CSCU9YW

Web Services

Report
Student Number-2626628

Index

1. Overview

The aim of this project is to create a web service API that acts as a contacts database.

The problem is to add the contacts in the database and provide methods to retrieve, add, edit and delete contacts. We should also be able to find the contacts by city.

A REST solution is provided using spring-boot, A controller class that provides CRUD operations. in order to build an API that helps to save, update and retrieve contacts in/from database. Once the contacts are saved another API is built, to fetch the saved contacts and then added a search feature to fetch particular contact based on the user's city. Then, API's were built to update and delete contacts in the database. Once the all the API's are built a simple HTML page is designed. Ajax and jQuery are used to work with already built API's. @controller, @RequestMapping, @Autowired, @GetMapping, @PostMapping, @DeleteMapping and @PutMapping annotations are used in the controller to build all the API's. JPA CRUD repository is added to perform the database operations and H2 in memory database is used to store the data. I have created 2 entity classes Address and Contact with a many-to-one mapping from contact to Address. @Entity, @Id, @GeneratedValue, @ManyT0One annotations from Java persistence library are used.

Thus, A basic UI providing few services to the client is created using the web services REST based on spring-boot technology.

2. Assumptions

- The details of a particular contact include only name, contact number, street, city/town, postal code submit button and search field.
- All fields are mandatory
- Contact number allows only numbers and unique
- On load of the index page should display all existing contacts in a tabular form
- Users can search contact by city name and this should not be case sensitive.
- "Edit" and "Delete" buttons should be provided to edit/delete the existing contacts
- J2EE and spring technology stack can be used.
- Search by city functionality should provide the full city name.

3. Solution and Testing

Figure 1 is an overview of the basic UI. This UI allows users to add contacts to the Database. Initially the contact table is empty as there are no contacts in the DB.

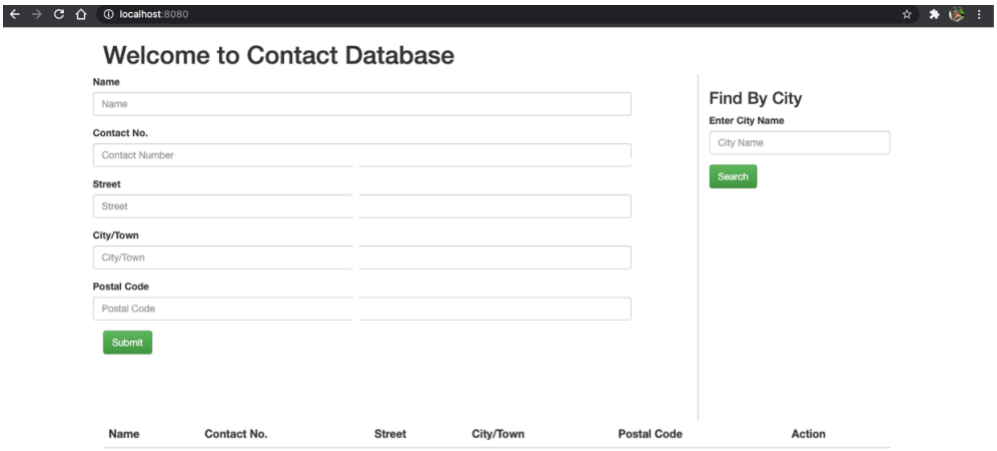

Figure 1

Figure 2 : Name field is provided for the user to enter his/her name.

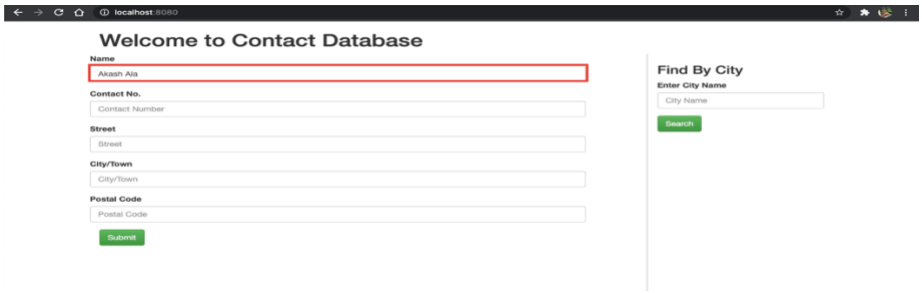

Figure 2

Figure 3: Contact field is provided for the user to enter his/her Contact.

Figure 3

Figure 4: Street field is provided for the user to enter his/her Street.

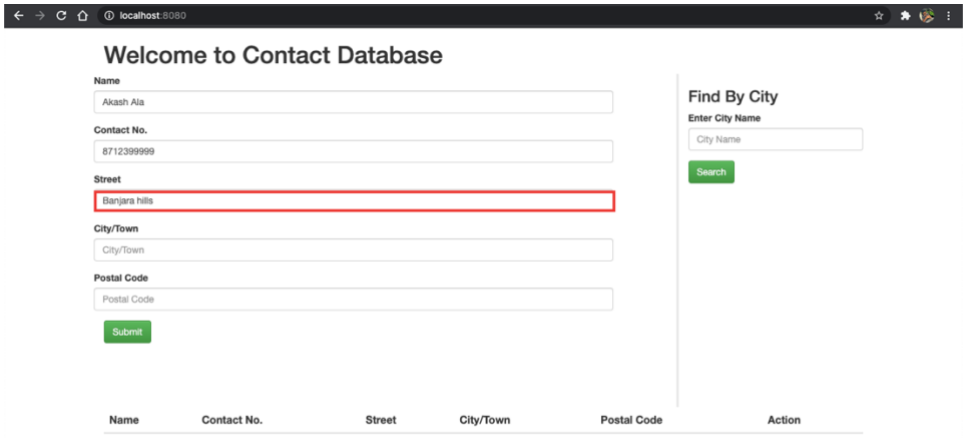

Figure 4

Figure 5: City/Town field is provided for the user to enter his/her City/Town.

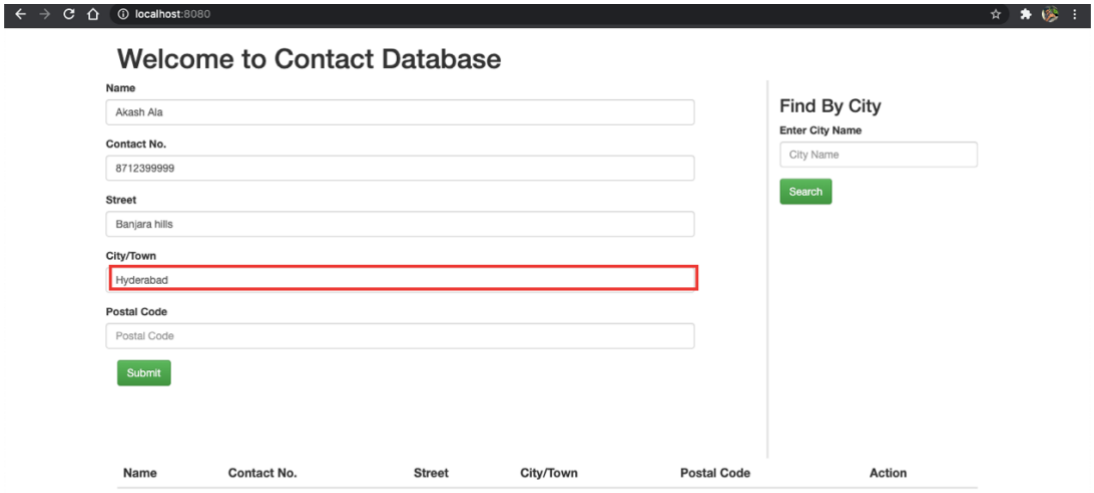

Figure 5

Figure 6: Postal Code field is provided for the user to enter his/her Postal Code.

Figure 6

Figure 7: Once the details are filled and submitted it shows the dialogue box displaying "Contact saved successfully"

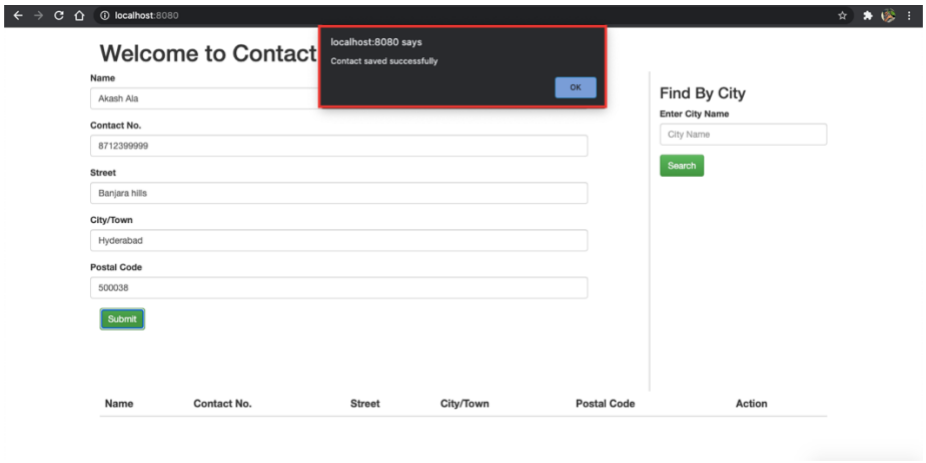

Figure 7

Figure 8: After clicking on ok, the saved contacts are displayed in the table below highlighted in Red.

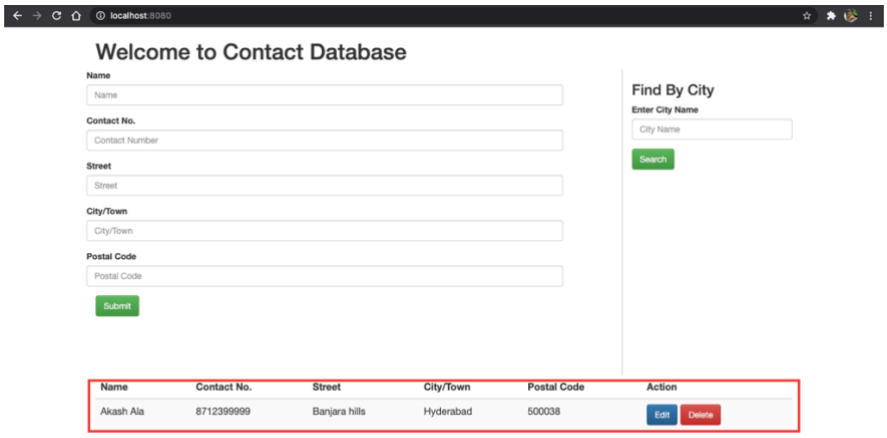

Figure 8

Figure 9: Edit option is provided to update the contact in the database. For example, here the contact "Akash Ala's" street is updated to "Jubilee hills". Figure 9, 10 and 11 demonstrates the update feature.

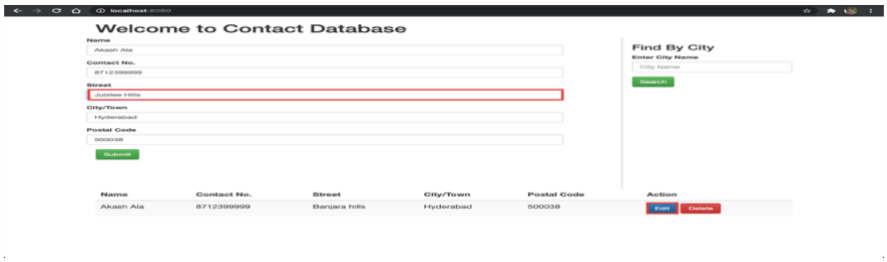

Figure 9

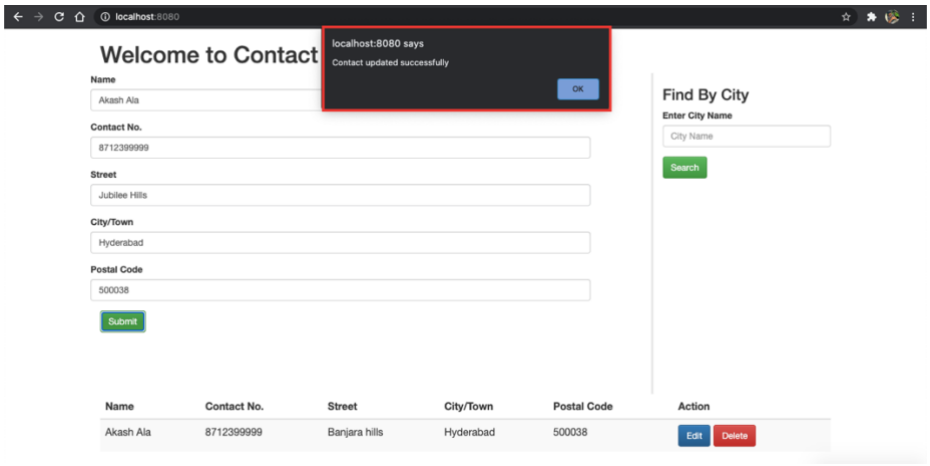

Figure 10

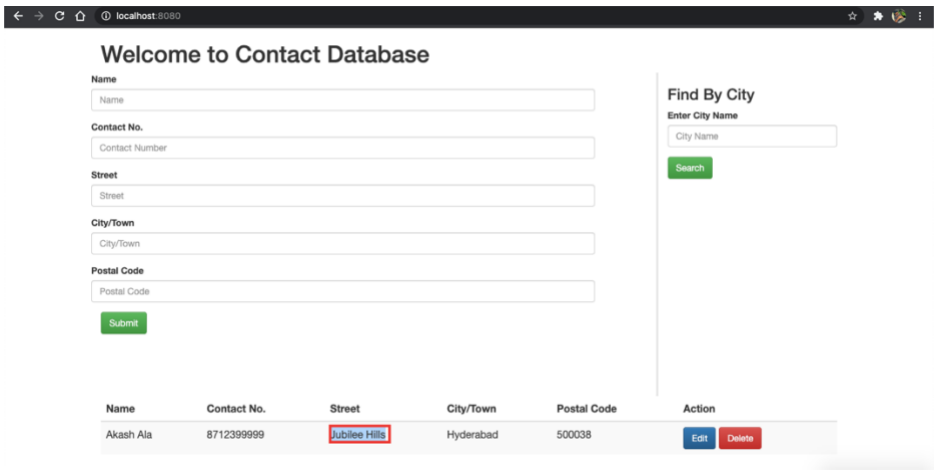

Figure 11

Figure 12: Delete field is provided for the user to delete the contact from the database.

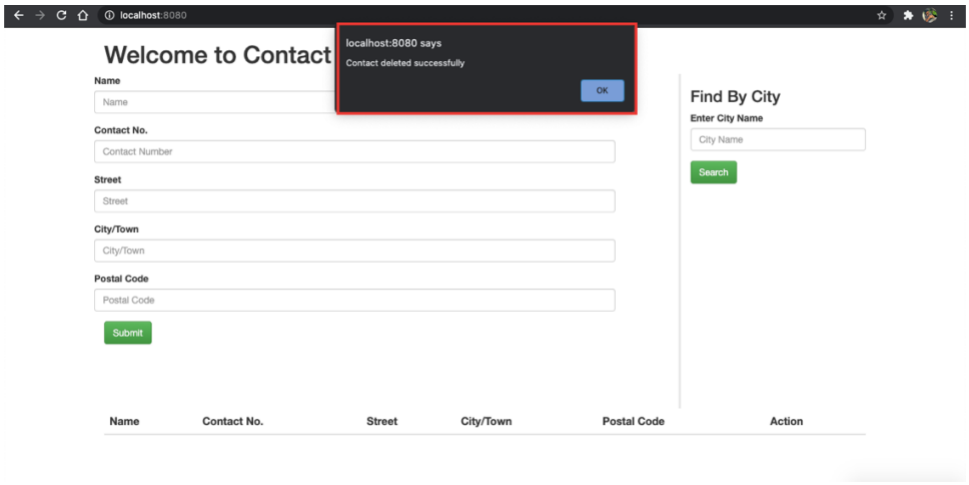

Figure 12

Figure 13: If the user tries to add the existing contact number, a dialogue box pop up saying "Contact already exists".

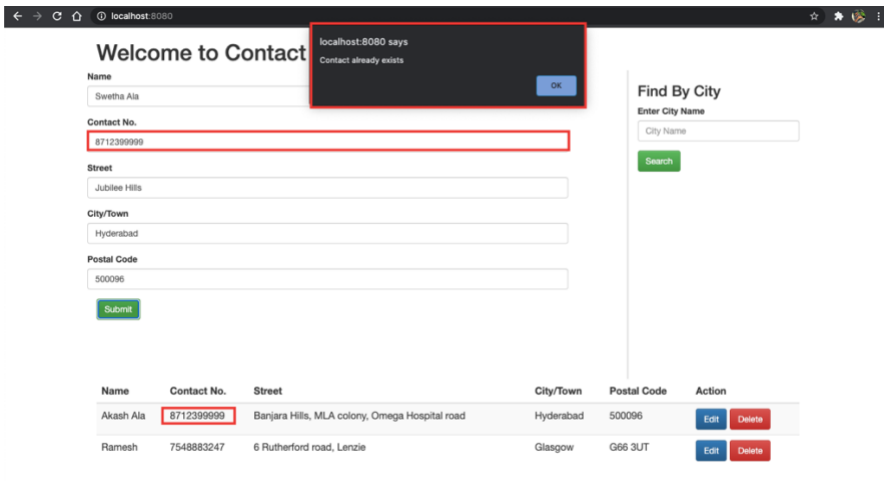

Figure 13

Figure 14: shows the contacts saved in the database.

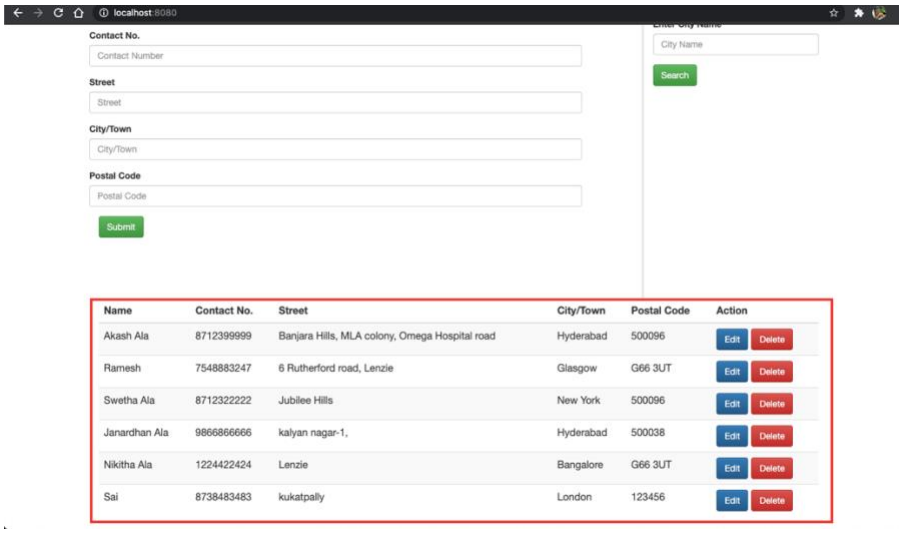

Figure 14

Figure 15: Search field is provided to search for a particular contact by city.
Note: The user has to enter the full name of city not just the few characters. For example, in order to search for contact with city Glasgow, it is required to type Glasgow not just gla. Search field is Case insensitive.

Figure 15

Figure 16: If the user tries to find the contact with the city which is not in the database, then dialogue box pop up saying “No Contact found on this city”.

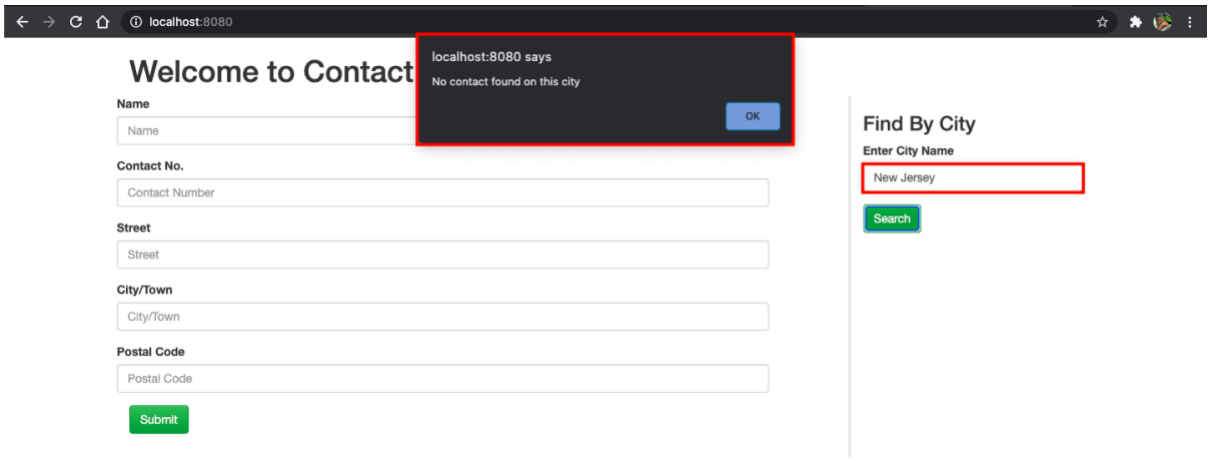

Figure 16

Figure 17: While saving the contacts, if any of the field is not filled then a warning is displayed showing “Please fill in this field”.

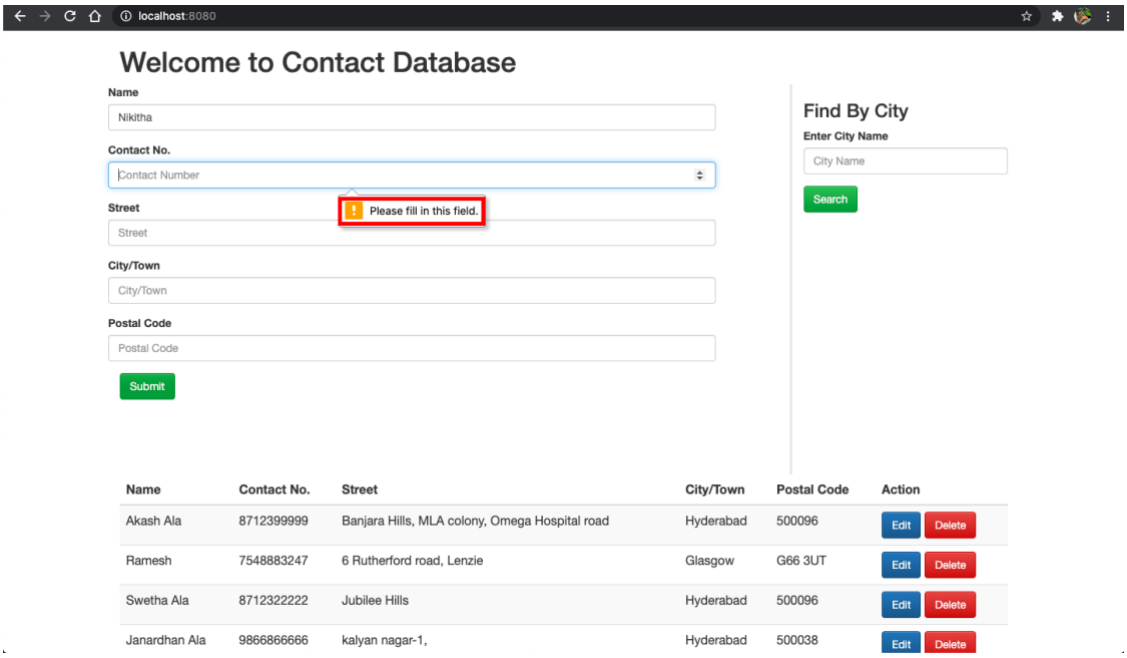

Figure 17

4. Appendix

4.1 pom.xml

Maven pom file to get all the dependency jars. Spring-boot-starter-parent 2.4.4 version is used.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.4</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.contactdb</groupId>
  <artifactId>assignment-contacts-database</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>ContactDB-1</name>
  <description>ContactDB</description>
  <packaging>war</packaging>
  <properties>
    <java.version>1.8</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web-services</artifactId>
    </dependency>
```

```
    <dependency>
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
      <scope>provided</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
    <finalName>${artifactId}</finalName>
  </build>

</project>
```

4.2 ContactController.java

```
package com.contactdb.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
```

```java
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;

import com.contactdb.entity.Contact;
import com.contactdb.repo.ContactRepository;

@Controller
@RequestMapping({ "/contact", "/contacts" })
public class ContactController {

  @Autowired
  ContactRepository contactRepository;

  /**
   * This API is for fetching all contacts in Database
   *
   * */

  @GetMapping
  public ResponseEntity<?> getAll() {
    return ResponseEntity.ok(contactRepository.findAll());
  }

  /**
   *
   * This API is for retrieving one single contact on Id
   *
   * */

  @GetMapping("/{id}")
  public ResponseEntity<?> getOne(@PathVariable long id) {
    return ResponseEntity.ok(contactRepository.findById(id));
  }

  /**
```

```
 *
 * This API is saving contact in Database
 *
 * */

@PostMapping
public ResponseEntity<?> save(@RequestBody Contact contact){

  if(contactRepository.existsByNumber(contact.getNumber()))
     return new ResponseEntity<Contact>(contact,HttpStatus.CONFLICT);

  contact=contactRepository.save(contact);
  return new ResponseEntity<Contact>(contact,HttpStatus.CREATED);
}

/**
 *
 * This API is for deleting one single contact on Id
 *
 * */

@DeleteMapping("/{id}")
public ResponseEntity<?> delete(@PathVariable long id){
  if(!contactRepository.existsById(id))
     return  new ResponseEntity<>(HttpStatus.NOT_FOUND);

     contactRepository.deleteById(id);
     return  new ResponseEntity<>(HttpStatus.NO_CONTENT);
}

/**
 *
 * This API is for updating contact
 *
 * */

@PutMapping
public ResponseEntity<?> update(@RequestBody Contact contact){
  contact = contactRepository.save(contact);
  return new ResponseEntity<Contact>(contact,HttpStatus.OK);
```

```
  }


  /**
   *
   * This API is for fetching contacts on city name
   *
   * */

  @GetMapping("/city/{name}")
  public ResponseEntity<?> contactsOfTown(@PathVariable String name){
     return ResponseEntity.ok(contactRepository.findByAddress_CityIgnoreCase(name));
  }


}
```

4.3 Address.java

```
package com.contactdb.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Address {

  @Id@GeneratedValue(strategy=GenerationType.IDENTITY)
  long id;

  String street;

  String city;

  String postCode;

  public long getId() {
```

```
    return id;
  }

  public void setId(long id) {
    this.id = id;
  }

  public String getStreet() {
    return street;
  }

  public void setStreet(String street) {
    this.street = street;
  }

  public String getCity() {
    return city;
  }

  public void setCity(String city) {
    this.city = city;
  }

  public String getPostCode() {
    return postCode;
  }

  public void setPostCode(String postCode) {
    this.postCode = postCode;
  }


}
```

4.4 Contact.java

```
package com.contactdb.entity;

import javax.persistence.CascadeType;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class Contact {

  @Id@GeneratedValue(strategy=GenerationType.IDENTITY)
  long id;

  String name;

  String  number;

  @ManyToOne(cascade=CascadeType.ALL)
  Address address;

  public long getId() {
    return id;
  }

  public void setId(long id) {
    this.id = id;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getNumber() {
    return number;
  }

  public void setNumber(String number) {
```

```
    this.number = number;
  }

  public Address getAddress() {
    return address;
  }

  public void setAddress(Address address) {
    this.address = address;
  }

}
```

4.5 AddressRepository.java

```
package com.contactdb.repo;

import com.contactdb.entity.Address;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface AddressRepository extends CrudRepository<Address,Long> {

}
```

4.6 ContactRepository.java

```
package com.contactdb.repo;

import java.util.List;

import org.springframework.data.repository.CrudRepository;
```

```
import org.springframework.stereotype.Repository;

import com.contactdb.entity.Contact;

@Repository
public interface ContactRepository extends CrudRepository<Contact,Long>{

  boolean existsByNumber(String number);
  List<Contact> findByAddress_CityIgnoreCase(String city);
}
```

4.7 ContactDbApplication

```
package com.contactdb;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication
public class ContactDbApplication extends SpringBootServletInitializer{

  public static void main(String[] args) {
    SpringApplication.run(ContactDbApplication.class, args);
  }

}
```

4.8 index.html

```
<!-- Latest compiled and minified CSS -->
<html>
<link rel="stylesheet"
  href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
  integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
  crossorigin="anonymous">
```

```
<!-- Optional theme -->
<link rel="stylesheet"
   href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
   integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
   crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->

<style>
.vl {
   border-right: 3px solid #e6e6e6;;
   height: 500px;
}
</style>
<div class="container">
   <h1>Welcome to Contact Database</h1>
   <div class="row">
      <div class="col-md-8">
         <form class="form-horizontal" action="contact" id="contact-form">
            <input type="hidden" name="id" value="" id="id"/>
            <input type="hidden" name="addressId" value="" id="addressId"/>
            <div class="form-group">
               <label for="inputName">Name</label> <input type="text"
                  class="form-control" name="name" id="name" placeholder="Name" required="required">
            </div>
            <div class="form-group">
               <label for="inputContactNo">Contact No.</label> <input
                  type="number" class="form-control" id="number" name="number" placeholder="Contact Number"
required="required">
            </div>
            <div class="form-group">
               <label for="inputStreet">Street</label> <input type="text"
                  class="form-control" name="street" id="street" placeholder="Street" required="required">
            </div>
            <div class="form-group">
               <label for="inputCityName">City/Town</label> <input type="text"
                  class="form-control" name="city" id="city" placeholder="City/Town" required="required">
            </div>
            <div class="form-group">
               <label for="inputPostCode">Postal Code</label> <input
```

```
            type="text" class="form-control" id="postCode" name="postCode" placeholder="Postal Code"
required="required">
          </div>

          <button type="submit" class="btn btn-success">Submit</button>
        </form>
      </div>
      <div class="vl col-md-1"></div>
      <div class="col-md-3">
        <h3>Find By City</h3>
        <div class="form-group">
          <label for="inputCityName">Enter City Name</label> <input
            type="text" class="form-control" placeholder="City Name" id="searchTxt">
        </div>
        <button type="button" class="btn btn-success" id="searchBtn">Search</button>
      </div>
    </div>
    <table class="table table-striped" id="table">
      <thead>
        <tr>
          <th>Name</th>
          <th>Contact No.</th>
          <th>Street</th>
          <th>City/Town</th>
          <th>Postal Code</th>
          <th>Action</th>
        </tr>
      </thead>
      <tbody id="tbody">
      </tbody>
    </table>
</div>
<script
    src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"
    integrity="sha512-
894YE6QWD5I59HgZOGReFYm4dnWc1Qt5NtvYSaNcOP+u1T9qYdvdihz0PPSiiqn/+/3e7Jo4EaG7TubfWGUrM
Q=="
    crossorigin="anonymous"></script>
<script type="text/javascript">
    $(document)
```

```
        .ready(
            function() {


                var url = "contacts";
                $.get(url,function(data) {
                    console.log(data)
                    if (data) {
                        $("#table > tbody").html("");
                        $.each(data,function(k, v) {
                            $("#table > tbody").append("<tr id='id'"+v.id+"><td>"
                                    + v.name
                                    + "</td><td>"
                                    + v.number
                                    + "</td><td>"
                                    + v.address.street
                                    + "</td><td>"
                                    + v.address.city
                                    + "</td><td>"
                                    + v.address.postCode
                                    + "</td><td><button type='button' class='btn btn-primary' style='margin-right:5px'
id='"+v.id+"'>Edit</button><button type='button' class='btn btn-danger' id='"+v.id+"'>Delete</button></td></tr>");
                        });
                    }
                });


                $(document).on('click', '.btn-primary', function() {
                    var id = $(this).attr('id');
                    var url="contact/"+id
                    $.get(url,function(data){
                        $('#id').val(data.id)
                        $('#name').val(data.name)
                        $('#number').val(data.number)
                        $('#addressId').val(data.address.id)
                        $('#street').val(data.address.street)
                        $('#city').val(data.address.city)
                        $('#postCode').val(data.address.postCode)

                    });
                });
```

```
$(document).on('click', '.btn-danger', function() {
  var id = $(this).attr('id');
  $(this).closest('tr').remove();
  var url = "contact/" + id;
  $.ajax({
    url : url,
    type : 'DELETE',
    success : function(result,status,xhr) {
      if(xhr.status == 204)
        alert('Contact deleted successfully');
      else if(xhr.status == 404)
        alert('Contact does not exist');
      window.location.reload();

    }
  });
});

$('#contact-form').submit(function(e){
  e.preventDefault();
  var methodType = 'POST';

  var formData = $('#contact-form').serializeArray();
  formData = objectifyForm(formData);
  if(formData.id>0)
    methodType='PUT';
  var url = "contact";
  $.ajax({
    url : url,
    type :methodType,
    data:JSON.stringify(formData),
    contentType: 'application/json',
    success : function(result,status,xhr) {
      if(xhr.status == 201)
        alert('Contact saved successfully');
      else if(xhr.status == 200)
        alert('Contact updated successfully');
      window.location.reload();
      $('#contact-form').find(":input").val("");
```

```
            },
              error: function(xhr, textStatus, errorThrown){
                if(xhr.status == 409)
                    alert('Contact already exists');
                }
            });


        });

        $('#searchBtn').click(function(){
          var city = $('#searchTxt').val()
          if(city){
            var url = "contact/city/"+city;
            $.get(url,function(data){
              if(!data.length>0){
                alert("No contact found on this city");
                return;
              }

              $("#table > tbody").html("");
                $.each(data,function(k, v) {
                  $("#table > tbody").append("<tr id='id'"+v.id+"><td>"
                                + v.name
                                + "</td><td>"
                                + v.number
                                + "</td><td>"
                                + v.address.street
                                + "</td><td>"
                                + v.address.city
                                + "</td><td>"
                                + v.address.postCode
                                + "</td><td><button type='button' class='btn btn-primary' style='margin-right:5px'
id='"+v.id+"'>Edit</button><button type='button' class='btn btn-danger' id='"+v.id+"'>Delete</button></td></tr>");
                            });

            });
          }
        });
```

```
            function objectifyForm(formArray) {
                //serialize data function
                var returnArray = {};
                var address = {};
                for (var i = 0; i < formArray.length; i++){
                  if(formArray[i]['name'] == "street" ||formArray[i]['name'] == "city"||formArray[i]['name'] ==
"postCode" || formArray[i]['name'] == "addressId"){
                    if(formArray[i]['name'] == "addressId")
                      address['id'] = formArray[i]['value'];
                    else
                      address[formArray[i]['name']] = formArray[i]['value'];
                  }
                  else
                    returnArray[formArray[i]['name']] = formArray[i]['value'];
                }
                returnArray['address'] = address;
                return returnArray;
              }
            });
</script>
<script
  src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
  integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"
  crossorigin="anonymous"></script>
</html>
```

5. Features that can be implemented later:

- A more rich UI can be developed using angular or reactjs.
- Country codes can be added
- Searching a city just with first few characters of city
- Adding more field like first name, last name, gender, email, profession etc.
- Providing a separate button “Show all contacts” i.e. when clicked displays all the contacts in the database.
- Searching the contact not only by city but also using other fields.