SOURCE CODE FOR PHASE END PROJECT VACCINATION CENTER

ServletInitializer.java (src/main/java/com.example.vaccinationcenter) :

```
package com.example.vaccinationcenter;

import org.springframework.boot.builder.SpringApplicationBuilder;
import
org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder
application) {
        return application.sources(VaccinationCenterApplication.class);
    }

}
```

VaccinationCenterApplication.java (src/main/java/com.example.vaccinationcenter) :

```
package com.example.vaccinationcenter;

import com.example.vaccinationcenter.entities.Citizen;
import com.example.vaccinationcenter.entities.VaccinationCenter;
import com.example.vaccinationcenter.repository.CitizenRepository;
import com.example.vaccinationcenter.repository.VaccinationCenterRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class VaccinationCenterApplication implements CommandLineRunner {

    @Autowired
    private CitizenRepository citizenRepository;

    @Autowired
    private VaccinationCenterRepository vaccinationCenterRepository;

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

    @Override
    public void run(String... args) throws Exception {
```

```
        Citizen citizen = new Citizen();
        citizen.setCity("CITY_1");
        citizen.setName("NAME_1");

        citizenRepository.save(citizen);

        VaccinationCenter vc = new VaccinationCenter();
        vc.setName("CLINIC_1");
        vc.setAddress("BLR1");

        vaccinationCenterRepository.save(vc);

        citizen.setCenter(vc);

        citizenRepository.save(citizen);
    }
}
```

CustomExceptionHandler.java (src/main/java/com.example.vaccinationcenter.advice) :

```
package com.example.vaccinationcenter.advice;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.Collections;
import java.util.Comparator;

@RestControllerAdvice
public class CustomExceptionHandler {

    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResponseEntity<ValidationErrorResponse>
handleValidationException(MethodArgumentNotValidException ex) {
        ValidationErrorResponse validationErrorResponse = new
ValidationErrorResponse();
        ex.getBindingResult().getFieldErrors().forEach(error -> {
            validationErrorResponse.addError(error.getField(),
error.getDefaultMessage());
        });
        Collections.sort(validationErrorResponse.getErrors(),
Comparator.comparing(ValidationErrorResponse.ErrorDetail::getField));
        return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(validationErrorResponse);
    }
```

```
    // Other exception handlers can be added here

}
```

ValidationErrorResponse.java (src/main/java/com.example.vaccinationcenter.advice) :

```
package com.example.vaccinationcenter.advice;

import java.util.ArrayList;
import java.util.List;

public class ValidationErrorResponse {

    private List<ErrorDetail> errors = new ArrayList<>();

    public void addError(String field, String message) {
        ErrorDetail errorDetail = new ErrorDetail(field, message);
        errors.add(errorDetail);
    }

    public List<ErrorDetail> getErrors() {
        return errors;
    }

    public void setErrors(List<ErrorDetail> errors) {
        this.errors = errors;
    }

    public static class ErrorDetail {
        private String field;
        private String message;

        public ErrorDetail(String field, String message) {
            this.field = field;
            this.message = message;
        }

        public String getField() {
            return field;
        }

        public void setField(String field) {
            this.field = field;
        }

        public String getMessage() {
            return message;
        }
```

```
        public void setMessage(String message) {
            this.message = message;
        }
    }
}
```

FilterConfig.java (src/main/java/com.example.vaccinationcenter.config) :

```
package com.example.vaccinationcenter.config;

import com.example.vaccinationcenter.filter.UserAuthenticationFilter;
import com.example.vaccinationcenter.repository.UserRepository;
import jakarta.servlet.DispatcherType;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.Ordered;

@Configuration
public class FilterConfig {

    @Bean
    public FilterRegistrationBean<UserAuthenticationFilter>
registrationFilter(@Autowired UserRepository userRepository) {
        FilterRegistrationBean<UserAuthenticationFilter> registrationBean =
new FilterRegistrationBean<>();
        registrationBean.setFilter(new
UserAuthenticationFilter(userRepository));
        registrationBean.addUrlPatterns("/citizens","/citizens/*","/vaccinatio
ncenter","/vaccinationcenter/*","/me");
        registrationBean.setOrder(Ordered.HIGHEST_PRECEDENCE);
        registrationBean.setDispatcherTypes(DispatcherType.REQUEST);
        return registrationBean;
    }
}
```

AuthenticateController.java (src/main/java/com.example.vaccinationcenter.controllers) :

```
package com.example.vaccinationcenter.controllers;

import com.example.vaccinationcenter.dtos.LoginRequestDto;
import com.example.vaccinationcenter.dtos.LoginResponse;
import com.example.vaccinationcenter.dtos.RegistrationRequestDto;
import com.example.vaccinationcenter.entities.User;
import com.example.vaccinationcenter.repository.UserRepository;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
```

```java
import jakarta.validation.Valid;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.Objects;

@RestController
@RequestMapping
public class AuthenticateController {

  @Autowired
  private UserRepository userRepository;

  @PostMapping(value = "/authenticate", produces = "application/json")
  public LoginResponse authenticate(@Valid @RequestBody LoginRequestDto
loginRequest, HttpServletRequest request, HttpServletResponse response) {
    LoginResponse loginResponse = new LoginResponse();
    User user = userRepository.findByEmail(loginRequest.getEmail());
    if(user == null)
    {
      throw new RuntimeException("user not found with email:
"+loginRequest.getEmail());
    }
    if(!Objects.equals(user.getPassword(), loginRequest.getPassword()))
    {
      throw new RuntimeException("password not matching with email:
"+loginRequest.getEmail());
    }
    String token =
Base64.getEncoder().encodeToString(user.getEmail().getBytes(StandardCharsets.U
TF_8));
    loginResponse.setName(user.getName());
    loginResponse.setEncodedToken(token);
    return loginResponse;
  }

  @PostMapping(value = "/registration", produces = "application/json")
  public int register(@Valid @RequestBody RegistrationRequestDto
registrationRequestDto) {
    User user = new User();
```

```
    user.setEmail(registrationRequestDto.getEmail());
    user.setName(registrationRequestDto.getName());
    user.setPassword(registrationRequestDto.getPassword());
    if (userRepository.findByEmail(user.getEmail()) != null) {
      throw new RuntimeException("user already exists with email: " +
user.getEmail());
    }
    userRepository.save(user);
    return 1;
  }

  @GetMapping(value = "/me", produces = "application/json")
  public LoginResponse me(HttpServletRequest request, HttpServletResponse
response) throws IOException {
    String authorizationHeader = request.getHeader("authorization");

    byte emailBytes[] =
Base64.getDecoder().decode(authorizationHeader.getBytes(StandardCharsets.UTF_8
));
    String email = new String(emailBytes);
    User user = userRepository.findByEmail(email);
    if (user == null) {
      throw new RuntimeException("user not found with email: " + email);
    }

    LoginResponse loginResponse = new LoginResponse();
    loginResponse.setEmail(email);
    loginResponse.setName(user.getName());
    loginResponse.setEncodedToken(authorizationHeader);
    return loginResponse;

  }
}
```

CitizenController.java (src/main/java/com.example.vaccinationcenter.controllers) :

```
package com.example.vaccinationcenter.controllers;

import com.example.vaccinationcenter.dtos.CitizenDto;
import com.example.vaccinationcenter.entities.Citizen;
import com.example.vaccinationcenter.entities.VaccinationCenter;
import com.example.vaccinationcenter.repository.CitizenRepository;
import com.example.vaccinationcenter.services.VaccinationCenterService;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
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;
```

```java
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.StreamSupport;

@RestController
@RequestMapping(value = "citizens", produces = "application/json")
public class CitizenController {

  @Autowired
  private CitizenRepository citizenRepository;

  @Autowired
  private VaccinationCenterService vaccinationCenterService;

  @GetMapping("/{id}")
  public Citizen findCitizen(@PathVariable long id) {
    return citizenRepository.findById(id).get();
  }

  @GetMapping()
  public List<Citizen> findCitizens() {
    List<Citizen> citizens =
StreamSupport.stream(citizenRepository.findAll().spliterator(), false)
        .collect(Collectors.toList());

    return citizens;
  }

  @PostMapping
  public Citizen addCitizen(@Valid @RequestBody CitizenDto citizenDto) {
    Citizen citizen = new Citizen();
    VaccinationCenter vaccinationCenter =
vaccinationCenterService.getVaccinationCenter(citizenDto.getCenterId());

    citizen.setName(citizenDto.getName());
    citizen.setCity(citizenDto.getCity());
    citizen.setCenter(vaccinationCenter);

    return citizenRepository.save(citizen);
  }

  @PutMapping
  public Citizen updateCitizen( @Valid @RequestBody CitizenDto citizenDto) {
```

```
    long centerId = citizenDto.getCenterId();
    VaccinationCenter vaccinationCenter =
vaccinationCenterService.getVaccinationCenter(centerId);
    Citizen citizen = findCitizen(citizenDto.getId());
    citizen.setCenter(vaccinationCenter);
    citizen.setName(citizenDto.getName());
    citizen.setCity(citizenDto.getCity());
    citizen.setDoesCount(citizenDto.getDoesCount());
    return citizenRepository.save(citizen);
  }

  @DeleteMapping("/{id}")
  public Boolean deleteCitizen(@PathVariable long id) {
    citizenRepository.deleteById(id);
    return true;
  }

  @GetMapping("/center/{centerId}")
  public List<Citizen> findByCenterId(@PathVariable long centerId) {
    return citizenRepository.findAllByCenterId(centerId);
  }
}
```

VaccinationCenterController.java (src/main/java/com.example.vaccinationcenter.controllers) :

```
package com.example.vaccinationcenter.controllers;

import com.example.vaccinationcenter.dtos.VaccinationCenterDto;
import com.example.vaccinationcenter.entities.VaccinationCenter;
import com.example.vaccinationcenter.services.VaccinationCenterService;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
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 org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping(value = "vaccinationcenter", produces = "application/json")
public class VaccinationCenterController {

  @Autowired
```

```
  private VaccinationCenterService vaccinationCenterService;

  @GetMapping("/{id}")
  public VaccinationCenter findVaccinationCenter(@PathVariable long id) {
    return vaccinationCenterService.getVaccinationCenter(id);
  }
  @DeleteMapping("/{id}")
  public boolean deleteVaccinationCenter(@PathVariable long id) {
    return vaccinationCenterService.delete(id);
  }

  @GetMapping()
  public List<VaccinationCenter> findVaccinationCenters() {
    return vaccinationCenterService.getVaccinationCenters();
  }

  @PostMapping
  public VaccinationCenter addNewVaccinationCenter(@Valid @RequestBody
VaccinationCenterDto vaccinationCenterDto) {

    VaccinationCenter vaccinationCenter = new VaccinationCenter();
    vaccinationCenter.setName(vaccinationCenterDto.getName());
    vaccinationCenter.setAddress(vaccinationCenterDto.getCity());
    return vaccinationCenterService.addOrUpdate(vaccinationCenter);
  }

  @PutMapping
  public VaccinationCenter updateVaccinationCenter(@Valid @RequestBody
VaccinationCenterDto vaccinationCenterDto) {
    VaccinationCenter vaccinationCenter = vaccinationCenterService.
        getVaccinationCenter(vaccinationCenterDto.getId());
    vaccinationCenter.setName(vaccinationCenterDto.getName());
    vaccinationCenter.setAddress(vaccinationCenterDto.getCity());
    return vaccinationCenterService.addOrUpdate(vaccinationCenter);
  }
}
```

ViewController.java (src/main/java/com.example.vaccinationcenter.controllers) :

```
package com.example.vaccinationcenter.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@Controller
public class ViewController {
```

```java
  @GetMapping("citizens")
  public String citizensPage() {
    return "citizens";
  }

  @GetMapping("citizens/{id}")
  public String citizensPage(@PathVariable int id, Model model) {
    model.addAttribute("id", id);
    return "view_citizen";
  }


  @GetMapping("citizens/center/{id}")
  public String citizensPageByCenterId(@PathVariable int id, Model model) {
    model.addAttribute("id", id);
    return "citizen_center";
  }

  @GetMapping("vaccinationcenter/{id}")
  public String vaccinationcenterByCenterId(@PathVariable int id, Model model)
{
    model.addAttribute("id", id);
    return "citizen_center";
  }

  @GetMapping("vaccinationcenter")
  public String vaccinationCenterPage() {
    return "vaccination_center";
  }

  @GetMapping(value = {"/","/login"})
  public String loginPage() {
    return "login";
  }

  @GetMapping("/register")
  public String registrationPage() {
    return "register";
  }



}
```

CitizenDto.java (src/main/java/com.example.vaccinationcenter.dtos) :

```java
package com.example.vaccinationcenter.dtos;

import com.example.vaccinationcenter.entities.Citizen;
import com.example.vaccinationcenter.validators.antn.CitizenValidator;

@CitizenValidator
public class CitizenDto {

  private String httpMethod;

  private Long id;
  private String name;
  private String city;
  private int doesCount;
  private Long centerId;

  public String getHttpMethod() {
    return httpMethod;
  }

  public void setHttpMethod(String httpMethod) {
    this.httpMethod = httpMethod;
  }

  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 getCity() {
    return city;
  }

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

```
  public int getDoesCount() {
    return doesCount;
  }

  public void setDoesCount(int doesCount) {
    this.doesCount = doesCount;
  }

  public Long getCenterId() {
    return centerId;
  }

  public void setCenterId(Long centerId) {
    this.centerId = centerId;
  }
}
```

LoginRequestDto.java (src/main/java/com.example.vaccinationcenter.dtos) :

```
package com.example.vaccinationcenter.dtos;

public class LoginRequestDto {

  private String email;
  private String password;

  public String getEmail() {
    return email;
  }

  public void setEmail(String email) {
    this.email = email;
  }

  public String getPassword() {
    return password;
  }

  public void setPassword(String password) {
    this.password = password;
  }
}
```

LoginResponse.java (src/main/java/com.example.vaccinationcenter.dtos) :

```
package com.example.vaccinationcenter.dtos;

public class LoginResponse {
```

```
  private String email;
  private String name;
  private String encodedToken;

  public String getEmail() {
    return email;
  }

  public void setEmail(String email) {
    this.email = email;
  }

  public String getName() {
    return name;
  }

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

  public String getEncodedToken() {
    return encodedToken;
  }

  public void setEncodedToken(String encodedToken) {
    this.encodedToken = encodedToken;
  }
}
```

RegistrationRequestDto.java (src/main/java/com.example.vaccinationcenter.dtos) :

```
package com.example.vaccinationcenter.dtos;

import
com.example.vaccinationcenter.validators.antn.RegistrationRequestValidator;

@RegistrationRequestValidator
public class RegistrationRequestDto extends LoginRequestDto {
  private String name;

  public String getName() {
    return name;
  }

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

VaccinationCenterDto.java (src/main/java/com.example.vaccinationcenter.dtos) :

```
package com.example.vaccinationcenter.dtos;

import
com.example.vaccinationcenter.validators.antn.VaccinationCenterValidator;

@VaccinationCenterValidator
public class VaccinationCenterDto {

  private String httpMethod;
  private Long id;
```

```
  private String name;
  private String city;

  public String getHttpMethod() {
    return httpMethod;
  }

  public void setHttpMethod(String httpMethod) {
    this.httpMethod = httpMethod;
  }

  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 getCity() {
    return city;
  }

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

Citizen.java (src/main/java/com.example.vaccinationcenter.entities) :

```
package com.example.vaccinationcenter.entities;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToMany;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Transient;


@Entity(name = "TBL_CITIZEN")
public class Citizen {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;

    private String name;
    private String city;
    @ManyToOne
    @JoinColumn(name = "centerId")
    private VaccinationCenter center;
//    private int centerId;
    private int doesCount;

    @Transient
    private String vaccinationStatus;

    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 getCity() {
        return city;
    }

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

    public int getDoesCount() {
        return doesCount;
    }

    public void setDoesCount(int doesCount) {
        this.doesCount = doesCount;
    }

    public VaccinationCenter getCenter() {
        return center;
    }
```

```
    public void setCenter(VaccinationCenter center) {
        this.center = center;
    }

    public String getVaccinationStatus() {
        return getDoesCount() == 2 ? "FULLY_VACCINATED": "NOT_VACCINATED";
    }

    public void setVaccinationStatus(String vaccinationStatus) {
        this.vaccinationStatus = vaccinationStatus;
    }
}
```

User.java (src/main/java/com.example.vaccinationcenter.entities) :

```
package com.example.vaccinationcenter.entities;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity(name = "TBL_USER")

public class User {

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

  private String name;
  @Column(unique = true)
  private String email;
  private String password;

  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 getEmail() {
    return email;
  }

  public void setEmail(String email) {
    this.email = email;
  }

  public String getPassword() {
    return password;
  }

  public void setPassword(String password) {
    this.password = password;
  }
}
```

VaccinationCenter.java (src/main/java/com.example.vaccinationcenter.entities) :

```
package com.example.vaccinationcenter.entities;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;

import java.util.List;

@Entity(name = "TBL_VACCINATION_CENTER")
public class VaccinationCenter {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  private String name;
  private String address;
  @OneToMany( mappedBy = "center",cascade = CascadeType.REMOVE, orphanRemoval
= true)
  private List<Citizen> citizenList;

  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 getAddress() {
    return address;
  }

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

}
```

UserAuthenticationFilter.java (src/main/java/com.example.vaccinationcenter.filter) :

```
package com.example.vaccinationcenter.filter;


import com.example.vaccinationcenter.entities.User;
import com.example.vaccinationcenter.repository.UserRepository;
import jakarta.servlet.Filter;
import jakarta.servlet.FilterChain;
import jakarta.servlet.FilterConfig;
import jakarta.servlet.ServletException;
import jakarta.servlet.ServletRequest;
import jakarta.servlet.ServletResponse;
import jakarta.servlet.annotation.WebFilter;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.HashSet;
import java.util.Set;
```

```
public class UserAuthenticationFilter implements Filter {

  private UserRepository userRepository;

  public UserAuthenticationFilter(UserRepository userRepository) {
    this.userRepository = userRepository;
  }

  @Override
  public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException, ServletException
{
    boolean flag = validateToken((HttpServletRequest) servletRequest,
(HttpServletResponse) servletResponse);
    if(!flag){
      return;
    }
    filterChain.doFilter(servletRequest,servletResponse);
  }

  private Set<String> servletPaths = new HashSet<>(){{
    add("citizens");
    add("vaccinationcenter");
    add("me");
  }};

  private boolean validateToken(HttpServletRequest request,
HttpServletResponse response) throws IOException {
    String servletPath = request.getServletPath();
    System.out.println(servletPath);

    String api = request.getHeader("x-api");

    if(!Boolean.valueOf(api))
    {
      return true;
    }

    String authorizationHeader = request.getHeader("authorization");
    if (StringUtils.isBlank(authorizationHeader)) {
      System.out.println("auth token missing, so redirecting to login");
      response.setStatus(302);
      return false;
    }
    byte emailBytes[] =
Base64.getDecoder().decode(authorizationHeader.getBytes(StandardCharsets.UTF_8
));
    String email = new String(emailBytes);
```

```
    User user = userRepository.findByEmail(email);
    if (user == null) {
      throw new RuntimeException("user not found with email: " + email);
    }
    return true;
  }
  @Override
  public void init(FilterConfig filterConfig) throws ServletException {
    Filter.super.init(filterConfig);
  }

  @Override
  public void destroy() {
    Filter.super.destroy();
  }

  private String getBaseUrl(HttpServletRequest request)
  {
    String baseUrl = request.getScheme() + "://" + request.getServerName() +
":" + request.getServerPort() + request.getContextPath();

    System.out.println(baseUrl);
    return baseUrl;
  }
}
```

CitizenRepository.java (src/main/java/com.example.vaccinationcenter.repository) :

```
package com.example.vaccinationcenter.repository;

import com.example.vaccinationcenter.entities.Citizen;
import org.springframework.data.repository.CrudRepository;

import java.util.List;

public interface CitizenRepository extends CrudRepository<Citizen,Long> {

  public List<Citizen> findAllByCenterId(long centerId);
}
```

UserRepository.java (src/main/java/com.example.vaccinationcenter.repository) :

```
package com.example.vaccinationcenter.repository;

import com.example.vaccinationcenter.entities.User;
import org.springframework.data.repository.CrudRepository;

public interface UserRepository extends CrudRepository<User, Long> {
```

```
  public User findByEmail(String email);
}
```

VaccinationCenterRepository.java (src/main/java/com.example.vaccinationcenter.repository) :

```
package com.example.vaccinationcenter.repository;

import com.example.vaccinationcenter.entities.Citizen;
import com.example.vaccinationcenter.entities.VaccinationCenter;
import org.springframework.data.repository.CrudRepository;

public interface VaccinationCenterRepository extends
CrudRepository<VaccinationCenter,Long> {
}
```

VaccinationCenterService.java (src/main/java/com.example.vaccinationcenter.services) :

```
package com.example.vaccinationcenter.services;

import com.example.vaccinationcenter.entities.Citizen;
import com.example.vaccinationcenter.entities.VaccinationCenter;
import com.example.vaccinationcenter.repository.CitizenRepository;
import com.example.vaccinationcenter.repository.VaccinationCenterRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.StreamSupport;

@Service
public class VaccinationCenterService {

  @Autowired
  private VaccinationCenterRepository vaccinationCenterRepository;

  @Autowired
  private CitizenRepository citizenRepository;

  public VaccinationCenter addOrUpdate(VaccinationCenter center) {
    return vaccinationCenterRepository.save(center);
  }

  public List<VaccinationCenter> getVaccinationCenters() {
```

```
    List<VaccinationCenter> vaccinationCenters =
StreamSupport.stream(vaccinationCenterRepository.findAll().spliterator(),
false)
        .collect(Collectors.toList());

    return vaccinationCenters;

  }

  public boolean delete(long id){
    VaccinationCenter vaccinationCenter =
vaccinationCenterRepository.findById(id).get();

    List<Citizen> citizenList = citizenRepository.findAllByCenterId(id);
    citizenList.forEach(e->{
      e.setCenter(null);
      citizenRepository.save(e);
    });

    vaccinationCenterRepository.delete(vaccinationCenter);
    return true;
  }
  public VaccinationCenter getVaccinationCenter(long id) {
    return vaccinationCenterRepository.findById(id).get();
  }
}
```

CitizenValidator.java (src/main/java/com.example.vaccinationcenter.validators.antn) :

```
package com.example.vaccinationcenter.validators.antn;

import com.example.vaccinationcenter.validators.impl.CitizenDtoValidator;
import jakarta.validation.Constraint;
import jakarta.validation.Payload;
import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
@Documented
@Constraint(validatedBy = {CitizenDtoValidator.class})
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface CitizenValidator {
    String message() default "Invalid Vaccination center data";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```

LoginRequestValidator.java
(src/main/java/com.example.vaccinationcenter.validators.antn) :

```java
package com.example.vaccinationcenter.validators.antn;

import com.example.vaccinationcenter.validators.impl.CitizenDtoValidator;
import com.example.vaccinationcenter.validators.impl.LoginRequestDtoValidator;
import jakarta.validation.Constraint;
import jakarta.validation.Payload;

import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Documented
@Constraint(validatedBy = {LoginRequestDtoValidator.class})
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface LoginRequestValidator {
  String message() default "Invalid Login Request data";
  Class<?>[] groups() default {};
  Class<? extends Payload>[] payload() default {};
}
```

RegistrationRequestValidator.java
(src/main/java/com.example.vaccinationcenter.validators.antn) :

```java
package com.example.vaccinationcenter.validators.antn;

import com.example.vaccinationcenter.validators.impl.LoginRequestDtoValidator;
import
com.example.vaccinationcenter.validators.impl.RegistrationRequestDtoValidator;
import jakarta.validation.Constraint;
import jakarta.validation.Payload;

import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Documented
@Constraint(validatedBy = {RegistrationRequestDtoValidator.class})
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface RegistrationRequestValidator {
  String message() default "Invalid Registration quest data";
```

```
  Class<?>[] groups() default {};
  Class<? extends Payload>[] payload() default {};
}
```

VaccinationCenterValidator.java (src/main/java/com.example.vaccinationcenter.validators.antn) :

```
package com.example.vaccinationcenter.validators.antn;

import
com.example.vaccinationcenter.validators.impl.VaccinationCenterDtoValidator;
import jakarta.validation.Constraint;
import jakarta.validation.Payload;

import java.lang.annotation.*;

@Documented
@Constraint(validatedBy = {VaccinationCenterDtoValidator.class})
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface VaccinationCenterValidator {
    String message() default "Invalid Vaccination center data";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```

CitizenDtoValidator.java (src/main/java/com.example.vaccinationcenter.validators.impl) :

```
package com.example.vaccinationcenter.validators.impl;

import com.example.vaccinationcenter.dtos.CitizenDto;
import com.example.vaccinationcenter.dtos.VaccinationCenterDto;
import com.example.vaccinationcenter.entities.Citizen;
import com.example.vaccinationcenter.entities.VaccinationCenter;
import com.example.vaccinationcenter.validators.antn.CitizenValidator;
import jakarta.validation.ConstraintValidator;
import jakarta.validation.ConstraintValidatorContext;
import org.apache.commons.lang3.StringUtils;

public class CitizenDtoValidator implements
ConstraintValidator<CitizenValidator, CitizenDto> {

    @Override
    public boolean isValid(CitizenDto dto, ConstraintValidatorContext context)
{
        return getRequestDtoValidator(dto).validate(dto, context);
    }

    @Override
```

```
    public void initialize(CitizenValidator constraintAnnotation) {
    }

    private static RequestDtoValidator<CitizenDto>
getRequestDtoValidator(CitizenDto dto) {
        if (dto == null)
            return null;
        if (dto.getHttpMethod().equalsIgnoreCase("post"))
            return new PostRequestValidator();
        if (dto.getHttpMethod().equalsIgnoreCase("put"))
            return new PutRequestValidator();

        return null;
    }

    private static class PostRequestValidator implements
RequestDtoValidator<CitizenDto> {

        public boolean validate(CitizenDto citizen, ConstraintValidatorContext
context) {
            boolean flag = true;
            if (citizen == null) {
                flag = false;
                context.disableDefaultConstraintViolation();
                context.buildConstraintViolationWithTemplate("Citizen details
required").addPropertyNode("citizen")
                        .addConstraintViolation();
                return flag;
            }

            if (StringUtils.isBlank(citizen.getName())) {
                flag = false;
                context.disableDefaultConstraintViolation();
                context.buildConstraintViolationWithTemplate("Citizen name
required").addPropertyNode("name")
                        .addConstraintViolation();
            }

            if (!(citizen.getName().length() > 2 && citizen.getName().length()
< 30)) {
                flag = false;
                context.disableDefaultConstraintViolation();
                context.buildConstraintViolationWithTemplate("Citizen name
should between 2 and 30 in length")
                        .addPropertyNode("name").addConstraintViolation();
            }

            if (StringUtils.isBlank(citizen.getCity())) {
```

```
                flag = false;
                context.disableDefaultConstraintViolation();
                context.buildConstraintViolationWithTemplate("Citizen city
required").addPropertyNode("city")
                        .addConstraintViolation();
            }

            if (!(citizen.getCity().length() > 2 && citizen.getCity().length()
< 30)) {
                flag = false;
                context.disableDefaultConstraintViolation();
                context.buildConstraintViolationWithTemplate("Citizen city
should between 2 and 30 in length")
                        .addPropertyNode("city").addConstraintViolation();
            }

            return flag;
        }
    }

    private static class PutRequestValidator extends PostRequestValidator {

        @Override
        public boolean validate(CitizenDto citizen, ConstraintValidatorContext
context) {
            boolean flag = super.validate(citizen, context);

            if (citizen.getDoesCount() > 2 || citizen.getDoesCount() < 0) {
                flag = false;
                context.disableDefaultConstraintViolation();
                context.buildConstraintViolationWithTemplate("Citizen vaccine
count should between 0 or 1 or 2")
                        .addPropertyNode("doesCount").addConstraintViolation()
;

            }

            if (citizen.getCenterId() < 1) {
                flag = false;
                context.disableDefaultConstraintViolation();
                context.buildConstraintViolationWithTemplate("vaccine centerId
should be greater than 0")
                        .addPropertyNode("centerId").addConstraintViolation();

            }

            if (citizen.getId() < 1) {
                flag = false;
```

```
                context.disableDefaultConstraintViolation();
                context.buildConstraintViolationWithTemplate("citizen Id
should be greater than 0")
                        .addPropertyNode("id").addConstraintViolation();
            }
            return flag;
        }
    }
}
```

LoginRequestDtoValidator.java (src/main/java/com.example.vaccinationcenter.validators.impl) :

```
package com.example.vaccinationcenter.validators.impl;

import com.example.vaccinationcenter.dtos.CitizenDto;
import com.example.vaccinationcenter.dtos.LoginRequestDto;
import com.example.vaccinationcenter.validators.antn.CitizenValidator;
import com.example.vaccinationcenter.validators.antn.LoginRequestValidator;
import jakarta.validation.ConstraintValidator;
import jakarta.validation.ConstraintValidatorContext;
import org.apache.commons.lang3.StringUtils;

public class LoginRequestDtoValidator implements
ConstraintValidator<LoginRequestValidator, LoginRequestDto> {

  @Override
  public boolean isValid(LoginRequestDto value, ConstraintValidatorContext
context) {
    boolean flag = true;
    if(StringUtils.isBlank(value.getEmail()))
    {
      flag = false;
      context.disableDefaultConstraintViolation();
      context.buildConstraintViolationWithTemplate("email required")
          .addPropertyNode("email").addConstraintViolation();
    }

    if(!(value.getEmail().length() > 4 && value.getEmail().length() < 30))
    {
      flag = false;
      context.disableDefaultConstraintViolation();
      context.buildConstraintViolationWithTemplate("email length should be
between 4 and 30")
          .addPropertyNode("email").addConstraintViolation();
    }

    if(StringUtils.isBlank(value.getPassword()))
    {
      flag = false;
```

```
      context.disableDefaultConstraintViolation();
      context.buildConstraintViolationWithTemplate("password required")
          .addPropertyNode("password").addConstraintViolation();
    }


    if(!(value.getPassword().length() > 6 && value.getPassword().length() <
10))
    {
      flag = false;
      context.disableDefaultConstraintViolation();
      context.buildConstraintViolationWithTemplate("password length should be
between 6 and 10")
          .addPropertyNode("password").addConstraintViolation();
    }

    return flag;
  }

  @Override
  public void initialize(LoginRequestValidator constraintAnnotation) {
    ConstraintValidator.super.initialize(constraintAnnotation);
  }
}
```

RegistrationRequestDtoValidator.java (src/main/java/com.example.vaccinationcenter.validators.impl) :

```
package com.example.vaccinationcenter.validators.impl;

import com.example.vaccinationcenter.dtos.LoginRequestDto;
import com.example.vaccinationcenter.dtos.RegistrationRequestDto;
import com.example.vaccinationcenter.validators.antn.LoginRequestValidator;
import
com.example.vaccinationcenter.validators.antn.RegistrationRequestValidator;
import jakarta.validation.ConstraintValidator;
import jakarta.validation.ConstraintValidatorContext;
import org.apache.commons.lang3.StringUtils;

public class RegistrationRequestDtoValidator implements
ConstraintValidator<RegistrationRequestValidator, RegistrationRequestDto> {

  @Override
  public boolean isValid(RegistrationRequestDto value,
ConstraintValidatorContext context) {
    boolean flag = true;

    if (StringUtils.isBlank(value.getName())) {
      flag = false;
```

```
      context.disableDefaultConstraintViolation();
      context.buildConstraintViolationWithTemplate("name required")
          .addPropertyNode("name")
          .addConstraintViolation();

    }

    if (!(value.getName().length() > 4 && value.getName().length() < 30)) {
      flag = false;
      context.disableDefaultConstraintViolation();
      context.buildConstraintViolationWithTemplate("name length should be
between 4 and 30 ")
          .addPropertyNode("name")
          .addConstraintViolation();
    }

    LoginRequestDto loginRequestDto = new LoginRequestDto();
    loginRequestDto.setEmail(value.getEmail());
    loginRequestDto.setPassword(value.getPassword());
    ConstraintValidator<LoginRequestValidator, LoginRequestDto> validator =
new LoginRequestDtoValidator();
    flag = validator.isValid(loginRequestDto, context);


    return flag;
  }

  @Override
  public void initialize(RegistrationRequestValidator constraintAnnotation) {
    ConstraintValidator.super.initialize(constraintAnnotation);
  }
}
```

RequestDtoValidator.java (src/main/java/com.example.vaccinationcenter.validators.impl) :

```
package com.example.vaccinationcenter.validators.impl;

import jakarta.validation.ConstraintValidatorContext;

public interface RequestDtoValidator<T> {

  public boolean validate(T requestData, ConstraintValidatorContext context);
}
```

VaccinationCenterDtoValidator.java (src/main/java/com.example.vaccinationcenter.validators.impl) :

```
package com.example.vaccinationcenter.validators.impl;
```

```java
import com.example.vaccinationcenter.dtos.VaccinationCenterDto;
import
com.example.vaccinationcenter.validators.antn.VaccinationCenterValidator;
import jakarta.validation.ConstraintValidator;
import jakarta.validation.ConstraintValidatorContext;
import org.springframework.util.StringUtils;

import java.util.function.Predicate;

public class VaccinationCenterDtoValidator
    implements ConstraintValidator<VaccinationCenterValidator,
VaccinationCenterDto> {

  @Override
  public void initialize(VaccinationCenterValidator constraintAnnotation) {
    // Initialization logic, if needed
  }

  @Override
  public boolean isValid(VaccinationCenterDto dto, ConstraintValidatorContext
context) {

    RequestDtoValidator<VaccinationCenterDto> validator =
getRequestDtoValidator(dto);
    if (validator == null)
      return true;

    return validator.validate(dto, context);

  }


  public static RequestDtoValidator<VaccinationCenterDto>
getRequestDtoValidator(VaccinationCenterDto dto) {
    if (dto == null)
      return null;
    if (dto.getHttpMethod().equalsIgnoreCase("post"))
      return new PostRequestValidator();
    if (dto.getHttpMethod().equalsIgnoreCase("put"))
      return new PutRequestValidator();

    return null;
  }

  private static class PostRequestValidator implements
RequestDtoValidator<VaccinationCenterDto> {

    public boolean validate(VaccinationCenterDto center,
ConstraintValidatorContext context) {
```

```
      boolean flag = true;
      if (center == null) {
        flag = false;
        context.disableDefaultConstraintViolation();
        context.buildConstraintViolationWithTemplate("Center Name Should Not
Be Null").addPropertyNode("center")
            .addConstraintViolation();
        return flag;
      }

      if (!StringUtils.hasText(center.getName())) {
        context.disableDefaultConstraintViolation();
        context.buildConstraintViolationWithTemplate("Center Name
Required").addPropertyNode("name")
            .addConstraintViolation();
        flag = false;
      }

      Predicate<VaccinationCenterDto> validateNameLength =
(VaccinationCenterDto center1) -> {
        return center1.getName().length() > 4 && center1.getName().length() <
60;
      };

      if (validateNameLength.negate().test(center)) {
        context.disableDefaultConstraintViolation();
        context.
            buildConstraintViolationWithTemplate("Center Name Should be
between 4 and 60 characters").addPropertyNode("name")
            .addConstraintViolation();
        flag = false;
      }

      if (!StringUtils.hasText(center.getCity())) {
        context.disableDefaultConstraintViolation();
        context.buildConstraintViolationWithTemplate("Center City
Required").addPropertyNode("city")
            .addConstraintViolation();
        flag = false;
      }

      Predicate<VaccinationCenterDto> validateAddressLength =
(VaccinationCenterDto center1) -> {
        return center1.getCity().length() > 3 && center1.getCity().length() <
100;
      };
```

```
      if (validateAddressLength.negate().test(center)) {
        context.disableDefaultConstraintViolation();
        context.buildConstraintViolationWithTemplate("Center City Should be
between 15 and 100 characters").addPropertyNode("city")
            .addConstraintViolation();
        flag = false;
      }

      return flag;
    }
  }

  private static class PutRequestValidator extends PostRequestValidator {

    @Override
    public boolean validate(VaccinationCenterDto dto,
ConstraintValidatorContext context) {
      boolean flag = super.validate(dto, context);

      if (dto.getId() < 1) {
        flag = false;
        context.disableDefaultConstraintViolation();
        context.buildConstraintViolationWithTemplate("center Id should be
greater than 0")
            .addPropertyNode("id")
            .addConstraintViolation();
      }
      return flag;
    }
  }
}
```

script.js (src/main/resources/static) :

```
console.log("Hello World");

window.onload = function (event) {
  citizenApi.list();
};

let citizenApi = {
  list: citizenList,
  edit: bindCitizenWithCallback,
  view: bindCitizenWithCallback,
  delete: deleteCitizen,
};

function getBaseUrl() {
  let baseUrl = document.getElementById("baseUrl");
```

```
  return baseUrl.value;
}

function logout() {
  localStorage.removeItem("token");
  window.location.href = getBaseUrl() + "/login";
}
function redirectToLogin(response) {
  if (response.status == 302) {
    logout();
  }
}
function getHeaders() {
  console.log(localStorage.getItem("token"));
  let headers = {
    "Content-Type": "application/json",
    Accept: "application/json",
    "x-api": true,
  };
  let authToken = localStorage.getItem("token");
  if (authToken) {
    headers.authorization = authToken;
  }
  return headers;
}

function citizenList() {
  if (document.title !== "Citizens") {
    return;
  }
  display("citizenView", "none");
  display("citizenDetails", "block");
  display("citizenEdit", "none");

  let baseUrl = document.getElementById("baseUrl");
  let url = getBaseUrl() + "/citizens";
  citizenListByUrl(url, _citizenDelete);
}

function citizenListByUrl(url, deleteAction, tableId) {
  let tableRow = {
    citizensTable: function (citizen) {
      return `<td>${citizen.id}</td>
        <td>${citizen.name}</td>
        <td>${citizen.city}</td>
        <td>${citizen.doesCount}</td>
        <td>${citizen.vaccinationStatus}</td>
        <td>${citizen.center?.name}</td>
```

```
        <td>
        <button id="${citizen.id}" onclick="_citizenView(event)">View</button>
        <button id="${citizen.id}" onclick="_citizenEdit(event)">Edit</button>
        <button id="${citizen.id}"
onclick="${deleteAction.name}(event)">Delete</button>
        </td>`;
    },

    centerCitizensTable: function (citizen) {
      return `<td>${citizen.id}</td>
      <td>${citizen.name}</td>
      <td>
      <button id="${citizen.id}" onclick="_citizenView(event)">View</button>
      </td>`;
    },
  };

  let listTableId = "citizensTable";
  if (tableId) {
    let centerId = document.getElementById("centerId").value;
    fetch(getBaseUrl() + "/vaccinationcenter/" + centerId, {
      headers: getHeaders(),
    })
      .then((response) => {
        redirectToLogin(response);
        if (response.ok) {
          return response.json();
        }

        throw new Error("Error while fetching center by id: " + centerId);
      })
      .then((center) => {
        let centerDetails = ` <ul class="list-group">
  <li class="list-group-item">ID: ${center.id}</li>
  <li class="list-group-item">Name: ${center.name}</li>
  <li class="list-group-item">City: ${center.address}</li>
      </ul>`;

        let centerElement = document.getElementById("oneCenter");
        centerElement.innerHTML = "";
        centerElement.innerHTML = centerDetails;
      })
      .catch((error) => {
        console.log(error);
      });
    listTableId = tableId;
  }
  var tableBody = document.getElementById(listTableId).querySelector("tbody");
```

```
  tableBody.innerHTML = "";

  // Fetch citizen data from API
  let baseUrl = document.getElementById("baseUrl");
  console.log(baseUrl.value);
  fetch(url, {
    method: "GET",
    headers: getHeaders(),
  })
    .then(function (response) {
      redirectToLogin(response);

      if (response.ok) {
        return response.json();
      } else {
        throw new Error("Error retrieving citizen data.");
      }
    })
    .then(function (citizens) {
      // Populate the table with citizen data
      let message = "";
      if (citizens.length > 0) {
        message = `<div class="alert alert-warning" role="alert">
        Total ${citizens.length} citizens found.
        </div>`;
        let countRowElement = document.getElementById("countRow");
        countRowElement.innerHTML = message;
      }
      var tableBody = document
        .getElementById(listTableId)
        .querySelector("tbody");
      citizens.forEach(function (citizen) {
        var row = document.createElement("tr");
        row.innerHTML = tableRow[listTableId](citizen);
        tableBody.appendChild(row);
      });
    })
    .catch(function (error) {
      console.error("Error:", error);
    });
}
function bindCitizenWithCallback(id, callback) {
  let baseUrl = document.getElementById("baseUrl");
  console.log(baseUrl.value);
  fetch(baseUrl.value + "/citizens/" + id, {
    method: "GET",
    headers: getHeaders(),
  })
```

```
    .then(function (response) {
      redirectToLogin(response);

      if (response.ok) {
        return response.json();
      } else {
        throw new Error("Error deleting citizen data.");
      }
    })
    .then(function (data) {
      console.log(data);
      callback(data);
    })
    .catch(function (error) {
      console.error("Error:", error);
    });
}

function deleteCitizen(id, callback) {
  let baseUrl = document.getElementById("baseUrl");
  console.log(baseUrl.value);
  fetch(baseUrl.value + "/citizens/" + id, {
    method: "DELETE",
    headers: getHeaders(),
  })
    .then(function (response) {
      redirectToLogin(response);

      if (response.ok) {
        return response.json();
      } else {
        throw new Error("Error deleting citizen data.");
      }
    })
    .then(function (data) {
      console.log(data);
      callback();
    })
    .catch(function (error) {
      console.error("Error:", error);
      citizenApi.list();
    });
}

function _layoutCitizenOnEdit(citizen) {
  let baseUrl = document.getElementById("baseUrl");
  console.log(baseUrl.value);
  fetch(baseUrl.value + "/vaccinationcenter", {
```

```
    method: "GET",
    headers: getHeaders(),
  })
    .then(function (response) {
      redirectToLogin(response);

      if (response.ok) {
        return response.json();
      } else {
        throw new Error("Error deleting citizen data.");
      }
    })
    .then(function (centers) {
      console.log(centers);
      console.log(citizen);
      console.log("design form");
      _layoutEditForm(citizen, centers);
    })
    .catch(function (error) {
      console.error("Error:", error);
    });
}

function _layoutCitizenOnNew() {
  let baseUrl = document.getElementById("baseUrl");
  console.log(baseUrl.value);
  fetch(baseUrl.value + "/vaccinationcenter", {
    method: "GET",
    headers: getHeaders(),
  })
    .then(function (response) {
      redirectToLogin(response);

      if (response.ok) {
        return response.json();
      } else {
        throw new Error("Error deleting citizen data.");
      }
    })
    .then(function (centers) {
      console.log(centers);
      console.log("design form");
      _layoutAppendCenters(centers);
    })
    .catch(function (error) {
      console.error("Error:", error);
    });
}
```

```
function _layoutAppendCenters(centers, citizen) {
  let center = document.getElementById("center");

  let centerOptions = centers;
  centerOptions.forEach(function (option) {
    var optionElement = document.createElement("option");
    optionElement.value = option.id;
    optionElement.text =
      option.id + " | " + option.name + " | " + option.address;
    center.appendChild(optionElement);
  });

  // Select a default option
  let defaultOption = citizen.center?.id; // Set the value of the default
option here
  if (defaultOption) {
    center.value = defaultOption;
  }
}
function _layoutEditForm(citizen, centers) {
  document.getElementById("citizenId").value = citizen.id;
  document.getElementById("citizenName").value = citizen.name;
  document.getElementById("citizenCity").value = citizen.city;
  document.getElementById("vaccinationCount").value = citizen.doesCount;
  document.getElementById("vaccinationStatus").innerHTML =
    citizen.vaccinationStatus;
  _layoutAppendCenters(centers, citizen);
}

function _layoutOneCitizenView(citizen) {
  let citizenElement = document.getElementById("oneCitizen");
  citizenElement.innerHTML = "";
  let listItems = `
    <ul id="oneCitizenView" class="list-group">
<li class="list-group-item">
    <label htmlFor="id">ID: </label>
    <span>${citizen.id}</span>
</li>
<li class="list-group-item">
    <label htmlFor="name">Name: </label>
    <span>${citizen.name}</span>
</li>
<li class="list-group-item">
    <label htmlFor="city">City: </label>
    <span>${citizen.city}</span>
</li>
<li class="list-group-item">
```

```
    <label htmlFor="count">Vaccination Count: </label>
    <span>${citizen.doesCount}</span>
</li>
<li class="list-group-item">
    <label htmlFor="status">Status: </label>
    <span>${citizen.vaccinationStatus}</span>
</li>
<li class="list-group-item">
    <label htmlFor="centerName">Vaccination Center: </label>
    <span>${citizen.center?.name}</span>
</li>
<li class="list-group-item">
    <label htmlFor="centerAddress">Vaccination Center Address: </label>
    <span>${citizen.center?.address}</span>
</li></ul>
`;
  citizenElement.innerHTML = listItems;
}

function _citizenEdit(event) {
  display("citizenView", "none");
  display("citizenDetails", "none");
  display("citizenEdit", "block");
  display("citizenOnEdit", "block");

  let id = event.target.id;
  citizenApi.edit(id, _layoutCitizenOnEdit);
}

function _citizenView(event) {
  console.log(event.target.id);

  let id = event.target.id;
  // citizenApi.view(id, _layoutOneCitizenView);
  let url = getBaseUrl() + "/citizens/" + id;
  window.location.href = url;
}

function _citizenDelete(event) {
  event.preventDefault();
  console.log(event.target.id);
  let id = event.target.id;
  citizenApi.delete(id, citizenApi.list);
}

function display(id, displayValue) {
  document.getElementById(id).style.display = displayValue;
}
```

```
/** vaccination center */

function centerFormAdd(event) {
  event.preventDefault(); // Prevent form submission

  let errorListElement = document.getElementById("errorList");
  errorListElement.innerHTML = "";
  errorListElement.className = "";
  // Get form data
  const centerName = document.getElementById("centerName").value;
  const centerCitySelect = document.getElementById("centerCity");
  const centerCity =
    centerCitySelect.options[centerCitySelect.selectedIndex].text;

  // Create payload object
  const payload = {
    name: centerName,
    city: centerCity,
    httpMethod: "POST",
  };

  let methodName = "POST";
  let centerId = document.getElementById("centerId").value;
  if (centerId) {
    methodName = "PUT";
    payload.httpMethod = "PUT";
    payload.id = centerId;
  }

  // Make a POST request to the vaccinationcenter endpoint
  fetch(getBaseUrl() + "/vaccinationcenter", {
    method: methodName,
    headers: getHeaders(),
    body: JSON.stringify(payload),
  })
    .then((response) => {
      redirectToLogin(response);

      if (response.ok) {
        // Successful response, handle it accordingly
        return response.json();
      } else if (response.status === 400) {
        // Bad Request, handle errors
        return response.json().then((data) => {
          if (Array.isArray(data.errors)) {
            // Display error messages as warnings
            let errorLabels = "";
            data.errors.forEach((error) => {
```

```
              errorLabels += `<div class="alert alert-warning"
role="alert">${error.message}</div>`;
            });

            let errorListElement = document.getElementById("errorList");
            errorListElement.innerHTML = errorLabels;

            document.getElementById("centerForm").appendChild(errorListElement
);
          } else {
            console.warn("Warning: Bad Request");
          }
          throw new Error("Bad Request");
        });
      } else {
        // Handle other response status codes
        throw new Error(`Request failed with status ${response.status}`);
      }
    })
    .then((data) => {
      console.log("Response:", data);
      // Handle the response as needed
      window.location.href = getBaseUrl() + "/vaccinationcenter";
    })
    .catch((error) => {
      console.error("Error:", error);
      // Handle errors
    });
}

function vaccinationCenters() {
  var tableBody =
document.getElementById("centerTable").querySelector("tbody");
  tableBody.innerHTML = "";

  // Fetch citizen data from API
  let baseUrl = document.getElementById("baseUrl");
  console.log(baseUrl.value);
  fetch(getBaseUrl() + "/vaccinationcenter", {
    method: "GET",
    headers: getHeaders(),
  })
    .then(function (response) {
      redirectToLogin(response);

      if (response.ok) {
        return response.json();
      } else {
```

```
        throw new Error("Error retrieving citizen data.");
      }
    })
    .then(function (centers) {
      // Populate the table with citizen data
      let message = "";
      if (centers.length > 0) {
        message = `<div class="alert alert-warning" role="alert">
        Total ${centers.length} vaccination center found.
        </div>`;
        let countRowElement = document.getElementById("countRow");
        countRowElement.innerHTML = message;
      }
      var tableBody = document
        .getElementById("centerTable")
        .querySelector("tbody");
      centers.forEach(function (center) {
        var row = document.createElement("tr");
        row.innerHTML = `
                    <td>${center.id}</td>
                    <td>${center.name}</td>
                    <td>${center.address}</td>
                    <td>
                    <button id="${center.id}"
onclick="_centerView(event)">View</button>
                    <button id="${center.id}"
onclick="_centerEdit(event)">Edit</button>
                    <button id="${center.id}"
onclick="_centerDelete(event)">Delete</button>
                    </td>
                `;
        tableBody.appendChild(row);
      });
    })
    .catch(function (error) {
      console.error("Error:", error);
    });
}

function _centerView(event) {
  let id = event.target.id;
  window.location.href = getBaseUrl() + "/vaccinationcenter/" + id;
}

function _centerEdit(event) {
  display("centers", "none");
  display("centerForm", "block");
  let id = event.target.id;
```

```
  fetch(getBaseUrl() + "/vaccinationcenter/" + id, {
    headers: getHeaders(),
  })
    .then((response) => {
      redirectToLogin(response);

      if (response.ok) {
        return response.json();
      }
      throw new Error("Error while getting center: " + id);
    })
    .then((data) => {
      bindDataWithEditForm(data);
    })
    .catch((error) => {
      console.log(error);
    });
}

function bindDataWithEditForm(center) {
  // Get the select element by its ID
  var select = document.getElementById("centerCity");
  let city = center.address;
  let selectedCityValue;
  // Loop through the options
  for (var i = 0; i < select.options.length; i++) {
    var option = select.options[i];

    // Access the value and text of each option
    var value = option.value;
    var text = option.text;

    if (city === text) {
      selectedCityValue = value;
      break;
    }
    // Do something with the option
    console.log("Option value: " + value);
    console.log("Option text: " + text);
  }

  if (selectedCityValue) {
    select.value = selectedCityValue;
  }
  document.getElementById("centerName").value = center.name;
  document.getElementById("centerId").value = center.id;
}
function _centerDelete(event) {
```

```
  let id = event.target.id;

  fetch(getBaseUrl() + "/vaccinationcenter/" + id, {
    method: "DELETE",
    headers: getHeaders(),
  })
    .then((response) => {
      redirectToLogin(response);

      if (response.ok) {
        return response.json();
      } else {
        throw new Error("Error while deleting vaccinationcenter: " + id);
      }
    })
    .then((data) => {
      vaccinationCenters();
    })
    .catch((error) => {
      console.log(error);
    });
}

function saveCitizen(event) {
  event.preventDefault(); // Prevent form submission

  let errorListElement = document.getElementById("errorList");
  errorListElement.innerHTML = "";
  errorListElement.className = "";
  // Get form data
  const citizenId = document.getElementById("citizenId").value;
  const citizenName = document.getElementById("citizenName").value;
  const citizenCity = document.getElementById("citizenCity").value;

  const vaccinationCountSelect = document.getElementById("vaccinationCount");
  const vaccinationCountSelectValue =
    vaccinationCountSelect.options[vaccinationCountSelect.selectedIndex].value
;

  const vaccinationCenterSelect = document.getElementById("center");
  const centerId = vaccinationCenterSelect.value;

  // Create payload object
  let methodName = "POST";

  const payload = {
    name: citizenName,
    city: citizenCity,
```

```
    doesCount: 0,
    httpMethod: methodName,
    centerId: centerId,
    doesCount: vaccinationCountSelectValue,
  };

  if (citizenId) {
    methodName = "PUT";
    payload.httpMethod = "PUT";
    payload.id = citizenId;
  }

  // Make a POST request to the vaccinationcenter endpoint
  fetch(getBaseUrl() + "/citizens", {
    method: methodName,
    headers: getHeaders(),
    body: JSON.stringify(payload),
  })
    .then((response) => {
      redirectToLogin(response);

      if (response.ok) {
        // Successful response, handle it accordingly
        return response.json();
      } else if (response.status === 400) {
        // Bad Request, handle errors
        return response.json().then((data) => {
          if (Array.isArray(data.errors)) {
            // Display error messages as warnings

            let errorLabels = "";
            data.errors.forEach((error) => {
              errorLabels += `<div class="alert alert-warning"
role="alert">${error.message}</div>`;
            });

            let errorListElement = document.getElementById("errorList");
            errorListElement.innerHTML = errorLabels;

            //
document.getElementById("centerForm").appendChild(errorListElement);
          } else {
            console.warn("Warning: Bad Request");
          }
          throw new Error("Bad Request");
        });
      } else {
        // Handle other response status codes
```

```
        throw new Error(`Request failed with status ${response.status}`);
      }
    })
    .then((data) => {
      console.log("Response:", data);
      let messageElement = document.getElementById("errorList");
      messageElement.innerHTML = `
      <div class="alert alert-success" role="alert">
  Citizen Saved
</div>
      `;
      setTimeout(() => {
        let messageElement = document.getElementById("errorList");
        messageElement.innerHTML = "";
      }, 2000);
      // Handle the response as needed
    })
    .catch((error) => {
      console.error("Error:", error);
      // Handle errors
    });
}

function _newCenter(event) {
  display("centers", "none");
  display("centerForm", "block");
}

function _newCitizen(event) {
  display("citizenDetails", "none");
  display("citizenEdit", "block");
  display("citizenOnEdit", "none");
  _layoutCitizenOnNew();
}

// registration form

function submitForm() {
  const form = document.getElementById("registrationForm");
  const name = form.elements["name"].value;
  const email = form.elements["email"].value;
  const password = form.elements["password"].value;

  // Prepare the form data
  const payload = {
    name: name,
    email: email,
    password: password,
```

```
  };

  // Submit the form data using Fetch API
  fetch("/registration", {
    method: "POST",
    body: JSON.stringify(payload),
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
    },
  })
    .then((response) => {
      if (response.ok) {
        onActionMessage(
          `<div class="alert alert-success" role="alert">Registration
Success.</div>`
        );
        return response.json();
      }
      console.log(response);
      if (response.status == 400) {
        // Bad Request, handle errors
        return response.json().then((data) => {
          if (Array.isArray(data.errors)) {
            let errorLabels = "";
            data.errors.forEach((error) => {
              errorLabels += `<div class="alert alert-warning"
role="alert">${error.message}</div>`;
            });
            onActionMessage(errorLabels);
          }
          throw new Error("invalid data for registration: ");
        });
      } else {
        onActionMessage(
          `<div class="alert alert-warning" role="alert">Registration Failed.
Try again!!</div>`
        );
      }
    })
    .then((data) => {
      console.log(data);
    })
    .catch((error) => {
      // Handle any errors
      console.error(error);
      // alert('Registration Failed');
    });
```

```
}

function onActionMessage(content) {
  let errorMessage = content;
  let messageListElement = document.getElementById("messageList");
  messageListElement.innerHTML = errorMessage;

  setTimeout(() => {
    messageListElement.innerHTML = "";
  }, 2000);
}
```

application.properties (src/main/resources) :

```
spring.datasource.url=jdbc:mysql://localhost:3306/vaccination
spring.datasource.username=root
spring.datasource.password=soumyaranjan@261412
spring.jpa.hibernate.ddl-auto=update
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

VaccinationCenterApplicationTests.java (src/test/java/com.example.vaccinationcenter) :

```
package com.example.vaccinationcenter;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class VaccinationCenterApplicationTests {

    @Test
    void contextLoads() {
    }

}
```

base.jsp (src/main/webapp/WEB-INF/views) :

```
<%
String baseUrl = request.getScheme() + "://" + request.getServerName() +
":" + request.getServerPort() + request.getContextPath();
%>

<br>
<script src="<%= baseUrl %>/script.js"></script>

<input type="hidden" id="baseUrl" value="<%= baseUrl %>"/>
```

citizen_center.jsp (src/main/webapp/WEB-INF/views) :

```
<!DOCTYPE html>
<html>

<head>
    <title>Citizens</title>
```

```
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css">
</head>

<body>
    <div class="container">

        <%@ include file="header.jsp" %>
            <div id="citizenEdit">
                <h2>Citizen Form</h2>

                <form id="citizenEditForm">
                    <input type="hidden" id="citizenId">
                    <div class="form-group">
                        <label for="name">Name:</label>
                        <input type="text" class="form-control"
id="citizenName" required>
                    </div>
                    <div class="form-group">
                        <label for="city">City:</label>
                        <input type="text" class="form-control"
id="citizenCity" required>
                    </div>
                    <div class="form-group">
                        <label for="count">Vaccination Count:</label>
                        <select class="form-control" id="vaccinationCount"
required>
                            <option value="0">None</option>
                            <option value="1">1</option>
                            <option value="2">2</option>
                        </select>
                    </div>
                    <div class="form-group">
                        <label for="status">Status:</label>
                        <label class="form-control" id="vaccinationStatus"
required>
                    </div>
                    <div class="form-group">
                        <label for="center">Vaccination Centers:</label>
                        <select class="form-control" id="center" required>
                            <option value="">Select Center</option>
                        </select>
                    </div>
                    <button type="submit" onclick="saveCitizen(event)"
class="btn btn-primary">Submit</button>

                    <br>
                    <span id="errorList"></span>

                </form>
            </div>
            <div id="citizenView">
                <h2> View Citizen</h2>
                <div id="oneCitizen"></div>
            </div>

            <div id="citizenDetails">
            <button type="button" onclick="_newCitizen(event)" class="btn
btn-primary">Add New Citizen</button>
```

```
                <h1>Center Details</h1>
                <div id="oneCenter">
                </div>
                <input type="hidden" id="centerId" value="${id}">
                <h2>Citizen Details</h2>
                <table id="centerCitizensTable" class="table">
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th>Name</th>
                            <th>Action</th>

                        </tr>
                    </thead>
                    <tbody></tbody>
                </table>

                <div id="countRow"></div>
            </div>
    </div>

</body>

<script>
    window.onload = function (event) {
        display("citizenView", "none");
        display("citizenDetails", "block");
        display("citizenEdit", "none");
        let id = document.getElementById("centerId").value;
        console.log(id);
        citizenListByUrl(getBaseUrl() + "/citizens/center/" + id,
deleteCallback,'centerCitizensTable');
    }

    function deleteCallback(event) {
        citizenApi.delete(event.target.id, redirectCallback);
    }

    function redirectCallback() {
        let id = document.getElementById("centerId").value;
        window.location.href = getBaseUrl() + "/citizens/center/" + value
    }
</script>

</html>
```

citizens.jsp (src/main/webapp/WEB-INF/views) :

```
<!DOCTYPE html>
<html>

<head>
<title>Citizens</title>

<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstra
p.min.css">
</head>

<body>
      <div class="container">
```

```
            <%@ include file="header.jsp"%>
            <div id="citizenEdit">
                  <h2>Citizen From</h2>


                  <form id="citizenEditForm">
                        <input type="hidden" id="citizenId">
                        <div class="form-group">
                              <label for="name">Name:</label> <input
type="text"
                                    class="form-control" id="citizenName"
required>
                        </div>

                        <div class="form-group">
                              <label for="citizenCity">Citizen
City:</label> <select
                                    class="form-control" id="citizenCity"
required>
                                    <option value="None">None</option>
                                    <option value="Mumbai">Mumbai</option>
                                    <option value="Jaipur">Jaipur</option>
                                    <option
value="Bengaluru">Bengaluru</option>
                                    <option
value="Chennai">Chennai</option>
                                    <option
value="Kolkata">Kolkata</option>
                                    <option
value="Visakhapatnam">Visakhapatnam</option>
                                    <option value="Pune">Pune</option>
                                    <option
value="Bhubaneswar">Bhubaneswar</option>
                                    <option value="New Delhi">New
Delhi</option>
                                    <option value="Noida">Noida</option>
                                    <option
value="Gurugram">Gurugram</option>
                                    <option
value="Chandigarh">Chandigarh</option>
                                    <option
value="Srinagar">Srinagar</option>
                                    <option
value="Hyderabad">Hyderabad</option>
                                    <!-- Add more city options as needed --
>
                              </select>

                        </div>

                        <div class="form-group">
                              <label for="center">Vaccination
Centers:</label> <select
                                    class="form-control" id="center"
required>
                              </select>
                        </div>

                        <div id="citizenOnEdit">
```

```
                                    <div class="form-group">
                                          <label for="count">Vaccination
Count:</label> <select
                                                class="form-control"
id="vaccinationCount" required>
                                                <option value="0">None</option>
                                                <option value="1">1</option>
                                                <option value="2">2</option>
                                          </select>
                                    </div>
                                    <div class="form-group">
                                          <label for="status">Status:</label>
<label class="form-control"
                                                id="vaccinationStatus" required>
                                    </div>

                              </div>
                              <button type="submit" onclick="saveCitizen(event)"
                                    class="btn btn-primary">Submit</button>

                              <br> <span id="errorList"></span>

                        </form>

                  </div>
                  <div id="citizenView">
                        <h2>View Citizen</h2>
                        <div id="oneCitizen"></div>
                  </div>
                  <div id="citizenDetails">
                        <button type="button" onclick="_newCitizen(event)"
                              class="btn btn-primary">Add New Citizen</button>

                        <h2>Citizen Details</h2>
                        <table id="citizensTable" class="table">
                              <thead>
                                    <tr>
                                          <th>ID</th>
                                          <th>Name</th>
                                          <th>City</th>
                                          <th>Vaccination Count</th>
                                          <th>Vaccination Status</th>
                                          <th>Vaccination Center</th>
                                          <th>Action</th>

                                    </tr>
                              </thead>
                              <tbody></tbody>
                        </table>

                        <div id="countRow"></div>
                  </div>
            </div>

</body>

</html>
```

header.jsp (src/main/webapp/WEB-INF/views) :

```
<%@ include file="base.jsp" %>
<style>
    /* Custom styles for the dark navbar */
    .navbar-dark {
      background-color: #333; /* Customize the background color */
      color: #fff; /* Customize the text color */
    }
  </style>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item">
            <a class="nav-link" href="<%= baseUrl %>/citizens">Citizens</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="<%= baseUrl
%>/vaccinationcenter">Vaccination Centers</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#" onclick="logout()">Logout</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Welcome, <span
id="userName"></span></a>
          </li>
        </ul>
      </div>
    </div>
  </nav>

  <script>
    function loadUserInfo()
    {
        fetch(getBaseUrl()+"/me",{
            headers: getHeaders()
        })
        .then(response=>{
            if(response.ok)
            {
                return response.json();
            }
            throw new Error('error while accessing user info');
        })
        .then(data=>{
            document.getElementById('userName').innerText= data.name;
        })
        .catch(error=>{
            console.log(error);
        })
    }
    setTimeout(loadUserInfo, 1000);
  </script>
```

login.jsp (src/main/webapp/WEB-INF/views) :

```
<!-- login.jsp -->

<!DOCTYPE html>
<html>
```

```
<head>
    <title>Login Page</title>
    <!-- Include Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css">
</head>

<body>

    <%@ include file="base.jsp" %>
        <div class="container">
            <div class="row justify-content-center mt-5">
                <div class="col-md-4">
                    <h2 class="text-center">Login</h2>
                    <form id="loginForm">
                        <div class="form-group">
                            <label for="email">Email:</label>
                            <input type="email" class="form-control"
id="email" name="email" required>
                        </div>
                        <div class="form-group">
                            <label for="password">Password:</label>
                            <input type="password" class="form-control"
id="password" name="password" required>
                        </div>
                        <button type="submit" class="btn btn-
success">Login</button>

                        <div id="messageList"></div>
                        <p>Register yourself: <a
href="/register">Register</a></p>


                    </form>
                </div>
            </div>
        </div>

        <!-- Include JavaScript -->
        <script>
            document.getElementById("loginForm").addEventListener("submit",
function (event) {
                event.preventDefault(); // Prevent form submission

                const payload = {
                    email: document.getElementById("email").value,
                    password: document.getElementById("password").value,
                }
                // Fetch POST request to /authenticate
                fetch("/authenticate", {
                    method: "POST",
                    body: JSON.stringify(payload),
                    headers: {
                        'Content-Type': 'application/json',
                        'Accept': 'application/json'
                    }
                })
                    .then(function (response) {
                        if (response.ok) {
```

```
                            // Redirect to a success page or perform
further actions

                            return response.json();
                        } else {

                          onLoginError();
                        }
                    })
                    .then(data=>{

                        localStorage.setItem("token",data.encodedToken);
                        window.location.href = getBaseUrl()+
"/vaccinationcenter";
                    })
                    .catch(function (error) {
                        // Handle network or server errors
                        onLoginError();
                        console.error("Error:", error);

                    });
            });

            function onLoginError()
            {
                let errorMessage = `<div class="alert alert-warning"
role="alert">Login Failed. Try again!!</div>`;
                            let messageListElement =
document.getElementById('messageList');
                            messageListElement.innerHTML = errorMessage;

                            setTimeout(()=>{
                                messageListElement.innerHTML ='';
                            }, 2000)
            }
        </script>
</body>

</html>
```

register.jsp (src/main/webapp/WEB-INF/views) :

```
<!DOCTYPE html>
<html>
<head>
  <title>Registration Page</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css
">
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"><
/script>

</head>

<body>

  <%@ include file="base.jsp" %>
    <div class="container">
      <h2>Registration Page</h2>
      <form id="registrationForm">
        <div class="form-group">
```

```
          <label for="name">Name:</label>
          <input type="text" class="form-control" id="name" name="name"
required>
        </div>
        <div class="form-group">
          <label for="email">Email:</label>
          <input type="email" class="form-control" id="email" name="email"
required>
        </div>
        <div class="form-group">
          <label for="password">Password:</label>
          <input type="password" class="form-control" id="password"
name="password" required>
        </div>
        <button type="button" class="btn btn-primary"
onclick="submitForm()">Register</button>

        <div id="messageList"></div>
        <p>Already have an account? <a href="/">Login</a></p>
      </form>
    </div>
</body>

</html>
```

vaccination_center.jsp (src/main/webapp/WEB-INF/views) :

```
<!DOCTYPE html>
<html>

<head>
<title>Vaccination Center Form</title>
<!-- Add Bootstrap CSS -->
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.m
in.css">
</head>

<body>

      <div class="container">
            <%@ include file="header.jsp"%>

            <div id="centers">
                  <button type="button" onclick="_newCenter(event)"
                        class="btn btn-primary">Add New Center</button>

                  <h2>Vaccination Centers</h2>
                  <table id="centerTable" class="table">
                        <thead>
                              <tr>
                                    <th>ID</th>
                                    <th>Name</th>
                                    <th>City</th>
                                    <th>Action</th>
                              </tr>
                        </thead>
                        <tbody></tbody>
                  </table>

                  <div id="countRow"></div>
            </div>
```

```
            <div id="centerForm">
                  <h2 class="text-center">Vaccination Center Form</h2>
                  <form id="centerForm">
                        <input type="hidden" id="centerId">
                        <div class="form-group">
                              <label for="centerName">Center Name:</label>
<input type="text"
                                    class="form-control" id="centerName"
required>
                        </div>
                        <div class="form-group">
                              <label for="centerCity">Center City:</label>
<select
                                    class="form-control" id="centerCity"
required>
                                    <option value="None">None</option>
                                    <option value="Mumbai">Mumbai</option>
                                    <option value="Jaipur">Jaipur</option>
                                    <option
value="Bengaluru">Bengaluru</option>
                                    <option
value="Chennai">Chennai</option>
                                    <option
value="Kolkata">Kolkata</option>
                                    <option
value="Visakhapatnam">Visakhapatnam</option>
                                    <option value="Pune">Pune</option>
                                    <option
value="Bhubaneswar">Bhubaneswar</option>
                                    <option value="New Delhi">New
Delhi</option>
                                    <option value="Noida">Noida</option>
                                    <option
value="Gurugram">Gurugram</option>
                                    <option
value="Chandigarh">Chandigarh</option>
                                    <option
value="Srinagar">Srinagar</option>
                                    <option
value="Hyderabad">Hyderabad</option>
                                    <!-- Add more city options as needed --
>
                              </select>

                        </div>
                        <button type="button"
onclick="centerFormAdd(event)"
                              class="btn btn-primary">Submit</button>
                        <br> <span id="errorList"></span>

                  </form>
            </div>
      </div>

      <!-- Add Bootstrap JS and your custom script -->

      <script>
            // Submit form using JavaScript fetch API
            window.onload = function(event) {
                  display('centerForm', 'none');
```

```
                vaccinationCenters();
            }
        </script>

</body>

</html>
```

view_citizen.jsp (src/main/webapp/WEB-INF/views) :

```
<!DOCTYPE html>
<html>

<head>
    <title>Citizens</title>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css">
</head>

<body>
    <div class="container">
        <%@ include file="header.jsp" %>
        <input type="hidden" id="citizenId" value="${id}">
            <div id="citizenView">
                <h2> View Citizen</h2>
                <div id="oneCitizen"></div>
            </div>
    </div>

    <script>
        window.onload = function(event)
        {
            let id = document.getElementById("citizenId").value;
            console.log(id);
            citizenApi.edit(id, _layoutOneCitizenView)
        }
    </script>
</body>

</html>
```

pom.xml (vaccination-center) :

```
<?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>3.1.0</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>vaccination-center</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>vaccination-center</name>
```

```
    <description>vaccination-center</description>
    <properties>
        <java.version>17</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>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springdoc</groupId>
            <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
            <version>2.0.3</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>

    </dependencies>


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

</project>
```