```java
package stanford.sp14springsecurity;
import lombok.*;
import jakarta.persistence.*;

@Setter
@Getter
@NoArgsConstructor
@ToString
@Entity

class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private String name;
    public Role(String name){
        this.name = name;
    }
}

```

```java
package stanford.sp14springsecurity;
import lombok.*;
import jakarta.persistence.*;
import java.util.Collection;

@Entity
@Setter
@Getter
@NoArgsConstructor
@ToString
@Table(name="user")
class User {
    @Id
    @GeneratedValue
    private long id;
    private String userName;
    private String firstName;
    private String lastName;
    private String email;
    private String phone;
    private String zip;
    private String password;

    @ManyToMany(fetch=FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinTable(name="users_roles", joinColumns = @JoinColumn(name="user_id"), inverseJoinColumns = @JoinColumn(name="role_id"))
    private Collection<Role> roles;

    public User(String userName, String firstName, String lastName, String email, String phone, String zip, String password ) {
        this.userName = userName;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.phone = phone;
        this.zip = zip;
        this.password = password;

```

```
37      }
38 }
39
```

```
 1 package stanford.sp14springsecurity;
 2 import jakarta.persistence.Column;
 3 import jakarta.validation.constraints.Email;
 4 import jakarta.validation.constraints.NotEmpty;
 5 import jakarta.validation.constraints.Pattern;
 6 import lombok.Getter;
 7 import lombok.NoArgsConstructor;
 8 import lombok.Setter;
 9 @Getter
10 @Setter
11 @NoArgsConstructor
12 @FieldMatch.List({
13         @FieldMatch(first="password", second="
   matchingPassword",message="The password fields must
   match")
14 })
15 class UserDTO {
16     @NotEmpty
17     private String userName;
18     @Pattern(regexp = "[A-Za-z]+$", message="Only
   alphabetic allowed")
19     private String firstName;
20     @Pattern(regexp = "[A-Za-z]+$", message="Only
   alphabetic allowed")
21     private String lastName;
22     @Column(unique = true)
23     @Email
24     private String email;
25     private String phone;
26
27     @Pattern(regexp = "[0-9]{5}$", message="Zip code
   wrong format")
28     private String zip;
29
30     @NotEmpty(message="Required")
31     private String password;
32
33     @NotEmpty(message="Required")
34     private String matchingPassword;
35
36     public UserDTO(String userName, String firstName
```

```
36 , String lastName, String email, String phone, String
    zip, String password, String matchingPassword) {
37         this.userName = userName;
38         this.firstName = firstName;
39         this.lastName = lastName;
40         this.email = email;
41         this.phone = phone;
42         this.zip = zip;
43         this.password = password;
44         this.matchingPassword = matchingPassword;
45     }
46 }
47
```

```java
package stanford.sp14springsecurity;
import jakarta.persistence.*;
import jakarta.validation.Constraint;
import jakarta.validation.Payload;
import java.lang.annotation.*;
@Constraint(validatedBy = FieldMatchValidator.class)
@Target({ElementType.TYPE, ElementType.
ANNOTATION_TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface FieldMatch {
    String message() default "";
    Class<?>[] groups() default {};
    Class<? extends Payload> [] payload() default{};

    String first();
    String second();
    @Target({ElementType.TYPE, ElementType.
ANNOTATION_TYPE})
    @Retention(RetentionPolicy.RUNTIME)
    @Documented
    @interface List
    {
        FieldMatch[] value();
    }
}

```

```java
package stanford.sp14springsecurity;

import java.util.List;

public interface RoleService {
    public void saveRole(Role role);
    public Role findRoleByRoleName(String name);
    public List<Role> getAllRoles();
    public List<Role> getRolesByUser(long id);
}

```

```java
package stanford.sp14springsecurity;

import org.springframework.security.core.userdetails.
UserDetailsService;
import org.springframework.security.core.userdetails.
UserDetails;

interface UserService extends UserDetailsService  {
    public UserDetails loadUserByUsername(String
userName);
    public void create(UserDTO userDTO);
    public User findUserByEmail(String email);
    public User findUserByName(String name);

}

```

```java
package stanford.sp14springsecurity;

import org.springframework.security.core.
GrantedAuthority;
import org.springframework.security.core.authority.
SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.
UserDetails;

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

class UserPrincipal implements UserDetails {
    private User user;
    private List<Role> roles;

    public UserPrincipal(User user, List<Role> roles
) {
        super();
        this.user = user;
        this.roles = roles;
    }
    @Override
    public Collection<? extends GrantedAuthority>
getAuthorities(){
        return roles.stream().map(role-> new
SimpleGrantedAuthority(role.getName())).collect(
Collectors.toList());
    }
    @Override
    public String getPassword(){
        return this.user.getPassword();
    }
    @Override
    public String getUsername(){
        return this.user.getEmail();
    }
    @Override
    public boolean isAccountNonExpired(){
        return true;
```

```
35      }
36      @Override
37      public boolean isAccountNonLocked(){
38          return true;
39      }
40      @Override
41      public boolean isCredentialsNonExpired(){
42          return true;
43      }
44
45      @Override
46      public boolean isEnabled(){
47          return true;
48      }
49
50  }
51
```

```java
package stanford.sp14springsecurity;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.List;

interface RoleRepository extends JpaRepository<Role, Long> {
    public Role findRoleByName(String role);
    @Query(value="select * from role where role.id=( select role_id from users_roles where user_id = :id)", nativeQuery=true)
    public List<Role> findRoleByUser(@Param("id") long id);
}

```

```java
package stanford.sp14springsecurity;

import jakarta.validation.Valid;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.propertyeditors.StringTrimmerEditor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.*;

@Controller
@Slf4j
public class UserController {
    @InitBinder
    public void initBinder(WebDataBinder dataBinder) {
        StringTrimmerEditor stringTrimmerEditor = new StringTrimmerEditor(true);
        dataBinder.registerCustomEditor(String.class, stringTrimmerEditor);
    }

    private UserServiceImpl userDetailsService;

    @Autowired
    public UserController(UserServiceImpl userDetailsService) {
        this.userDetailsService = userDetailsService;
    }

    @GetMapping("/")
    private String redirectToLogin() {
        return "redirect:/login";
    }

    @GetMapping("/sign-up")
    public String signUp(Model model) {
```

```
36          model.addAttribute("userDto", new UserDTO());
37          return "sign-up";
38      }
39
40      @PostMapping("/signup-process")
41      public String signupProcess(@Valid @
    ModelAttribute("userDto") UserDTO userDTO,
    BindingResult bindingResult) {
42          if (bindingResult.hasErrors()) {
43              log.warn("Wrong attempt");
44              return "sign-up";
45          }
46          userDetailsService.create(userDTO);
47          return "confirmation";
48      }
49
50      @GetMapping("/login")
51      public String getLoginPage() {
52          log.info("Login page displayed");
53          return "login";
54      }
55
56      @RequestMapping("/home")
57      public String getHome() {
58          log.info("home page displayed");
59          return "home";
60      }
61  }
62
```

```java
package stanford.sp14springsecurity;

import org.springframework.data.jpa.repository.
JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
interface UserRepository extends JpaRepository<User,
Long> {
 public User findUserByEmail(String email);
 public User findUserByUserName(String name);
}

```

```java
package stanford.sp14springsecurity;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
public class RoleServiceImpl implements RoleService {
    private RoleRepository roleRepository;

    @Autowired
    public RoleServiceImpl(RoleRepository roleRepository) {
        this.roleRepository = roleRepository;
    }

    @Override
    @Transactional
    public void saveRole(Role role) {
        roleRepository.save(role);
    }

    @Override
    @Transactional
    public Role findRoleByRoleName(String name) {
        return roleRepository.findRoleByName(name);
    }

    @Override
    public List<Role> getAllRoles() {
        return (List<Role>) roleRepository.findAll();
    }

    @Override
    public List<Role> getRolesByUser(long id) {
        return roleRepository.findRoleByUser(id);
    }
```

```
39 }
40
```

```java
package stanford.sp14springsecurity;

import lombok.extern.slf4j.Slf4j;
import org.modelmapper.ModelMapper;
import org.modelmapper.convention.MatchingStrategies;
import org.springframework.beans.factory.annotation.
Autowired;
import org.springframework.security.authentication.
InternalAuthenticationServiceException;
import org.springframework.security.core.
GrantedAuthority;
import org.springframework.security.core.authority.
SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.
UserDetails;
import org.springframework.security.core.userdetails.
UserDetailsService;
import org.springframework.security.core.userdetails.
UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.
BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.
Transactional;

import javax.sound.midi.SysexMessage;
import java.util.Arrays;
import java.util.Collection;
import java.util.List;
import java.util.stream.Collectors;

@Service
@Slf4j
class UserServiceImpl implements UserService {
    @Autowired
    private UserRepository userRepository;

    @Autowired
    private RoleService roleService;

    @Autowired
```

```
33      private BCryptPasswordEncoder encoder;
34
35      @Override
36      @Transactional
37      public UserDetails loadUserByUsername(String
    userName) throws UsernameNotFoundException {
38          User user = userRepository.findUserByEmail(
    userName);
39          log.debug(userName);
40          if (user == null) {
41              log.warn("Invalid username or password
     {}", userName);
42              throw new UsernameNotFoundException("
    Invalid username or password.");
43
44          }
45          return new UserPrincipal(user, roleService.
    getRolesByUser(user.getId()));
46      }
47
48      private Collection<? extends GrantedAuthority>
    mapRolesToAuthorities(Collection<Role> roles) {
49          return roles.stream()
50                  .map(role -> new
    SimpleGrantedAuthority(role.getName()))
51                  .collect(Collectors.toList());
52      }
53
54      @Transactional
55      public void create(UserDTO userDTO){
56          ModelMapper modelMapper = new ModelMapper();
57          modelMapper.getConfiguration().
    setMatchingStrategy(MatchingStrategies.STRICT);
58          User user = modelMapper.map(userDTO, User.
    class);
59          user.setPassword(encoder.encode(user.
    getPassword()));
60          user.setRoles(Arrays.asList(roleService.
    findRoleByRoleName("ROLE_USER")));
61
62          userRepository.save(user);
```

```
63      }
64
65      public User findUserByEmail(String email){
66          return userRepository.findUserByEmail(email
    );
67      }
68
69      public User findUserByName(String name){
70          return userRepository.findUserByUserName(
    name);
71      }
72  }
73
```

```java
package stanford.sp14springsecurity;

import jakarta.validation.ConstraintValidator;
import jakarta.validation.ConstraintValidatorContext;
import org.springframework.beans.BeanWrapperImpl;
import jakarta.persistence.*;


public class FieldMatchValidator implements
ConstraintValidator<FieldMatch, Object> {
    private String firstFieldName;
    private String secondFieldName;
    private String message;

    @Override
    public void initialize(final FieldMatch
constraintAnnotation) {
        firstFieldName = constraintAnnotation.first
();
        secondFieldName = constraintAnnotation.second
();
        message = constraintAnnotation.message();
    }

    @Override
    public boolean isValid(final Object value, final
ConstraintValidatorContext context) {
        boolean valid = true;
        try {
            final Object firstObj = new
BeanWrapperImpl(value).getPropertyValue(
firstFieldName);
            final Object secondObj = new
BeanWrapperImpl(value).getPropertyValue(
secondFieldName);
            valid = firstObj == null && secondObj ==
null || firstObj != null && firstObj.equals(secondObj
);
        } catch (final Exception ignore) {

        }
```

```
31          if (!valid) {
32              context.
   buildConstraintViolationWithTemplate(message)
33                      .addPropertyNode(firstFieldName)
34                      .addConstraintViolation()
35                      .
   disableDefaultConstraintViolation();
36          }
37          return valid;
38      }
39  }
40
```

```java
package stanford.sp14springsecurity;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

@Configuration
@EnableWebSecurity
public class SecurityConfiguration {

    @Autowired
    private UserServiceImpl userDetailsService;

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
        auth.setUserDetailsService(userDetailsService);
        auth.setPasswordEncoder(passwordEncoder());
        return auth;
    }

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder(11);
```

```
31      }
32
33      @Bean
34      protected SecurityFilterChain filterChain(
    HttpSecurity http) throws Exception {
35          http.authorizeHttpRequests(auth -> auth.
    requestMatchers("/", "/login", "/css/*", "/js/*", "/
    sign-up", "/signup-process")
36                              .permitAll()
37                              .requestMatchers("/home")
38                              .hasAnyRole("USER", "ADMIN")
39                              .anyRequest()
40                              .authenticated())
41                  .formLogin(form -> form.loginPage("/
    login")
42                              .loginProcessingUrl("/login")
43                              .successForwardUrl("/home")
44                              .permitAll())
45                  .logout(logout -> logout.
    invalidateHttpSession(true)
46                              .clearAuthentication(true)
47                              .logoutRequestMatcher(new
    AntPathRequestMatcher("/logout")).permitAll());
48          return http.build();
49      }
50  }
51
```

```java
package stanford.sp14springsecurity;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.
SpringBootApplication;

@SpringBootApplication
public class Sp14SpringSecurityApplication {

    public static void main(String[] args) {
        SpringApplication.run(
Sp14SpringSecurityApplication.class, args);
    }

}

```