Tools Girinka App Documentation

1. Tools and technologies used

- Intellij
- Spring Boot
- Spring Framework
- Maven
- Postgres
- Java 17
- Spring Data JPA (Hibernate)
- Thymeleaf
- Spring Security
- Lombok

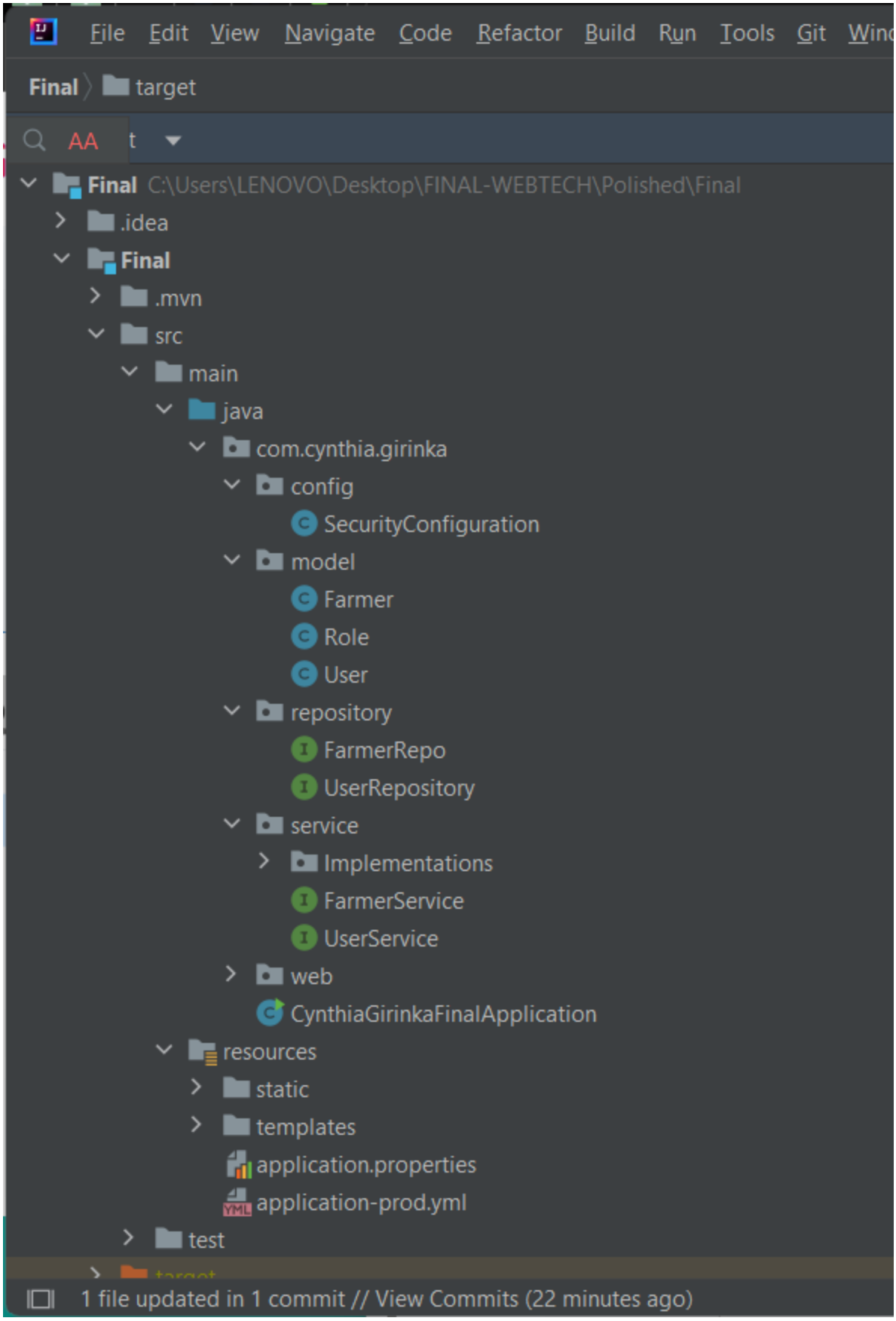
2. Maven Dependencies

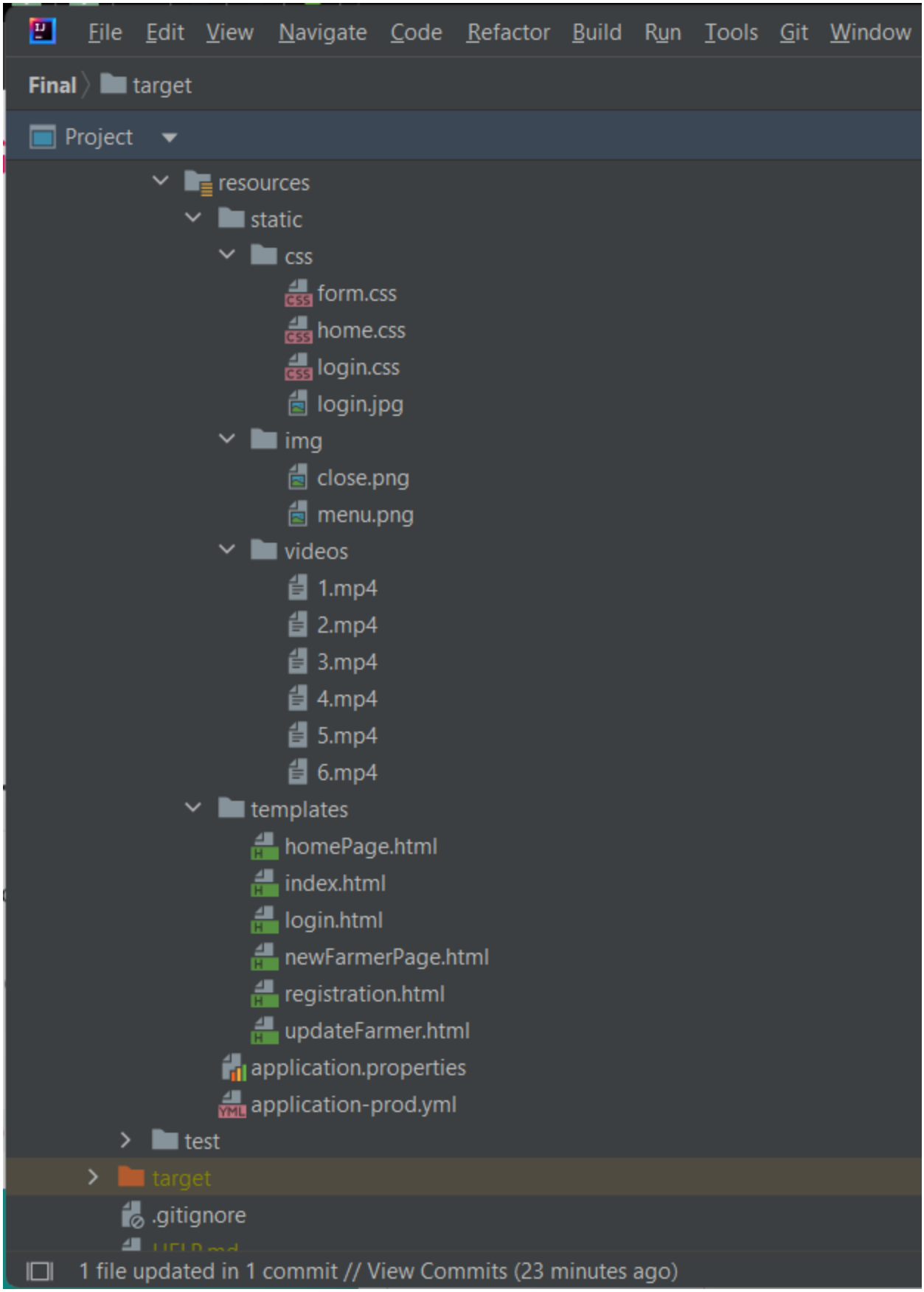
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>2.7.11</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>com.cynthia</groupId>
   <artifactId>Final</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>Cynthia-GirinkaFINAL</name>
   <description>Demo project for Spring Boot</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-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.thymeleaf.extras</groupId>
        <artifactId>thymeleaf-extras-springsecurity5</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
```

```
      <plugins>
         <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
               <excludes>
                  <exclude>
                     <groupId>org.projectlombok</groupId>
                     <artifactId>lombok</artifactId>
                  </exclude>
               </excludes>
            </configuration>
         </plugin>
      </plugins>
   </build>

</project>
```

3. Project Structure


File Edit View Navigate Code Refactor Build Run Tools Git Winc
Final target
AA t
Final C:\Users\LENOVO\Desktop\FINAL-WEBTECH\Polished\Final
.idea
Final
.mvn
src
main
java
com.cynthia.girinka
config
SecurityConfiguration
model
Farmer
Role
User
repository
FarmerRepo
UserRepository
service
Implementations
FarmerService
UserService
web
CynthiaGirinkaFinalApplication
resources
static
templates
application.properties
application-prod.yml
test
target
1 file updated in 1 commit // View Commits (22 minutes ago)


File Edit View Navigate Code Refactor Build Run Tools Git Window
Final target
Project
resources
static
css
form.css
home.css
login.css
login.jpg
img
close.png
menu.png
videos
1.mp4
2.mp4
3.mp4
4.mp4
5.mp4
6.mp4
templates
homePage.html
index.html
login.html
newFarmerPage.html
registration.html
updateFarmer.html
application.properties
application-prod.yml
test
target
.gitignore
1 file updated in 1 commit // View Commits (23 minutes ago)

4. Configure and Setup PostgreSQL Database

Open `application.properties` and add following Postgres database configuration:

```
# DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
spring.datasource.url=jdbc:postgresql://localhost:5432/girinka1
spring.datasource.username=postgres
spring.datasource.password=cynthia

# Hibernate

# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.PostgreSQLDialect

# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto = update

spring.jpa.show-sql=true
spring.main.allow-circular-references=true
```

5. Model Layer - Create JPA Entity - Farmer.java

```
package com.cynthia.girinka.model;

import lombok.Data;

import javax.persistence.*;

@Entity
@Data
@Table(name = "farmers")
public class Farmer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "first_name")
    private String firstName;
    @Column(name = "last_name")
```

```
    private String lastName;
    @Column(name = "email")
    private String email;
}
```

Role.java

```
package com.cynthia.girinka.model;

import javax.persistence.*;

@Entity
@Table(name = "role")
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;


    public Role(){

    }
    public Role(String name) {
        this.name = name;
    }

    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;
    }
```

```
}
```

User.java

```
package com.cynthia.girinka.model;

import javax.persistence.*;
import java.util.Collection;
@Entity
@Table(name = "_user", uniqueConstraints = @UniqueConstraint(columnNames =
"email"))
public class User {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   @Column(name = "first_name")
   private String firstName;
   @Column(name = "last_name")
   private String lastName;
   private String email;
   private String password;
   @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL) //relation
to Role table
   @JoinTable(
           name = "users_roles",
           joinColumns = @JoinColumn(
                   name = "user_id", referencedColumnName = "id"))
   private Collection<Role> roles;




   public User(){

   }
   public User(String firstName, String lastName, String email, String
password, Collection<Role> roles) {
       this.firstName = firstName;
       this.lastName = lastName;
       this.email = email;
       this.password = password;
       this.roles = roles;
   }

   public Long getId() {
```

```
        return id;
    }

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

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    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;
    }

    public Collection<Role> getRoles() {
        return roles;
    }

    public void setRoles(Collection<Role> roles) {
        this.roles = roles;
    }
}
```

6. Repository Layer - FarmerRepository.java

```java
package com.cynthia.girinka.repository;

import com.cynthia.girinka.model.Farmer;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface FarmerRepo extends JpaRepository<Farmer, Long> {
}
```

UserRepository.java

```java
package com.cynthia.girinka.repository;

import com.cynthia.girinka.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends JpaRepository<User, Long> {
   User findByEmail(String email); //will retrieve a user object by email
}
```

7. Service Layer

7.1 FarmerService.java interface

```java
package com.cynthia.girinka.service;

import com.cynthia.girinka.model.Farmer;
import org.springframework.data.domain.Page;

import java.util.List;

public interface FarmerService {
   List<Farmer> getAllFarmers(); //display (READ)
   void saveFarmer(Farmer farmer); //save (CREATE)
   Farmer getFarmerById(Long id); //update by ID (UPDATE)
```

```
    void deleteFarmerById(Long id);
    Page<Farmer> findPaginated(int pageNo, int pageSize, String sortField,
String sortDirection); //pagination and Sorting

}
```

7.1.1 FarmerServiceImpl.java class

```
package com.cynthia.girinka.service.Implementations;

import com.cynthia.girinka.model.Farmer;
import com.cynthia.girinka.repository.FarmerRepo;
import com.cynthia.girinka.service.FarmerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class FarmerServiceImpl implements FarmerService{

    @Autowired
    private FarmerRepo farmerRepo;
    @Override
    public List<Farmer> getAllFarmers() {
        return farmerRepo.findAll();
    }

    @Override
    public void saveFarmer(Farmer farmer) {
        this.farmerRepo.save(farmer);
    }

    @Override
    public Farmer getFarmerById(Long id) {
        Optional<Farmer> optional = farmerRepo.findById(id);
        Farmer farmer = null;
        if(optional.isPresent()){
            farmer = optional.get();
        }else {
            throw new RuntimeException("Farmer not found :: " + id);
        }
```

```
        return farmer;
    }

    @Override
    public void deleteFarmerById(Long id) {
        this.farmerRepo.deleteById(id);
    }

    @Override
    public Page<Farmer> findPaginated(int pageNo, int pageSize, String
sortField, String sortDirection) {
        Sort sort = sortDirection.equalsIgnoreCase(Sort.Direction.ASC.name()) ?
Sort.by(sortField).ascending() :
                Sort.by(sortField).descending();
        Pageable pageable = PageRequest.of(pageNo -1, pageSize, sort);
        return this.farmerRepo.findAll(pageable);
    }
}
```

7.2 UserService.java interface

```
package com.cynthia.girinka.service;

import com.cynthia.girinka.model.User;
import com.cynthia.girinka.web.dto.UserRegistrationDto;
import org.springframework.security.core.userdetails.UserDetailsService;

public interface UserService extends UserDetailsService{
    User save(UserRegistrationDto registrationDto);
}
```

7.2.1 UserServiceImpl.java interface

```
package com.cynthia.girinka.service.Implementations;

import com.cynthia.girinka.model.Role;
import com.cynthia.girinka.model.User;
import com.cynthia.girinka.repository.UserRepository;
import com.cynthia.girinka.service.UserService;
import com.cynthia.girinka.web.dto.UserRegistrationDto;
import org.springframework.beans.factory.annotation.Autowired;
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.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

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

@Service
public class UserServiceImpl implements UserService {
   private UserRepository userRepository;

   @Autowired
   private BCryptPasswordEncoder passwordEncoder;

   public UserServiceImpl(UserRepository userRepository) {
       super();
       this.userRepository = userRepository;
   }
   /*@Autowired
   private UserRepository userRepository;*/ //not recommended so use the above

   @Override
   public User save(UserRegistrationDto registrationDto) {
       User user = new User(registrationDto.getFirstName(),
               registrationDto.getLastName(), registrationDto.getEmail(),
               passwordEncoder.encode(registrationDto.getPassword()),
Arrays.asList(new Role("ROLE_USER")));

       return userRepository.save(user);
   }

   @Override
   public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
       User user = userRepository.findByEmail(username);
       if (user == null) {
           throw new UsernameNotFoundException("Invalid username or
password.");
       }
       return new
org.springframework.security.core.userdetails.User(user.getEmail(),
user.getPassword(), mapRolesToAuthorities(user.getRoles()));
   }

   private Collection<? extends GrantedAuthority>
mapRolesToAuthorities(Collection<Role> roles){
```

```
        return roles.stream().map(role -> new
SimpleGrantedAuthority(role.getName())).collect(Collectors.toList());
    }
}
```

8. Controller Layer

8.1 DTO

UserRegistrationDto

```
package com.cynthia.girinka.web.dto;

public class UserRegistrationDto {
    private String firstName;
    private String lastName;
    private String email;
    private String password;

    public UserRegistrationDto() {

    }

    public UserRegistrationDto(String firstName, String lastName, String email,
String password) {
        super();
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
    }

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
```

```
    }
    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;
    }
}
```

8.2

FarmerController

```
package com.cynthia.girinka.web;

import com.cynthia.girinka.model.Farmer;
import com.cynthia.girinka.service.Implementations.FarmerServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Controller
public class FarmerController {
    @Autowired
    private FarmerServiceImpl farmerService;
   @GetMapping("/showFarmerPage")
    public String viewHomePage(Model model){
        return "index"; //REMEBER TO CHANGE INDEX WHEN YOU ADD THE HOMEPAGE*/
      }

    //display list of paginated farmers
    @GetMapping("/getfarmer")
    public String viewFarmerListPage(Model model){
```

```
      /* model.addAttribute("listFarmers", farmerService.getAllFarmers());
       return "index"; //REMEBER TO CHANGE INDEX WHEN YOU ADD THE HOMEPAGE*/
       return findPaginated(1, "firstName", "asc", model); //farmers will be
sorted in ascending order
   }

   //SAVE farmer
       //1. getting the save page
   @GetMapping("/showNewFarmerForm")
   public String showNewFarmerForm(Model model){
       //connecting data from form to db
       Farmer farmer = new Farmer();
       model.addAttribute("farmer", farmer);
       return "newFarmerPage";
   }
       //2. for the save button
   @PostMapping("/saveFarmer")
   public String saveFarmer(@ModelAttribute("farmer") Farmer farmer){
       //save farmer to db
       farmerService.saveFarmer(farmer);
       return "redirect:/getfarmer"; //REMEMBER TO CHANGE REDIRECTION WHEN YOU
CHANGE TO FARMERLISTPAGE!!!!!
   }

   //UPDATE farmer page
   //1. for the update page
   //2. Basically used the save codes too
   @GetMapping("/showUpdateFarmer/{id}")
   public String showUpdateFarmer(@PathVariable(value = "id") Long id, Model
model){
       //get farmer from the service
       Farmer farmer = farmerService.getFarmerById(id);
       //set farmer as a model attribute to pre-populate the form
       model.addAttribute("farmer2", farmer);
       return "updateFarmer";
   }

   //DELETE farmer
   @GetMapping("/showDeleteFarmer/{id}")
   public String showDeleteFarmer(@PathVariable(value = "id") Long id){
   //call delete method from service impl
       this.farmerService.deleteFarmerById(id);
       return "redirect:/getfarmer"; //REMEMBER TO CHANGE THIS AS WELL TO NEW
FARMERLISTPAGE!!!
   }
//Sorting and Pagination
   @GetMapping("/page/{pageNo}")
   public String findPaginated(@PathVariable(value = "pageNo") int pageNo,
                               @RequestParam("sortField") String sortField,
```

```
                                @RequestParam("sortDir") String sortDir,
                                Model model){
                                int pageSize = 5;

        Page<Farmer> page = farmerService.findPaginated(pageNo, pageSize,
sortField, sortDir);
        List<Farmer> listFarmers = page.getContent();
        model.addAttribute("currentPage", pageNo);
        model.addAttribute("totalPages", page.getTotalPages());
        model.addAttribute("totalItems", page.getTotalElements());

        model.addAttribute("sortField", sortField);
        model.addAttribute("sortDir", sortDir);
        model.addAttribute("reverseSortDir", sortDir.equals("asc") ? "desc" :
"asc");

        model.addAttribute("listFarmers", listFarmers);

        return "index"; //REMEMBER TO CHANGE THIS AS WELL TO NEW
FARMERLISTPAGE!!!
    }
}
```

MainController

```
package com.cynthia.girinka.web;
```

```
import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;



@Controller

public class MainController {

   @GetMapping("/login") //just calls login

   public String login() {
```

```
        return "login";

    }

    @GetMapping("/") //just calls index

    public String home() {

        return "homePage";

    }

}
```

UserRegistrationController

```
package com.cynthia.girinka.web;

import com.cynthia.girinka.service.UserService;
import com.cynthia.girinka.web.dto.UserRegistrationDto;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/registration")
public class UserRegistrationController {
    //call the service not using @Autowired but constructor
    private UserService userService;
    public UserRegistrationController(UserService userService){
        super();
        this.userService = userService;
    }

    //SAVE from registration form
```

```
    @ModelAttribute("user") //returns user object from registration page intead
of using the model.addAttribute Model model
    public UserRegistrationDto userRegistrationDto() {
        return new UserRegistrationDto();
    }

    @GetMapping //mapping from register page
    public String showRegistrationForm() {
        return "registration";
    }
    @PostMapping //register button?
    public String
registerUserAccount(@ModelAttribute("user")UserRegistrationDto
registrationDto){
        userService.save(registrationDto);
        return "redirect:/registration?success";
    }


}
```

9. Config

SecurityConfiguration

```
package com.cynthia.girinka.config;

import com.cynthia.girinka.service.UserService;
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.authentication.builders.Authenti
cationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecur
ity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityCon
figurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
```

```
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter{
   @Autowired
   private UserService userService;

   @Bean
   public BCryptPasswordEncoder passwordEncoder() {
       return new BCryptPasswordEncoder();
   }

   @Override
   protected void configure(HttpSecurity http) throws Exception {
       http.authorizeRequests().antMatchers(
                       "/registration**",
                       "/js/**",
                       "/css/**",
                       "/img/**").permitAll()
               .anyRequest().authenticated()
               .and()
               .formLogin()
               .loginPage("/login")
               .permitAll()
               .and()
               .logout()
               .invalidateHttpSession(true)
               .clearAuthentication(true)
               .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
               .logoutSuccessUrl("/login?logout")
               .permitAll();
   }

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

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception
{
       auth.authenticationProvider(authenticationProvider());
   }
}
```

10. View Layer

10.1 homePage.html

Create new `homePage.html` file under "resources/templates" folder and add the following content to it:

```
<!DOCTYPE html>
<html lang="en" dir="ltr" xmlns:th="http://www.w3.org/1999/xhtml"
xmlns:sec="http://www.w3.org/1999/xhtml">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Girinka</title>

 <!-- insert link for pet.css file for looks and everything -->
 <link th:href="@{/css/home.css}" rel="stylesheet" />


 <!-- insert cdn link for fonts for the page -->
 <!-- search for 'font awesome' on cdnjs.com (bootstrap kinda thing)-->
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.14.0/css/all.min.cs
s">

</head>

<body>
   <!-- start of header section -->
   <!-- The class attribute is set to "header", which is a CSS class that can
be used to style this element. -->
   <header>
      <a href="#" class="logo"><i class="fas fa-solid fa-seedling"></i>
Girinka</a>
       <!--a href="#" class="logo"><i class="fas fa-paw"></i> Girinka</a-->
      <div class="menu-btn"></div>
      <div class="navigation">
         <div class="navigation-items">
            <a th:href = "@{/}">Home</a>
            <a th:href="@{/getfarmer}">Farmers List</a>
               <a sec:authorize="isAuthenticated()" th:href = "@{/logout}">Sign
Out</a>
         </div>
      </div>
   </header>
   <!-- end of header section -->
```

```
   <!--
      <div class="icons">
         <div class="fas fa-bars" id="menu-btn"></div>
         <a href="#" class="fas fa-dog"></a>
         <div class="fas fa-user" id="login-btn"></div>
      </div>

   -->
   <section class="home">
       <video class="video-slide active" th:src="@{/videos/1.mp4}" autoplay
muted loop></video>
      <video class="video-slide" th:src="@{/videos/2.mp4}" autoplay muted
loop></video>
       <video class="video-slide" th:src="@{/videos/3.mp4}" autoplay muted
loop></video>
       <video class="video-slide" th:src="@{/videos/4.mp4}" autoplay muted
loop></video>
       <video class="video-slide" th:src="@{/videos/5.mp4}" autoplay muted
loop></video>
       <video class="video-slide" th:src="@{/videos/.mp4}" autoplay muted
loop></video>

       <div class="content active">
         <h1>Home-grown<br><span>solution</span></h1>
         <p>Girinka is a home-grown solution developed by the government of
Rwanda to address food insecurity and poverty reduction in the country. The
word Girinka can be translated to mean "a cow for a family" in the Kinyarwanda
language.</p>
      </div>

       <div class="content">
         <h1>Food Security<br><span>and nutrition</span></h1>
         <p> The program was initiated in 2006 by President Paul Kagame as a
way to provide cows to poor households, with the aim of improving food
security, nutrition, and income.</p>
      </div>

       <div class="content">
         <h1>Girinka<br><span>Pass-on</span></h1>
         <p>Under the Girinka program, the government provides a cow to a poor
family, which can then provide milk for the family's consumption or sale, and
manure to fertilize their crops. The program also encourages the recipients of
cows to pass on the first female calf to another needy family in their
community, thus creating a sustainable chain of cow ownership and helping to
promote social cohesion.</p>
      </div>

       <div class="content">
         <h1>Reducing<br><span>Poverty</span></h1>
```

```
        <p>Girinka has been a huge success in Rwanda, with the program helping
to reduce poverty and malnutrition, while also promoting social cohesion and
community development. By 2020, the program had provided over 350,000 cows to
poor families across the country.</p>
      </div>

       <div class="content">
         <h1>Social<br><span>Cohesion</span></h1>
         <p>Girinka promotes social cohesion and solidarity among Rwandan
communities. The program emphasizes the importance of giving a cow to a
neighbor in need, which creates a culture of generosity and reciprocity. This
strengthens relationships within communities, fosters trust, and promotes a
sense of unity among Rwandans. Girinka is not just a cattle distribution
program; it is a social program that contributes to the overall development and
well-being of Rwandans.</p>
      </div>

       <div class="content">
         <h1>Hope<br><span>Symbol</span></h1>
         <p>Girinka is not only a homegrown solution to poverty, but it is also
a symbol of hope for Rwandans. Through this program, farmers have access to a
cow, which provides milk for their family's consumption and can be sold for
additional income. This enables farmers to improve their standard of living,
meet their basic needs, and invest in their future.</p>
      </div>

      <div class="media-icons">
         <a th:href="@{/viewHomePage}"><i class="fab fa-whatsapp"></i></a>
         <a th:href="@{/viewHomePage}"><i class="fab fa-instagram"></i></a>
         <a th:href="@{/viewHomePage}"><i class="fab fa-twitter"></i></a>
      </div>
       <div class="slider-navigation">
           <div class="nav-btn active"></div>
           <div class="nav-btn"></div>
           <div class="nav-btn"></div>
           <div class="nav-btn"></div>
           <div class="nav-btn"></div>
           <div class="nav-btn"></div>
       </div>
   </section>
   <!-- insert link to javascript source file -->
   <!--<script src="pet.js"></script>-->

   <!-- add javascript codes to make navigation menu responsive -->
   <script type="text/javascript">
       const menuBtn = document.querySelector(".menu-btn");
       const navigation = document.querySelector(".navigation");

       menuBtn.addEventListener("click", () => {
```

```
            menuBtn.classList.toggle("active");
            navigation.classList.toggle("active");
        });

        //javascript for video slider navigation
        const btns = document.querySelectorAll(".nav-btn");
        const slides = document.querySelectorAll(".video-slide");
        const contents = document.querySelectorAll(".content");

        var sliderNav = function(manual){
         btns.forEach((btn) => {
            btn.classList.remove("active");
         });

         slides.forEach((slide) => {
slide.classList.remove("active");
         });

         contents.forEach((content) => {
            content.classList.remove("active");
         });
            btns[manual].classList.add("active");
            slides[manual].classList.add("active");
            contents[manual].classList.add("active");
        }
        btns.forEach((btn, i) => {
            btn.addEventListener("click", () => {
                sliderNav(i);
            });
        });
    </script>
</body>
</html>
```

10.2 index.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml"
xmlns:sec="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <title>Girinka</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknL
PMO" crossorigin="anonymous">
```

```
   <link th:href="@{/css/form.css}" rel="stylesheet" />


   <!-- insert cdn link for fonts for the page -->
   <!-- search for 'font awesome' on cdnjs.com (bootstrap kinda thing)-->
   <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.14.0/css/all.min.cs
s">

</head>
<body>
<header>
   <a href="#" class="logo"><i class="fas fa-solid fa-seedling"></i>
Girinka</a>
   <div class="menu-btn"></div>
   <div class="navigation">
       <div class="navigation-items">
           <a th:href = "@{/}">Home</a>
           <a sec:authorize="isAuthenticated()" th:href = "@{/logout}">Sign
Out</a>
       </div>
   </div>
</header>
<div class="container my-2">
   <br><br>
   <h1>Farmers List</h1>
   <a th:href="@{/showNewFarmerForm}" class="btn btn-primary btn-sm mb-2"> New
Farmer </a>
   <table border="1" class="table table-hover table-light">
       <thead>
       <tr>
           <th>No</th>
           <th>
               <a th:href="@{'/page/' + ${currentPage} +
'?sortField=firstName&sortDir=' + ${reverseSortDir}}">
                   First Name</a>
           </th>
           <th>
               <a th:href="@{'/page/' + ${currentPage} +
'?sortField=lastName&sortDir=' + ${reverseSortDir}}">
                   Last Name</a>
           </th>
           <th>
               <a th:href="@{'/page/' + ${currentPage} +
'?sortField=email&sortDir=' + ${reverseSortDir}}">
                   Email</a>
           </th>
           <th> Actions </th>
       </tr>
```

```
      </thead>
      <tbody>
      <tr  th:each="farmer, counter : ${listFarmers}">
          <td th:text="${counter.count}"></td>
          <td th:text="${farmer.firstName}"></td>
          <td th:text="${farmer.lastName}"></td>
          <td th:text="${farmer.email}"></td>
          <td> <a th:href="@{/showUpdateFarmer/{id}(id=${farmer.id})}"
class="btn btn-primary">Update</a>
              <a th:href="@{/showDeleteFarmer/{id}(id=${farmer.id})}"
class="btn btn-danger">Delete</a>
          </td>
      </tr>
      </tbody>
   </table>

   <div th:if = "${totalPages > 1}">
       <div class = "row col-sm-10">
           <div class = "col-sm-2">
               Total Rows: [[${totalItems}]]
           </div>
           <div class = "col-sm-1">
    <span th:each="i: ${#numbers.sequence(1, totalPages)}">
     <a th:if="${currentPage != i}" th:href="@{'/page/' + ${i}+ '?sortField=' +
${sortField} + '&sortDir=' + ${sortDir}}">[[${i}]]</a>
      <span th:unless="${currentPage != i}">[[${i}]]</span>     
    </span>
           </div>
           <div class = "col-sm-1">
               <a th:if="${currentPage < totalPages}" th:href="@{'/page/' +
${currentPage + 1}+ '?sortField=' + ${sortField} + '&sortDir=' +
${sortDir}}">Next</a>
               <span th:unless="${currentPage < totalPages}">Next</span>
           </div>

           <div class="col-sm-1">
               <a th:if="${currentPage < totalPages}" th:href="@{'/page/' +
${totalPages}+ '?sortField=' + ${sortField} + '&sortDir=' +
${sortDir}}">Last</a>
               <span th:unless="${currentPage < totalPages}">Last</span>
           </div>
       </div>
   </div>
</div>
</body>
</html>
```

10.3 login.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
<head>
   <meta charset="UTF-8">
   <title>Girinka/login</title>
   <link rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"

integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEj
h4u"
         crossorigin="anonymous">
   <link th:href="@{/css/form.css}" rel="stylesheet" />

</head>
<body>

<!-- create navigation bar ( header) -->

  <!--
  <nav>
  <div class="container">
       <div class="navbar-header">
           <button type="button" class="navbar-toggle collapsed"
                   data-toggle="collapse" data-target="#navbar"
aria-expanded="false"
                   aria-controls="navbar">
               <span class="sr-only">Toggle navigation</span> <span
                   class="icon-bar"></span> <span class="icon-bar"></span>
<span
                   class="icon-bar"></span>
           </button>
           <a class="navbar-brand" href="#" th:href="@{/}">Girinka</a>
       </div>
   </div>
   </nav>-->
   <header>
       <a href="#" class="logo"><i class="fas fa-solid fa-seedling"></i>
Girinka</a>
       <div class="menu-btn"></div>
       <div class="navigation">
           <div class="navigation-items">
               <!--<a th:href = "@{/}">Home</a>-->
               <!--<a sec:authorize="isAuthenticated()" th:href =
"@{/logout}">Sign Out</a>-->
           </div>
       </div>
```

```
   </header>


<br>
<br>
<div class = "container">
   <div class = "row">
       <div class = "col-md-6 col-md-offset-3">

           <h1> User Login Page </h1>
           <form th:action="@{/login}" method="post">

               <!-- error message -->
               <div th:if="${param.error}">
                   <div class="alert alert-danger">Invalid username or
                       password.</div>
               </div>

               <!-- logout message -->
               <div th:if="${param.logout}">
                   <div class="alert alert-info">You have been logged
out.</div>
               </div>

               <div class = "form-group">
                   <label for ="username"> Username </label> :
                   <input type="text" class = "form-control" id ="username"
name = "username"
                          placeholder="Enter Email ID" autofocus="autofocus">
               </div>

               <div class="form-group">
                   <label for="password">Password</label>: <input
type="password"
                                                                  id="password"
name="password" class="form-control"

placeholder="Enter Password" />
               </div>

               <div class="form-group">
                   <div class="row">
                       <div class="col-sm-6 col-sm-offset-3">
                           <input type="submit" name="login-submit"
id="login-submit"
                                  class="form-control btn btn-primary"
value="Log In" />
                       </div>
                   </div>
```

```
            </div>
          </form>
          <div class="form-group">
     <span>New user? <a href="/" th:href="@{/registration}">Register
       here</a></span>
          </div>
       </div>
   </div>
</div>
</body>
</html>
```

10.4 newFarmerPage.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
<head>
   <meta charset="UTF-8">
   <title>Girinka/SaveFarmer</title>
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknL
PMO" crossorigin="anonymous">

</head>
<body>
<div class="container">
   <hr>
<h1>Save Farmer</h1>
<form action="#" th:action="@{/saveFarmer}" th:object="${farmer}"
method="post">
   <label>Firstname</label>
   <input type="text" th:field="*{firstName}" required class="form-control mb-4
col-7">
   <label>Lastname</label>
   <input type="text" th:field="*{lastName}" required class="form-control mb-4
col-7">
   <label>Email</label>
   <input type="text" th:field="*{email}" required class="form-control mb-4
col-7">
   <button type="submit" class="btn btn-info col-7"> Save Farmer </button>

</form>
   <hr>
```

```
   <a th:href="@{/getfarmer}"> Farmer List </a> <!-- REMEBER TO CHANGE THIS
LINK TO FARMERLISTFORM AS WELL!!!! -->

</div>
</body>
</html>
```

10.5 registration.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
<head>
   <meta charset="UTF-8">
   <title>Girinka/register</title>
   <link rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"

integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEj
h4u"
         crossorigin="anonymous">
</head>
<body>

<!-- create navigation bar ( header) -->
<nav class="navbar navbar-inverse navbar-fixed-top">
   <div class="container">
       <div class="navbar-header">
           <button type="button" class="navbar-toggle collapsed"
                   data-toggle="collapse" data-target="#navbar"
aria-expanded="false"
                   aria-controls="navbar">
               <span class="sr-only">Toggle navigation</span> <span
                   class="icon-bar"></span> <span class="icon-bar"></span>
<span
                   class="icon-bar"></span>
           </button>
           <a class="navbar-brand" href="#" th:href="@{/}">Registration and
               Login Module</a>
       </div>
   </div>
</nav>

<br>
<br>
```

```
<!-- Create HTML registration form -->
<div class="container">
   <div class="row">
       <div class="col-md-6 col-md-offset-3">

           <!-- success message -->
           <div th:if="${param.success}">
               <div class="alert alert-info">You've successfully registered
                   to our Girinka app!</div>
           </div>

           <h1>Registration</h1>

           <form th:action="@{/registration}" method="post"
th:object="${user}">
               <div class="form-group">
                   <label class="control-label" for="firstName"> First Name
</label>
                   <input id="firstName" class="form-control"
th:field="*{firstName}"
                           required autofocus="autofocus" />
               </div>

               <div class="form-group">
                   <label class="control-label" for="lastName"> Last Name
</label> <input
                       id="lastName" class="form-control"
th:field="*{lastName}"
                       required autofocus="autofocus" />
               </div>

               <div class="form-group">
                   <label class="control-label" for="email"> Email </label>
<input
                       id="email" class="form-control" th:field="*{email}"
required
                       autofocus="autofocus" />
               </div>

               <div class="form-group">
                   <label class="control-label" for="password"> Password
</label> <input
                       id="password" class="form-control" type="password"
                       th:field="*{password}" required autofocus="autofocus" />
               </div>

               <div class="form-group">
                   <button type="submit" class="btn
btn-success">Register</button>
```

```
                    <span>Already registered? <a href="/"
th:href="@{/login}">Login
       here</a></span>
               </div>
           </form>
       </div>
   </div>
</div>
</body>
</html>
```

10.6 updateFarmer.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
<head>
 <meta charset="UTF-8">
 <title>Girinka/UpdateFarmer</title>
 <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknL
PMO" crossorigin="anonymous">

</head>
<body>
<div class="container">
 <hr>
 <h1>Update Farmer</h1>
 <form action="#" th:action="@{/saveFarmer}" th:object="${farmer2}"
method="post">
   <!-- Add hidden form field to handle update -->
   <input type="hidden" th:field="*{id}" />
   <label>Firstname</label>
   <input type="text" th:field="*{firstName}" required class="form-control mb-4
col-7">
   <label>Lastname</label>
   <input type="text" th:field="*{lastName}" required class="form-control mb-4
col-7">
   <label>Email</label>
   <input type="text" th:field="*{email}" required class="form-control mb-4
col-7">
   <button type="submit" class="btn btn-info col-7"> Update Farmer </button>
```

```
 </form>
 <hr>

 <a th:href="@{/}"> Farmer List </a> <!-- REMEBER TO CHANGE THIS LINK "@{/}" TO
FARMERLISTFORM AS WELL!!!! -->

</div>
</body>
</html>
```