# Final Project Document

Ziyao Qiao 001835152 qiao.z@husky.neu.edu

## Summary

In this project, I focus on developing a shopping cart application using Spring + Hibernate framework as the backend development tools. I also applied bootstrap and using JSP and JavaScript as frontend development method. My shopping cart application can perform a whole process of shopping including register, login, add to cart, order review and receive items. Also, my applications contain admin role which is used to maintain the user list, item list and order status.

## Functions

1. Register: this function will require you to input your username, email, password, phone number and address.
2. Login: this function is divided into two parts, regular user login and admin login. The login will use username or email and password as credential.
3. View item: user can click the item to see it's detail
4. Add to cart: by choosing the number of item you need, user can add it temporary into the cart
5. Buy: by clicking the button and confirm, user will send an order
6. Manage User: this is a admin function and admin can delete user and its related orders
7. Manage items: this is a admin function and admin can delete or add new items into system
8. Order status: user can view order status and confirm to receive goods which is under transport status.
9. Handle order: this is a admin function and admin can decide to send the items to user

## Technologies used

1. Using Spring as the basic framework of the server. Spring MVC for dispatcher the request, Spring IOC for dependency injection and Spring Security for protecting the safety of the admin system
2. Using hibernate to connect the database, using MySQL as database.
3. Using JSP, JSTL for frontend logical expression
4. Using bootstrap to beautify the page
5. Using JavaScript to perform varies kind of notifications
6. Used AJAX to communicate with server

## Role

1. User Role:
   - User need to register before using this system
   - User can view all items

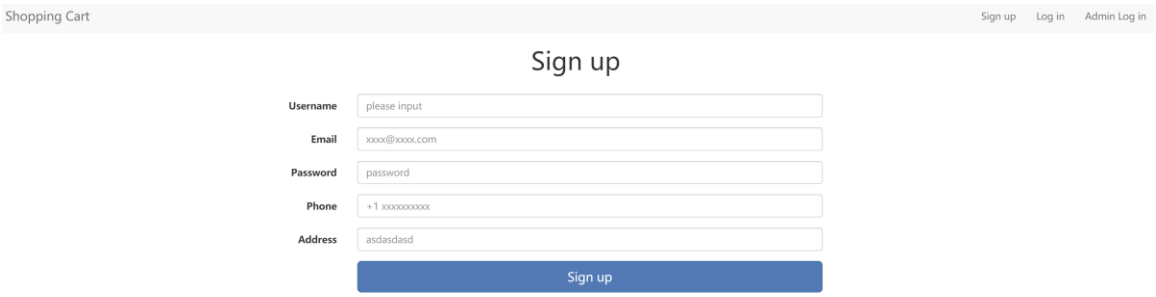
- User can click each item and view its detail
- User can set the number of item he need and add to shopping cart, the cart will save the record even the user logout the system
- User can directly buy an item in the item detail page
- User can review all the orders (sorted according to the order status) in the detail page
- User can manually select receive button to confirm he received the items.

2. Admin Role:
   - Admin can manage user list including delete a user with its related order
   - Admin can manage item list including delete an item or add an item
   - Admin can go to handle orders page to handle pending orders

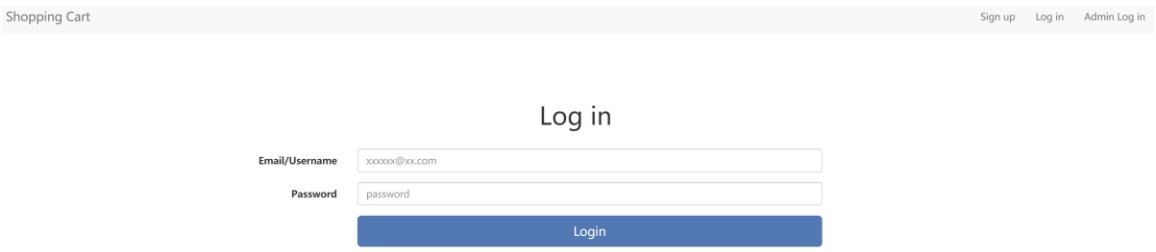
## Screenshots

### Main

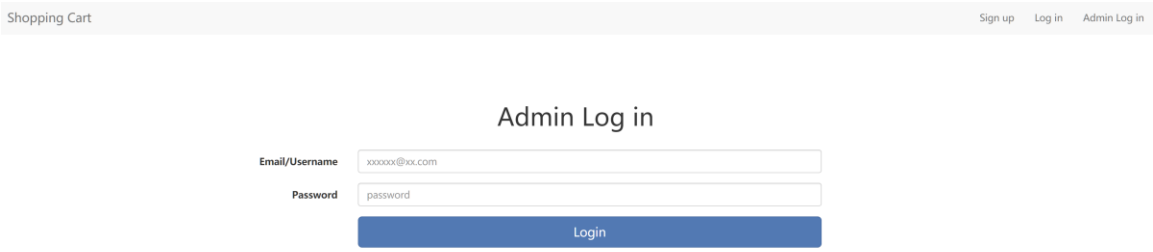

### Sign Up



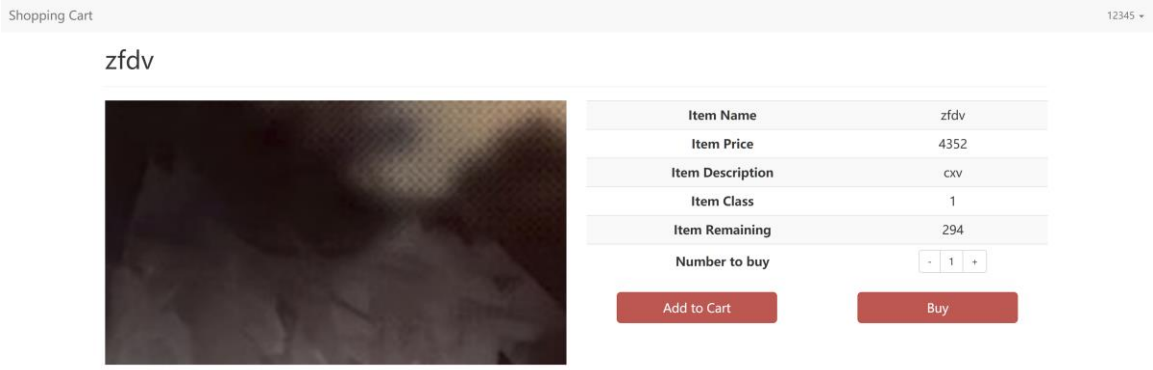

### Log in



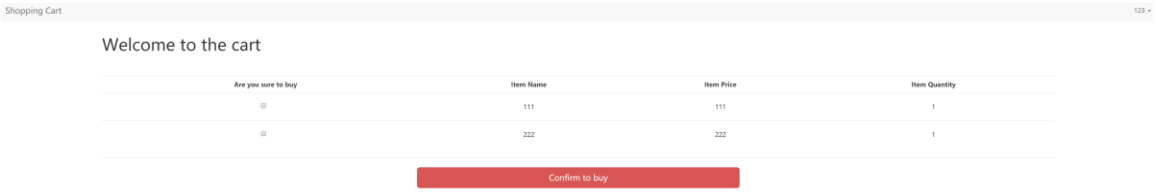
## Admin Log in



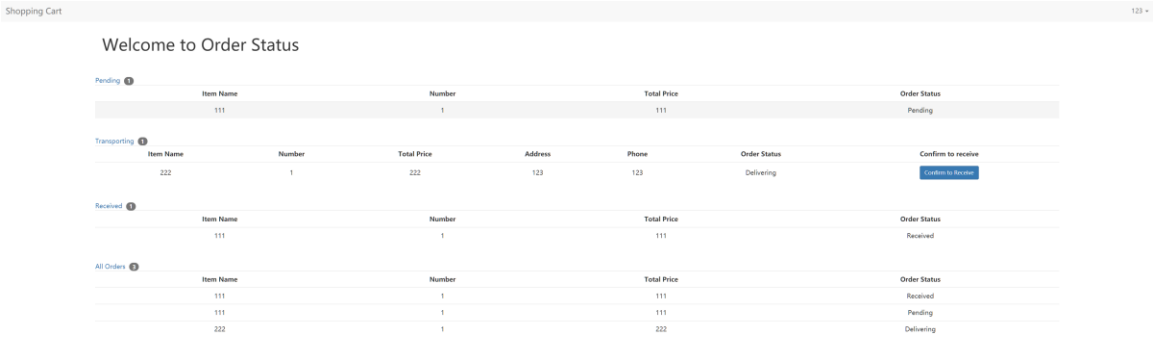

## Item Detail

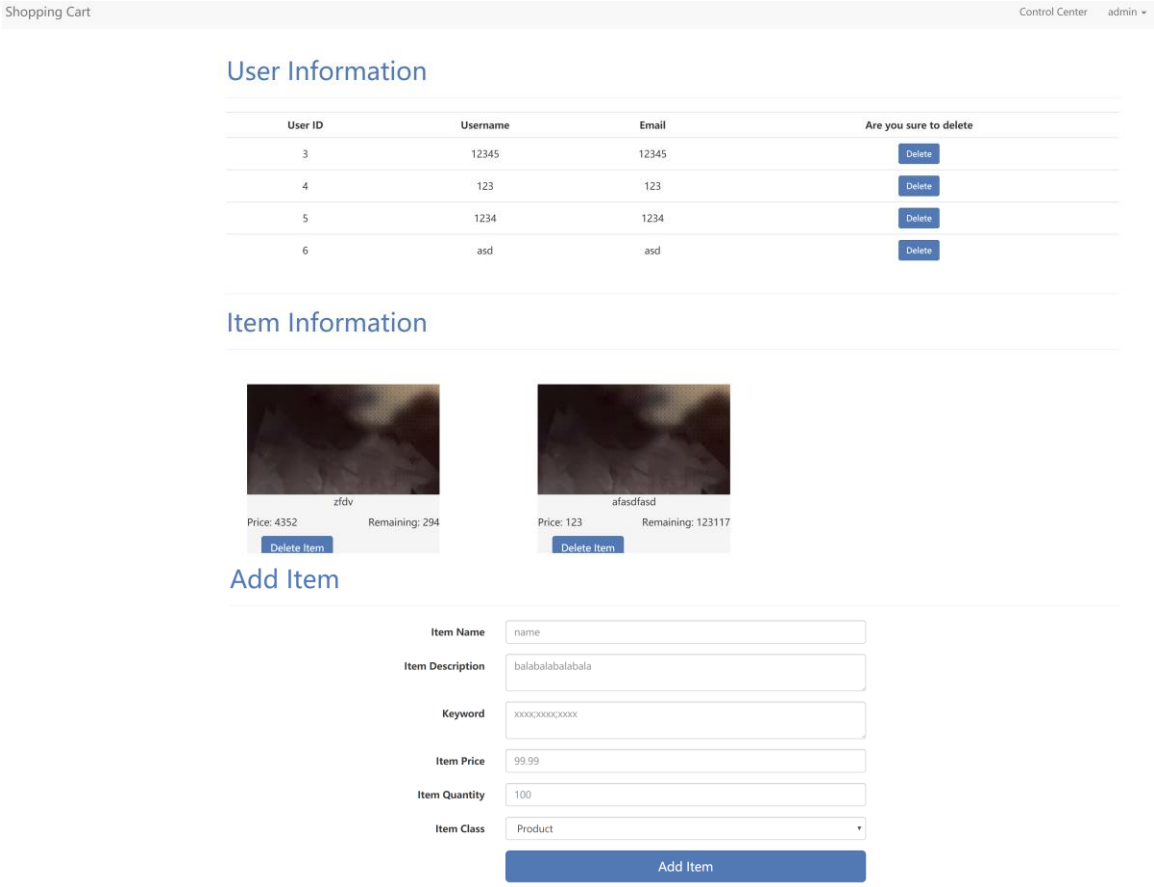



## Shopping Cart

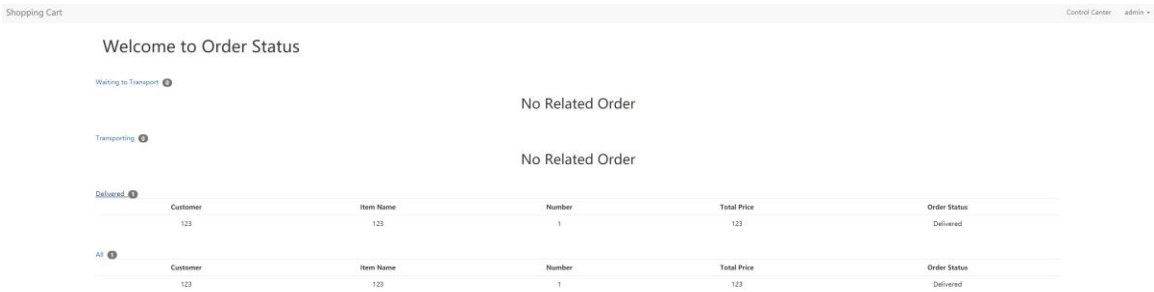



## Order Status






## Handle Order




# Code

## Controller

```
@Controller
public class ProductController {
   @Resource
   private ProductDAOService productDAOService;

   @RequestMapping(value = "/getAllProducts")
   @ResponseBody
   public Map<String,Object> getAllProducts(){
```

```
    List<Product> allProductList;
    allProductList = productDAOService.getAllProduct();
    String allProducts = JSONArray.toJSONString(allProductList);
    Map<String,Object> ajaxResult = new HashMap<>();
    ajaxResult.put("allProducts",allProducts);
    return ajaxResult;
}

@RequestMapping(value = "/deleteProduct")
@ResponseBody
public ServiceResponse deleteProduct(int id, HttpSession session) {
    ServiceResponse serviceResponse = productDAOService.deleteProduct(id);
    session.setAttribute("allProducts", productDAOService.getAllProduct());
    return serviceResponse;
}

@RequestMapping(value = "/addProduct")
@ResponseBody
public Map<String, Object> addProduct(String name,,int productPrice,int productCount) {
    String resultText;
    Product itemTmp = new Product();
    itemTmp.setName(name);
    itemTmp.setPrice(productPrice);
    itemTmp.setproductCount(productCount);
    productDAOService.addProduct(itemTmp);
    resultText = "functionSuccess";
    Map<String,Object> ajaxResult = new HashMap<>();
    ajaxResult.put("resultText",resultText);
    return ajaxResult;
}

@RequestMapping(value = "/productDetail")
@ResponseBody
public Map<String, Object> getProductDetail(int id, HttpSession httpSession) {
    Product itemTmp = productDAOService.getProduct(id);
    httpSession.setAttribute("productDetail",itemTmp);
    Map<String,Object> ajaxResult = new HashMap<>();
    ajaxResult.put("resultText","functionSuccess");
    return ajaxResult;
}

@RequestMapping(value = "/product_detail")
public String product_detail() {
    return "product_detail";
}


@RequestMapping(value = "/getProductById")
@ResponseBody
public Map<String, Object> getProductById(int id) {
    Product itemTmp = productDAOService.getProduct(id);
    String resultText = JSON.toJSONString(itemTmp);
    Map<String,Object> ajaxResult = new HashMap<>();
```

```
        ajaxResult.put("resultText",resultText);
        return ajaxResult;
    }

}

@Controller
public class ShoppingCartController {
    @Resource
    private ProductDAOService productDAOService;
    @Resource
    private ShoppingCartDAOService shoppingCartDAOService;

    @RequestMapping(value = "/shoppingcart")
    public String shoppingcart() {
        return "shoppingcart";
    }

    @RequestMapping(value = "/addShoppingCart", method = RequestMethod.POST)
    @ResponseBody
    public Map<String, Object> addToShoppingCart(int userId, int Id, int productCount) {
        ShoppingCart shoppingCart = shoppingCartDAOService.getShoppingCart(userId, Id);
        if (shoppingCart == null) {
            ShoppingCart shoppingCartTmp = new ShoppingCart();
            shoppingCartTmp.setUserId(userId);
            shoppingCartTmp.setId(Id);
            shoppingCartTmp.setproductCount(productCount);
            int productPrice = productDAOService.getProduct(Id).getPrice() * productCount;
            shoppingCartTmp.setprice(productPrice);
            shoppingCartDAOService.addShoppingCart(shoppingCartTmp);
        } else {
            shoppingCart.setproductCount(shoppingCart.getproductCount() + productCount);
            int productPrice = productDAOService.getProduct(Id).getPrice() * shoppingCart.getproductCount();
            shoppingCart.setprice(productPrice);
            shoppingCartDAOService.updateShoppingCart(shoppingCart);
        }
        Map<String, Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText", "functionSuccess");
        return ajaxResult;
    }

    @RequestMapping(value = "/getShoppingCart")
    @ResponseBody
    public Map<String, Object> getShoppingCart(int userId) {
        List<ShoppingCart> shoppingCartList = shoppingCartDAOService.getShoppingCart(userId);
        String shoppingCart = JSONArray.toJSONString(shoppingCartList);
        Map<String, Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText", shoppingCart);
        return ajaxResult;
    }

    @RequestMapping(value = "/deleteShoppingCart")
    @ResponseBody
```

```
    public Map<String, Object> deleteShoppingCart(int userId, int Id) {
        shoppingCartDAOService.deleteShoppingCart(userId, Id);
        Map<String, Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText", "functionSuccess");
        return ajaxResult;
    }
}

@Controller
public class ShoppingRecordController {
    @Resource
    private ProductDAOService productDAOService;
    @Resource
    private ShoppingRecordService shoppingRecordService;

    @RequestMapping(value = "/shoppingrecord")
    public String shoppingrecord(){
        return "shoppingrecord";
    }

    @RequestMapping(value = "/shoppinghandler")
    public String shoppinghandler(){
        return "shoppinghandler";
    }

    @RequestMapping(value = "/addShoppingRecord")
    @ResponseBody
    public Map<String,Object> addShoppingRecord(int userId,int Id,int productCount){
        String resultText = null;
        Product itemTmp = productDAOService.getProduct(Id);
        if(productCount<=itemTmp.getproductCount()) {
            ShoppingRecord shoppingRecord = new ShoppingRecord();
            shoppingRecord.setUserId(userId);
            shoppingRecord.setId(Id);
            int productPrice = itemTmp.getPrice() * productCount;
            shoppingRecord.setprice(productPrice);
            shoppingRecord.setproductCount(productCount);
            shoppingRecord.setOrderStatus(0);
            Date date = new Date();
            SimpleDateFormat dateFormat = new SimpleDateFormat("HH-mm-ss MM-dd");
            shoppingRecord.setTime(dateFormat.format(date));

            itemTmp.setproductCount(itemTmp.getproductCount()-productCount);
            productDAOService.updateProduct(itemTmp);
            shoppingRecordService.addShoppingRecord(shoppingRecord);
            resultText = "functionSuccess";
        }
        else{
            resultText = "unEnough";
        }
        Map<String,Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText",resultText);
        return ajaxResult;
```

```
    }

    @RequestMapping(value = "/updateShoppingRecord")
    @ResponseBody
    public Map<String,Object> updateShoppingRecord(int userId,int Id,String orderTime, int orderStatus){
        ShoppingRecord shoppingRecord = shoppingRecordService.getShoppingRecord(userId,Id,orderTime);
        shoppingRecord.setOrderStatus(orderStatus);
        shoppingRecordService.updateShoppingRecord(shoppingRecord);
        Map<String,Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText","functionSuccess");
        return ajaxResult;
    }

    @RequestMapping(value = "/getShoppingRecords")
    @ResponseBody
    public Map<String,Object> getShoppingRecords(int userId){
        List<ShoppingRecord> shoppingRecordList = shoppingRecordService.getShoppingRecords(userId);
        String shoppingRecords = JSONArray.toJSONString(shoppingRecordList);
        Map<String,Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText",shoppingRecords);
        return ajaxResult;
    }

    @RequestMapping(value = "/getRecordsByStatus")
    @ResponseBody
    public Map<String,Object> getRecordsByStatus(int orderStatus){
        List<ShoppingRecord> shoppingRecordList =
shoppingRecordService.getRecordsByStatus(orderStatus);
        String shoppingRecords = JSONArray.toJSONString(shoppingRecordList);
        Map<String,Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText",shoppingRecords);
        return ajaxResult;
    }

    @RequestMapping(value = "/getAllShoppingRecords")
    @ResponseBody
    public Map<String,Object> getAllShoppingRecords(){
        List<ShoppingRecord> shoppingRecordList = shoppingRecordService.getAllShoppingRecords();
        String shoppingRecords = JSONArray.toJSONString(shoppingRecordList);
        Map<String,Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText",shoppingRecords);
        return ajaxResult;
    }

    @RequestMapping(value = "/getProductRecord")
    @ResponseBody
    public Map<String,Object> getProductRecord(int userId,int Id){
        String resultText = "false";
        if(shoppingRecordService.getProductRecord(userId,Id)){
            resultText = "true";
        }
        Map<String,Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText",resultText);
```

```
        return ajaxResult;
    }
}


@Controller
public class UserController {

    @Resource
    UserDAOService userDAOService;

    @Resource
    ProductDAOService productDAOService;

    @Resource
    UserDetailDAOService userDetailDAOService;


    @RequestMapping(value = "/register")
    public String register() {
        return "register";
    }

    @RequestMapping(value = "/login")
    public String login() {
        return "login";
    }

    @RequestMapping(value = "/admin_login")
    public String adminLogin() {
        return "admin_login";
    }

    @RequestMapping(value = "/main")
    public String main(HttpSession session) {
        session.setAttribute("allUser", userDAOService.getAllUser());
        session.setAttribute("allProduct", productDAOService.getAllProduct());
        return "main";
    }

    @RequestMapping(value = "/control")
    public String control(HttpSession session) {
        User user = new User();
        user.setName("admin");
        user.setRole(1);
        UserDetail userDetail = new UserDetail();
        session.setAttribute("currentUser", user);
        session.setAttribute("currentUserDetail", userDetail);
        session.setAttribute("allUser", userDAOService.getAllUser());
        session.setAttribute("allProduct", productDAOService.getAllProduct());
        return "control";
    }
```

```
    @RequestMapping(value = "/Login", method = RequestMethod.POST)
    @ResponseBody
    public Map<String, Object> Login(String userNameOrEmail, String password, HttpSession httpSession) {
        String resultText = "fail";
        User user = userDAOService.get(userNameOrEmail);
        if (user == null)
            resultText = "unexist";
        else {
            UserDetail userDetail = userDetailDAOService.getDetail(user.getId());
            if (userDetail.getPassword().equals(password) && user.getRole() == 0) {
                resultText = "functionSuccess";
                httpSession.setAttribute("currentUser", user);
                httpSession.setAttribute("currentUserDetail", userDetail);
            } else
                resultText = "wrong";
        }
        Map<String, Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText", resultText);
        return ajaxResult;
    }

    @RequestMapping(value = "/register", method = RequestMethod.POST)
    @ResponseBody
    public Map<String, Object> register(String userName, String userEmail, String password, String
phoneNumber, String address) {
        String resultText = "fail";
        Map<String, Object> ajaxResult = new HashMap<String, Object>();
        if (isValid(userName) && isValid(userEmail) && isValid(password) && isValid(phoneNumber) &&
isValid(address)) {
            User user = userDAOService.get(userName);
            if (user != null) {
                resultText = "nameExist";
            } else {
                user = userDAOService.get(userEmail);
                if (user != null)
                    resultText = "emailExist";
                else {
                    User userTmp = new User();
                    userTmp.setName(userName);
                    userTmp.setEmail(userEmail);
                    userTmp.setRole(0);
                    userDAOService.addUser(userTmp);

                    UserDetail userDetail = new UserDetail();
                    userDetail.setId(userTmp.getId());
                    userDetail.setAddress(address);
                    userDetail.setPassword(password);
                    userDetail.setPhoneNumber(phoneNumber);

                    userDetailDAOService.addUserDetail(userDetail);
                    resultText = "functionSuccess";
                }
            }
```

```
        }
        ajaxResult.put("resultText", resultText);
        return ajaxResult;
    }

    @RequestMapping(value = "/getAllUser")
    @ResponseBody
    public Map<String, Object> getAllUser() {
        List<User> userList;
        userList = userDAOService.getAllUser();
        String allUsers = JSONArray.toJSONString(userList);
        Map<String, Object> ajaxResult = new HashMap<>();
        ajaxResult.put("allUsers", allUsers);
        return ajaxResult;
    }

    @RequestMapping(value = "/deleteUser")
    @ResponseBody
    public ServiceResponse deleteUser(int id, HttpSession session) {
        ServiceResponse serviceResponse = userDAOService.deleteUser(id);
        session.setAttribute("allUser", userDAOService.getAllUser());
        return serviceResponse;
    }

    @RequestMapping(value = "/logout")
    public String logout(HttpSession httpSession, HttpServletRequest request, HttpServletResponse
serviceResponse) {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if (auth != null){
            new SecurityContextLogoutHandler().logout(request, serviceResponse, auth);
        }
        return "redirect:login";
    }

    @RequestMapping(value = "/getById")
    @ResponseBody
    public Map<String, Object> getById(int id) {
        User user = userDAOService.get(id);
        String resultText = JSON.toJSONString(user);
        Map<String, Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText", resultText);
        return ajaxResult;
    }

    @RequestMapping(value = "/getDetailById")
    @ResponseBody
    public Map<String, Object> getDetailById(int id) {
        UserDetail userDetail = userDetailDAOService.getDetail(id);
        String resultText = JSON.toJSONString(userDetail);
        Map<String, Object> ajaxResult = new HashMap<>();
        ajaxResult.put("resultText", resultText);
        return ajaxResult;
    }
```

```
   private static String reg = "(?:')|(?:--)|(/\\*(?:.|[\\n\\r])*?\\*/)|"
         +
"(\\b(select|update|union|and|or|delete|insert|trancate|char|into|substr|ascii|declare|exec|product
Count|master|into|drop|execute)\\b)";

   private static Pattern sqlPattern = Pattern.compile(reg, Pattern.CASE_INSENSITIVE);

   private boolean isValid(String str) {
      if (sqlPattern.matcher(str).find()) {
         return false;
      }
      if (str.length() > 15)
         return false;
      return true;
   }

}

POJO
@Entity
@Table(name="products")
public class Product {
   private int id;
   private String name;
   private int productPrice;
   private int productCount;

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)

   @Column(name="id")
   public int getId() {
      return id;
   }

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

   @Column(name="name")
   public String getName() {
      return name;
   }

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

   @Column(name="productPrice")
   public int getPrice() {
      return productPrice;
```

```
    }

    public void setPrice(int productPrice) {
        this.productPrice = productPrice;
    }

    @Column(name="productCount")
    public int getproductCount() {
        return productCount;
    }

    public void setproductCount(int productCount) {
        this.productCount = productCount;
    }

}

@Entity
@Table(name="shoppingcart")
@IdClass(value=ShoppingCartPrimaryKey.class)
public class ShoppingCart {
    private int userId;
    private int Id;
    private int productPrice;
    private int productCount;

    @Id
    @Column(name="user_id")
    public int getId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    @Id
    @Column(name="product_id")
    public int getId() {
        return Id;
    }

    public void setId(int Id) {
        this.Id = Id;
    }

    @Column(name="product_price")
    public int getprice() {
        return productPrice;
    }

    public void setprice(int productPrice) {
        this.productPrice = productPrice;
```

```
    }

    @Column(name="productCount")
    public int getproductCount() {
        return productCount;
    }

    public void setproductCount(int productCount) {
        this.productCount = productCount;
    }
}

public class ShoppingCartPrimaryKey implements Serializable {
    private int userId;
    private int Id;

    public int getId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getId() {
        return Id;
    }

    public void setId(int Id) {
        this.Id = Id;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof ShoppingCartPrimaryKey)) return false;

        ShoppingCartPrimaryKey that = (ShoppingCartPrimaryKey) o;

        if (userId != that.userId) return false;
        return Id == that.Id;

    }

    @Override
    public int hashCode() {
        int resultText = userId;
        resultText = 31 * resultText + Id;
        return resultText;
    }
}

@Entity
```

```
@Table(name="shoppingrecord")
@IdClass(value=ShoppingRecordPrimaryKey.class)
public class ShoppingRecord {
   private int userId;
   private int Id;
   private int orderStatus;
   private int productPrice;
   private int productCount;
   private String orderTime;

   @Id
   @Column(name="user_id")
   public int getId() {
      return userId;
   }

   public void setUserId(int userId) {
      this.userId = userId;
   }

   @Id
   @Column(name="product_id")
   public int getId() {
      return Id;
   }

   public void setId(int Id) {
      this.Id = Id;
   }

   @Id
   @Column(name="orderTime")
   public String getTime() {
      return orderTime;
   }

   public void setTime(String orderTime) {
      this.orderTime = orderTime;
   }

   @Column(name="order_status")
   public int getOrderStatus() {
      return orderStatus;
   }

   public void setOrderStatus(int orderStatus) {
      this.orderStatus = orderStatus;
   }

   @Column(name="product_price")
   public int getprice() {
      return productPrice;
   }
```

```
    public void setprice(int productPrice) {
        this.productPrice = productPrice;
    }

    @Column(name="productCount")
    public int getproductCount() {
        return productCount;
    }

    public void setproductCount(int productCount) {
        this.productCount = productCount;
    }
}

public class ShoppingRecordPrimaryKey implements Serializable {
    private int userId;
    private int Id;
    private String orderTime;

    public int getId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getId() {
        return Id;
    }

    public void setId(int Id) {
        this.Id = Id;
    }

    public String getTime() {
        return orderTime;
    }

    public void setTime(String orderTime) {
        this.orderTime = orderTime;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof ShoppingRecordPrimaryKey)) return false;

        ShoppingRecordPrimaryKey that = (ShoppingRecordPrimaryKey) o;

        if (getId() != that.getId()) return false;
        if (getId() != that.getId()) return false;
```

```
        return getTime().equals(that.getTime());
    }

    @Override
    public int hashCode() {
        int resultText = getId();
        resultText = 31 * resultText + getId();
        resultText = 31 * resultText + getTime().hashCode();
        return resultText;
    }
}

@Entity
@Table(name="user_main")
public class User {

    private int id;
    private String name;
    private String userEmail;
    private int role;

    @Id
    @GenericGenerator(name = "generator", strategy = "increment")
    @GeneratedValue(generator = "generator")

    @Column(name="id")
    public int getId() {
        return id;
    }

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

    @Column(name="name")
    public String getName() {
        return name;
    }

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

    @Column(name="userEmail")
    public String getEmail() {
        return userEmail;
    }

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

    @Column(name="role")
```

```
    public int getRole() {
        return role;
    }

    public void setRole(int role) {
        this.role = role;
    }
}


@Entity
@Table(name="user_detail")
public class UserDetail {
    private int id;
    private String password;
    private String address;

    @Id
    @GenericGenerator(name = "generators", strategy = "assigned")
    @GeneratedValue(generator = "generators")

    @Column(name="id")
    public int getId() {
        return id;
    }

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

    @Column(name="password")
    public String getPassword() {
        return password;
    }

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


    @Column(name="address")
    public String getAddress() {
        return address;
    }

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

}
```