Summary of Project

Name: Jianwei Zhang

NUID: 001447259

In this project, we will build a People`s relationship Management System, which implements the function of managing the information of People (Basic Create Read Update Delete). There are different roles who have different authority to do things.

Functionality Performed

Login/Logout: You need to login with authorized user so you could move forward.

Roles: Different Roles has different roles and authority to access resources.

CRUD: You could Create, Read, Update and Delete data from database.

Search: Search object with its property.

Technologies used

Spring MVC + **Hibernate**: for CRUD and Search

Spring Security: For Login/Logout and Roles

CSS + HTML for basic view.

Roles and tasks

Three different kind roles: **ADMIN**, **USER**, **TEACHER**

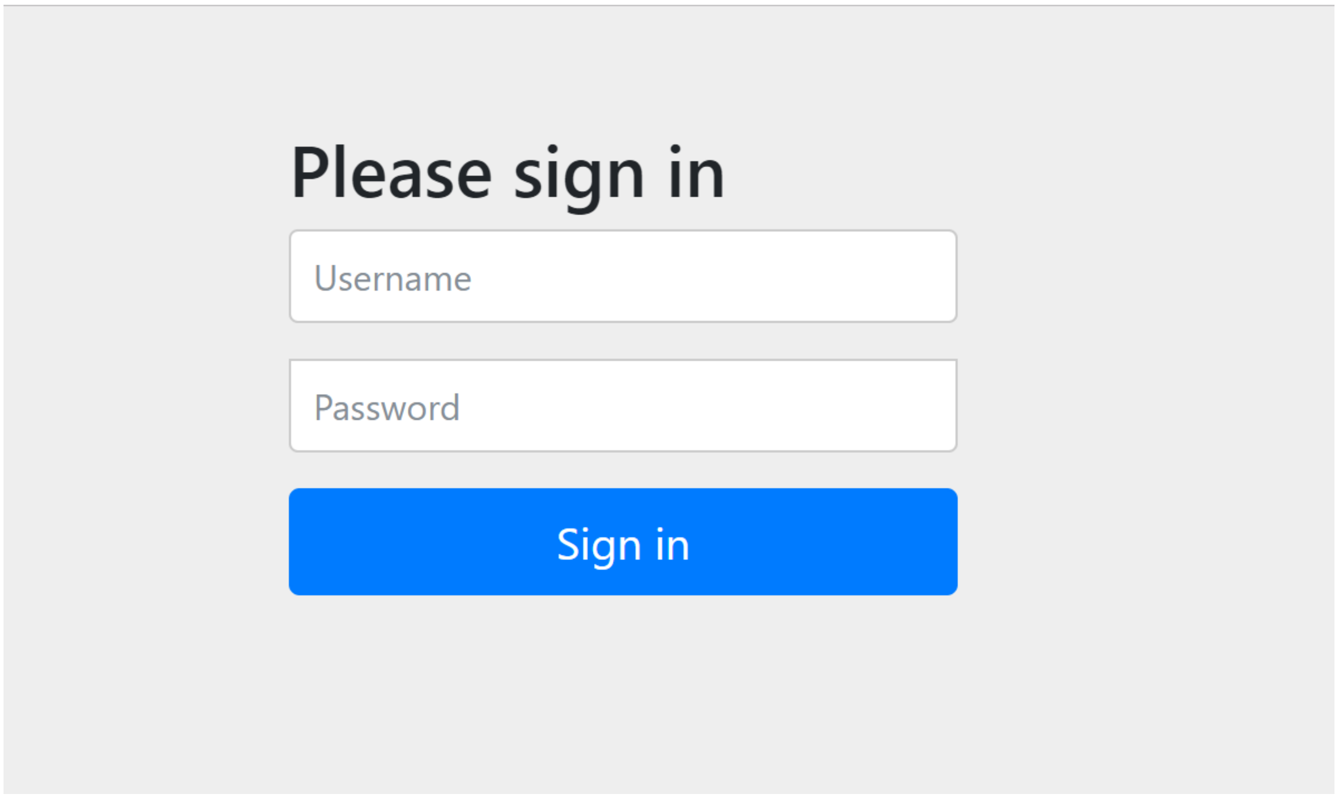
ADMIN could do everything on the project, including CRUD and search.

TEACHER could add information to database.

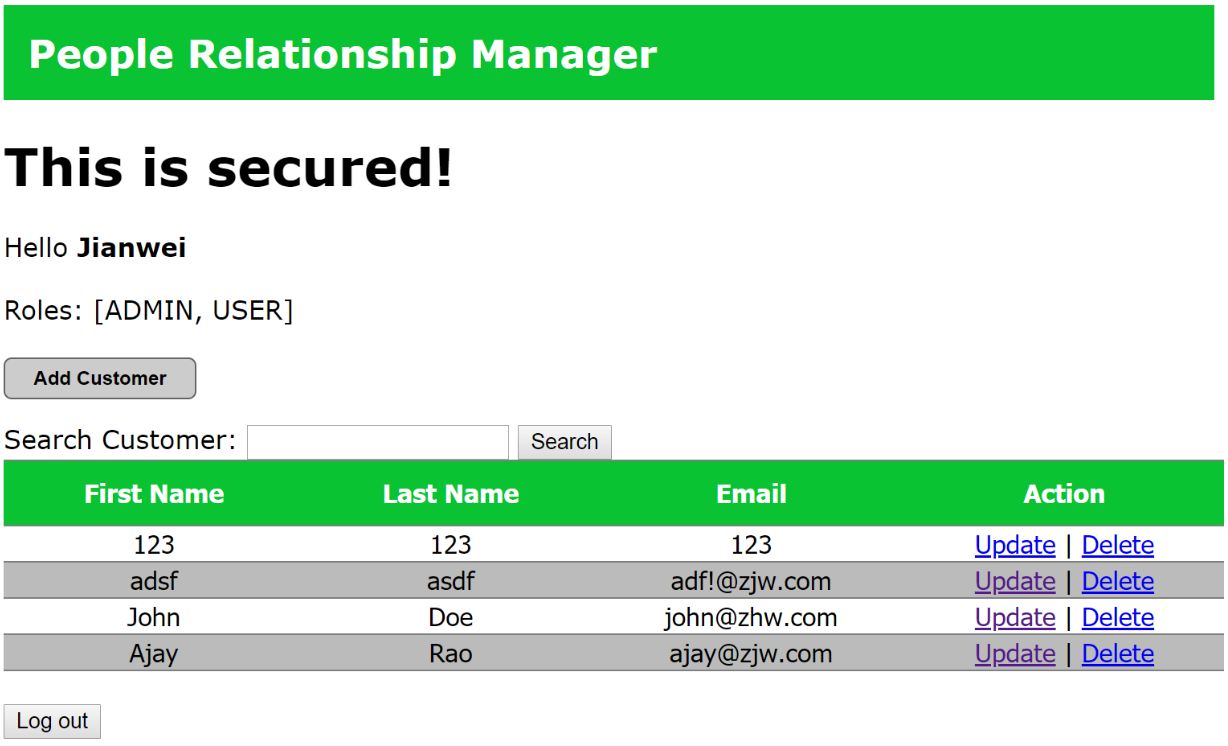
USER could only view data retrieved from database.

Screenshots of key screens

- Login



- Admin Screen



People Relationship Manager

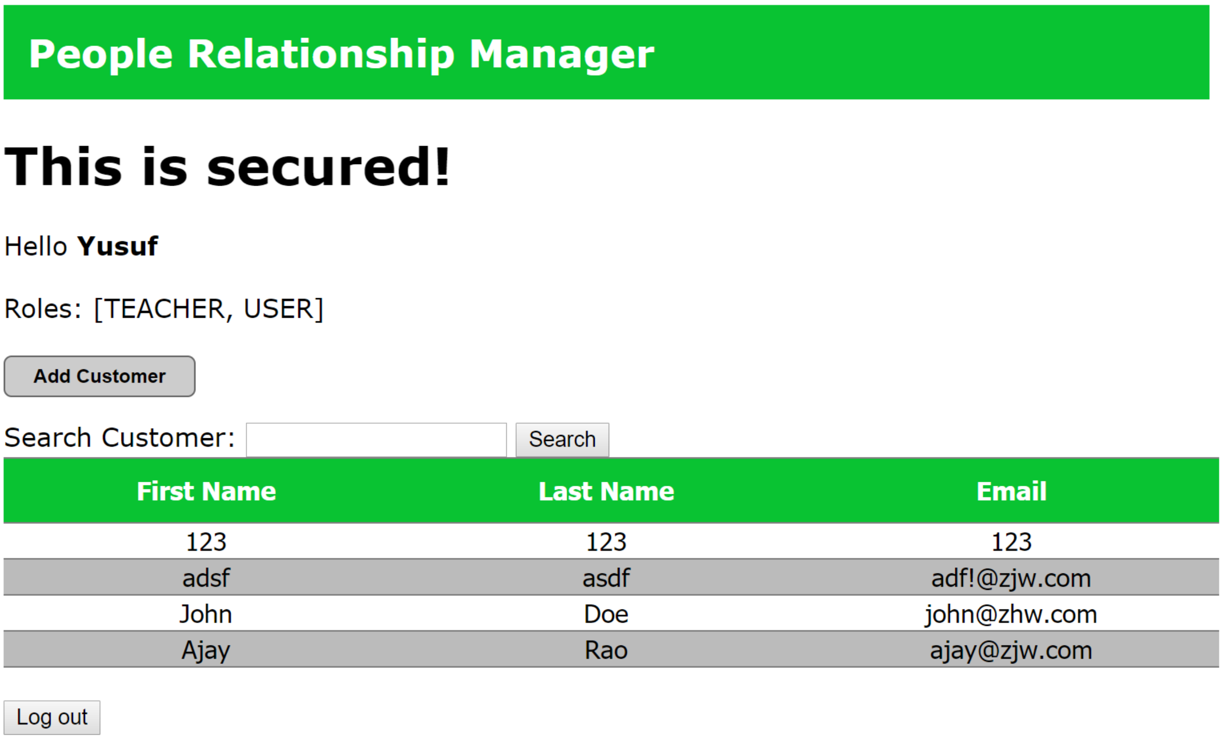
This is secured!

Hello **Jianwei**

Roles: [ADMIN, USER]

First Name	Last Name	Email	Action
123	123	123	Update | Delete
adsf	asdf	adf!@zjw.com	Update | Delete
John	Doe	john@zhw.com	Update | Delete
Ajay	Rao	ajay@zjw.com	Update | Delete

- Teacher Screen



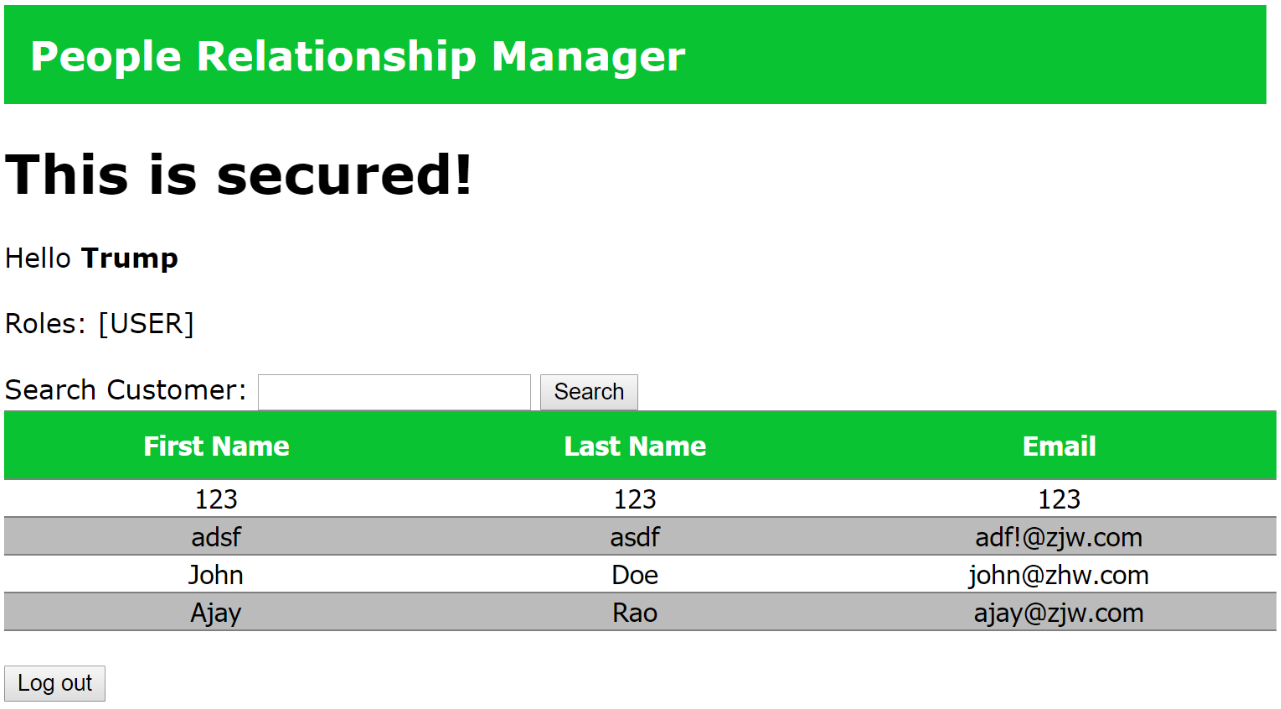
This is secured!

Hello **Yusuf**

Roles: [TEACHER, USER]

First Name	Last Name	Email
123	123	123
adsf	asdf	adf!@zjw.com
John	Doe	john@zhw.com
Ajay	Rao	ajay@zjw.com

- User Screen



This is secured!

Hello **Trump**

Roles: [USER]

First Name	Last Name	Email
123	123	123
adsf	asdf	adf!@zjw.com
John	Doe	john@zhw.com
Ajay	Rao	ajay@zjw.com

APPENDIX

- Controller/View src

```
package com.zjw.controller;
```

```java
import com.zjw.entity.Customer;
import com.zjw.service.CustomerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.List;
@Controller
@RequestMapping("/customer")
public class CustomerController {


    //need to inject our customer service
    @Autowired
    private CustomerService customerService;

    @GetMapping("/list")
    public String listCustomers(Model theModel) {
        System.out.print("entering controller");
        // get customers from the service
        List<Customer> theCustomers = customerService.getCustomers();
        // add the customers to the model
        theModel.addAttribute("customers", theCustomers);
        System.out.println("Jumping...");
        return "list-customers";
    }

    @GetMapping("/showFormForAdd")
    public String showFormForAdd(Model theModel) {
        //create model attribute to bind form data
        Customer theCustomer = new Customer();
        theModel.addAttribute("customer", theCustomer);
        return "customer-form";
    }

    @PostMapping("/saveCustomer")
    public String saveCustomer(@ModelAttribute("customer") Customer theCustomer) {
        //save the customer
        customerService.saveCustomer(theCustomer);
        return "redirect:/customer/list";
    }

    @GetMapping("/showFormForUpdate")
    public String showFormForUpdate(@RequestParam("customerId") int theId,
                                    Model theModel) {
        //get the customer from the database
        Customer theCustomer = customerService.getCustomer(theId);
        //set customer as a model attribute to pre-populate the form
        theModel.addAttribute("customer",theCustomer);
        return "customer-form";
    }
    @GetMapping("/delete")
    public String deleteCustomer(@RequestParam("customerId") int theId){
        //delete the customer from the database
        customerService.deleteCustomer(theId);
        return "redirect:/customer/list";
    }

    @GetMapping("/search")
    public String searchCustomers(@RequestParam("theSearchName") String theSearchName,
Model theModel){
        List<Customer> theCustomers = customerService.searchCustomers(theSearchName);
        theModel.addAttribute("customers", theCustomers);
        return "list-customers";
    }
}
```

- View

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="security" uri="http://www.springframework.org/security/tags" %>
<!DOCTYPE html>
<html>
<head>
    <title>List Customers</title>
    <!-- reference our style sheet -->
    <link type="text/css"
          rel="stylesheet"
 href="${pageContext.request.contextPath}/resources/css/style.css"/>
</head>

<body>
    <div id="wrapper">
        <div id="header">
            <h2>People Relationship Manager</h2>
        </div>
    </div>
    <div id="container">
        <h1>This is secured!</h1>
        <p>
            Hello <b><c:out value="${pageContext.request.remoteUser}"/></b> <br><br>
            Roles: <security:authentication property="principal.authorities"/>
        </p>
        <div id="content">
            <security:authorize access="hasAnyAuthority('TEACHER','ADMIN')">
            <!-- put new buttion: Add Customer -->
            <input type="button" value = "Add Customer"
            onclick="window.location.href='showFormForAdd';return false;"
                class="add-button"
            />
            </security:authorize>
            <%-- add search function--%>
            <form:form action="search" method="get">
                Search Customer: <input type = "text" name = "theSearchName"/>
                <input type = "submit" value="Search" calss="add-button">
            </form:form>
    <!--  add our html table here -->

            <table>
                <tr>
                    <th>First Name</th>
                    <th>Last Name</th>
                    <th>Email</th>
                    <security:authorize access="hasAuthority('ADMIN')">
                    <th>Action</th>
                    </security:authorize>
                </tr>

                <!-- loop over and print our customers -->
                <c:forEach var="tempCustomer" items="${customers}">


                    <!-- Construct an "update link with customer id -->
                    <c:url var = "updateLink" value = "/customer/showFormForUpdate">
                        <c:param name = "customerId" value="${tempCustomer.id}"/>
                    </c:url>

                    <!-- Construct an "delete link with customer id -->
                    <c:url var = "deleteLink" value = "/customer/delete">
                        <c:param name = "customerId" value="${tempCustomer.id}"/>
                    </c:url>

                    <tr>
                        <td> ${tempCustomer.firstName} </td>
                        <td> ${tempCustomer.lastName} </td>
                        <td> ${tempCustomer.email} </td>

                        <security:authorize access="hasAuthority('ADMIN')">
                        <td>
                            <!-- display the update link -->
                            <a href = "${updateLink}">Update</a>
                            |
                            <a href = "${deleteLink}"
                            onclick = "if(!(confirm('Are you sure you want to delete
this'))) return false">Delete</a>
                        </td>
                        </security:authorize>



</tr>
</c:forEach>
</table>
</div>
</div>
    <br>
    <c:url var="logoutUrl" value="/logout"/>
    <form class="form-inline" action="${logoutUrl}" method="post">
        <input type="submit" value="Log out" />
        <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"/>
    </form>
</body>
</html>
```

- POJO

```
@Entity
@Table(name = "customer")
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;
    @Column(name = "first_name")
    private String firstName;
    @Column(name = "last_name")
    private String lastName;
    @Column(name = "email")
    private String email;
}
```

- DAO

```
    @Override
    public List<Customer> getCustomers() {
        //get the current hibernate session
        Session currentSession = sessionFactory.getCurrentSession();
        //create a query ... sort by last name
        Query<Customer> theQuery =
                currentSession.createQuery("from Customer order by lastName",
                        Customer.class);
        //execute query and get result list
        List<Customer> customers = theQuery.getResultList();
        //return the results
        return customers;
    }
    @Override
    public void saveCustomer(Customer theCustomer) {
        //get curreent hibernate session
        Session currentSession = sessionFactory.getCurrentSession();
        //save the customer to the database
        currentSession.saveOrUpdate(theCustomer);

    }
```

- SERVICE

```java
@Service
public class CustomerServiceImpl implements CustomerService{
    //need to inject customerDAO
    @Autowired
    private CustomerDAO customerDAO;
    @Override
    @Transactional
    public List<Customer> getCustomers() {
        return customerDAO.getCustomers();
    }
    @Override
    @Transactional
    public void saveCustomer(Customer theCustomer) {

        customerDAO.saveCustomer(theCustomer);
    }
    @Override
    @Transactional
    public Customer getCustomer(int theId) {
        return customerDAO.getCustomers(theId);
    }
    @Override
    @Transactional
    public void deleteCustomer(int theId) {
        customerDAO.deleteCustomer(theId);
    }
    @Override
    @Transactional
    public List<Customer> searchCustomers(String theSearchName) {
        return customerDAO.searchCustomers(theSearchName);
    }
}
```