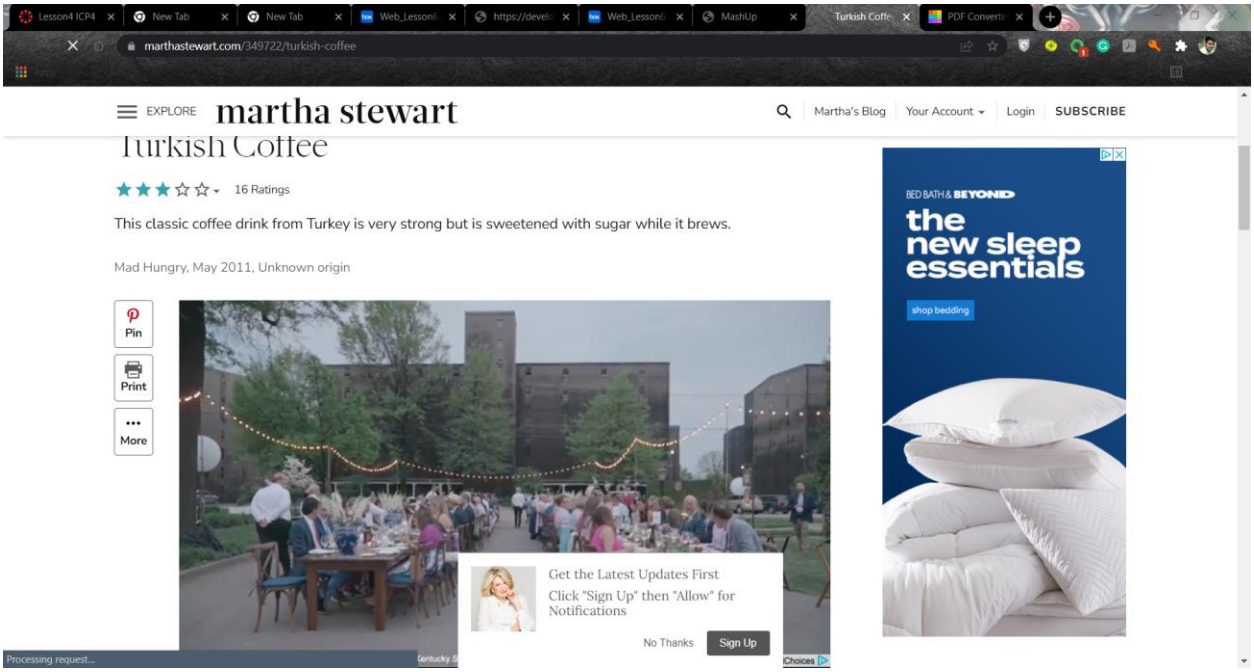
**Output:**



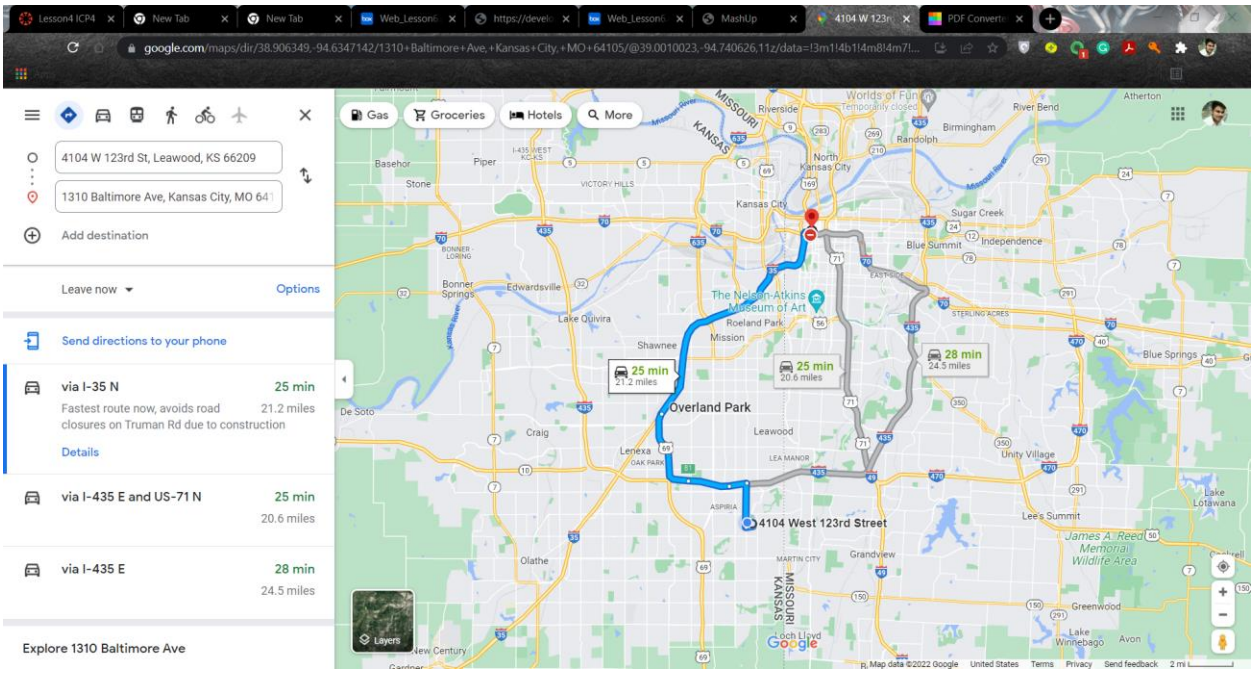

In above screenshot user can give input with recipe name and place to get details of requested data.




In above screenshot we can find the data as requested by user with recipe name and place, after displaying the list of data we can click on any value in list to redirect to respected webpage.




In above screenshot it shows when we click on any link in search result it is redirected to respected website.




In above screenshot it shows when we click on map icon it is redirected to respected place location in googlemaps.

## Conclusion:

In this ICP we learned about how to develop a web page by using REST API and Angular and at last we have designed and developed web page successfully. We didn’t face any major issues while doing this ICP.