Clothes Shopping Website

**GROUP 5 of BigData3:**

Leader:

闫冰洁(Yan Bingjie)

Members:

牛聪(Niu Cong)

滕畅(Teng Chang)

王楠(Wang Nan)

2019.12.18

CERTIFICATE

This clothes shopping website project report reflects the original project independently completed by 4 students from the fifth group of class 3, big data, according to the requirements of niit .

Coordinator: Rupali、Abhijeet、Manmeet、delucia

ACKNOWLEDGEMENT

**1.What inspired us to write this project**

First of all, in terms of technology, there are many codes similar to online clothing sales websites. If we encounter a bottleneck in production, there will be a lot of information for us to refer to.Then, in terms of popularity, online clothing sales sites are more easily popularized, which can effectively improve the quantity and quality of user feedback we can get within a limited time, providing us with a greater chance to maintain and upgrade the project.

**2.Who helped us**

In the process of making the difficult project, we have received a lot of help from many people, including Mr. Liu decai, Mr. Rupali, Mr. Abhijeet and all the classmates and relatives who have put forward ideas, ideas and provided help for our project, all of whom have provided us with great technical and emotional help.

**3. Show appreciation**

We would like to express our sincere thanks to all the people who helped us overcome the difficulties and stayed with us through this boring and interesting day.

**4. Briefly introduce the process of developing this project**

On the axis of our group discussions, we had four major discussions on the development of the project.In the first discussion, we determined the division of labor. The group leader arranged different Numbers of team members to complete the task according to the weight of each task, indicating the direction of our future efforts.The second time, nearly three weeks after the first discussion, we completed and combined all our project contents, and made evaluation and improvement Suggestions within the group. Then we carried out the next round of rectification.The third time, one week after the second discussion, we consolidated and discussed all the contents again, and confirmed the completion of the project.During the fourth discussion, we determined the content and presentation of PPT.

**Description about Technologies used**

Operating system: Windows、Linux

Database: Mysql、HBase

Servlet: Tomcat

Database.Connection.driver: Java web integrated development environment.

Data Analysis: Hadoop MapReduce

**System Analysis(Requirements)**

**1. Functional requirement**

|  |
| --- |
| **User login and registration** |
| **View the main category of goods** |
| **Search for goods** |
| **Add to shopping cart** |
| **Payment** |
| **Data Analysis** |

1.1 Function block description

(1)Registration function.

Customers first have to register as onlinemall users. When registering,youonly need to fill in the login user name, password and E-mail address. After registration, the user can continue to truthfully fill in the detailed personal information and consignee information, and can change the password, query and modify the order.  
(2)Select product features.

Customers browse online shopping malls, their own needs of products into the shopping cart, can continue to add goods.

(3) Manage the shopping cart.

After selecting the goods, the customer can go to the shopping cart page, view the goods he wantsto buy, modify the quantity of a certain item, cancel the purchase of a certain item and empty the whole shopping cart.

(4) Order function.

The customer submits the order after identifying the items in the shopping cart. If the customer has filled in the consignee information, the page displays the information and the customer confirms it. If it is not filled in, the corresponding form will be displayed for it to fill in. The system will record the consignee information submitted by the customer for use in the next shopping. After submitting the order, customers can check the order inthe online mall and cancel or modify the order that has not been processed.

1. **Analysis requirement for own log**
   1. Top 5 most popular items

The number of clicks on a product can reflect the popularity of a product, so our definition of the most popular product is Top5 of the most clicked products of all products. Use Map to count all clicked products and use Reduce to calculate The number of clicks on all products is finally counted in the cleanup stage and Top5 is output to the result file.

* 1. Top 5 best sellers

The sales volume of a product can reflect the best-selling degree of a product. Use Map to count all the purchased products, use Reduce to count the number of purchases of all products, and finally count Top5 in the cleanup stage and output it to the result file.

* 1. Top 3 purchase rate items

The product purchase rate can reflect the user group's loyalty to the product and the degree of acceptance of the product description. Use Map to count all products, and reduce the number of clicks and purchases for each product in Reduce. The formula purchase rate = purchase / (Clicks + purchases) to record, and Top3 is counted in the cleanup stage.

* 1. Top 5 purchase combinations

By analyzing the combination of purchases, you can use bundled sales or special promotions to increase sales. Use Map to count the individual purchases of all users, count all product combinations during the Reduce phase, and finally calculate Top5 during the cleanup phase.

**3.Analysis requirement for internet log**

3.1 Top 5 most popular items

The number of clicks on a product can reflect the popularity of a product, so our definition of the most popular product is Top5 of the most clicked products of all products. Use Map to count all clicked products and use Reduce to calculate The number of clicks on all products is finally counted in the cleanup stage and Top5 is output to the result file.

3.2 Top 5 best sellers

The sales volume of a product can reflect the best-selling degree of a product. Use Map to count all the purchased products, use Reduce to count the number of purchases of all products, and finally count Top5 in the cleanup stage and output it to the result file.

3.3 Top 3 purchase rate items

The product purchase rate can reflect the user group's loyalty to the product and the degree of acceptance of the product description. Use Map to count all products, and reduce the number of clicks and purchases for each product in Reduce. The formula purchase rate = purchase / (Clicks + purchases) to record, and Top3 is counted in the cleanup stage.

3.4 Top 5 purchase combinations

By analyzing the combination of purchases, you can use bundled sales or special promotions to increase sales. Use Map to count the individual purchases of all users, count all product combinations during the Reduce phase, and finally calculate Top5 during the cleanup phase.

**Detailed introduction**

**1. Software positioning**

The positioning of this website is to provide users with a more convenient and fast way to buy clothes.That is, it breaks through the limitation of region in space, and gives users more power to choose clothes.In terms of time, it saves the time of going to different shopping malls and stores to choose and try on clothes, adding more fun to people's life.

**2. Basic functions of the software**

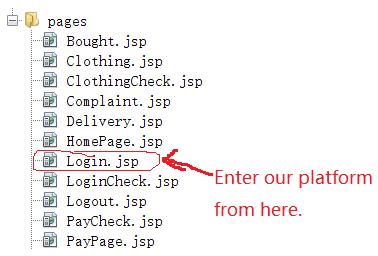
The website is a user-oriented platform, including: online registration, online login, browse the product page, shopping, order submission, payment, view orders, complaints and other operations.

**3. How to run your website**

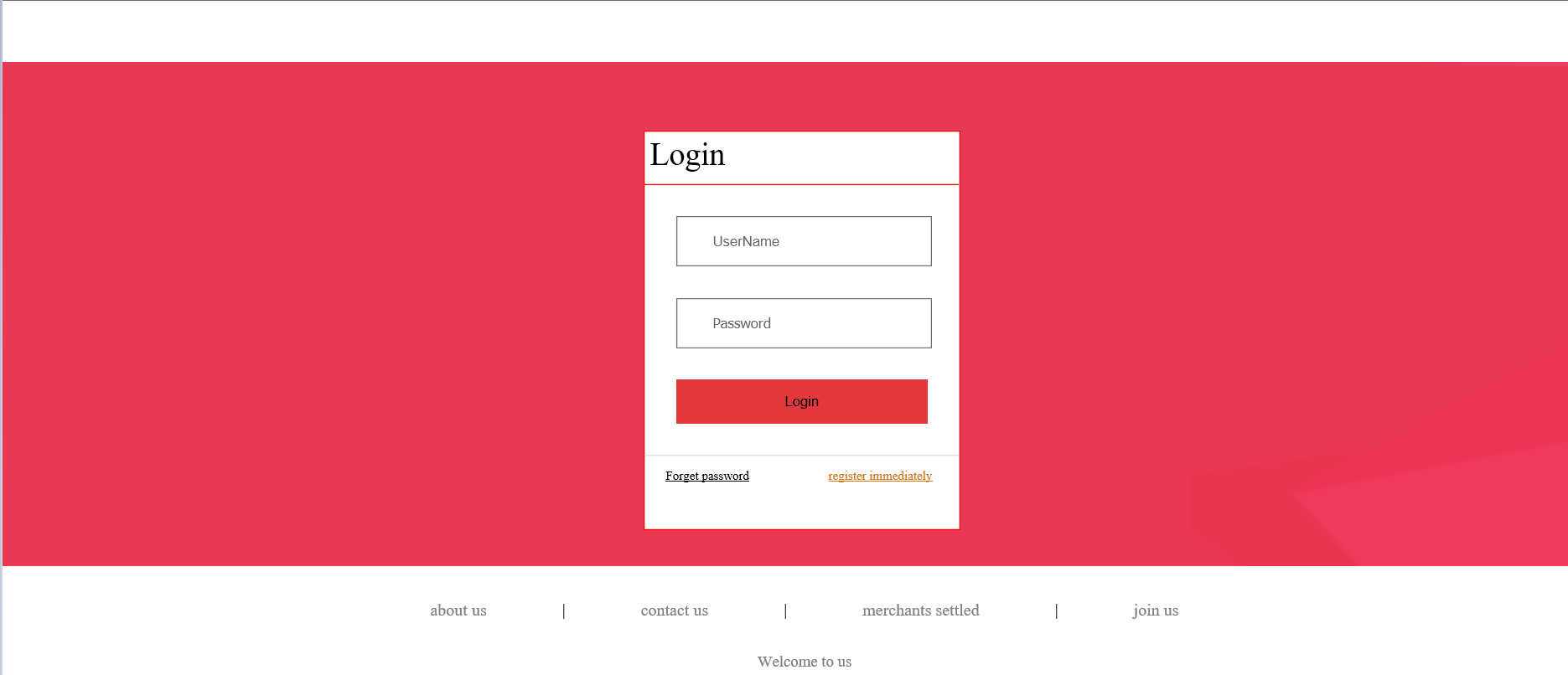
Use netbeans IDE software to run the program.

**4. Instructions**

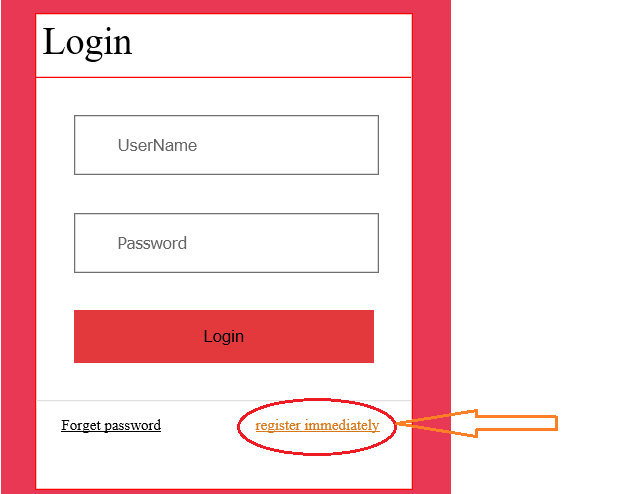
(1) From/Project/webpage/first/pages for the Login. jsp, the right to run the. The JSP file



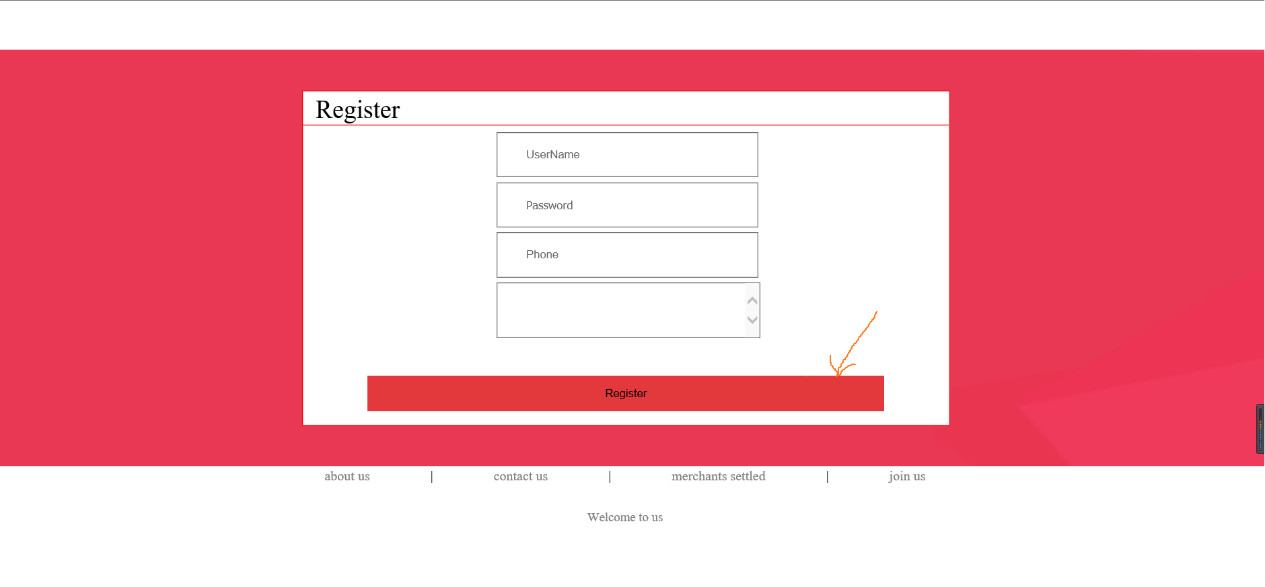
1. After running, the following interface appears



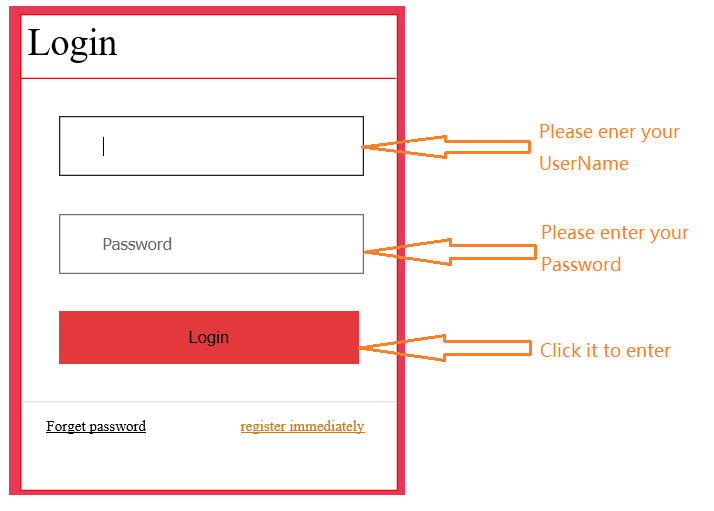
1. If this is the first time for you to log into our website, please click "Register immediately" in the bottom right corner for user registration (if you have already registered, ignore this step)



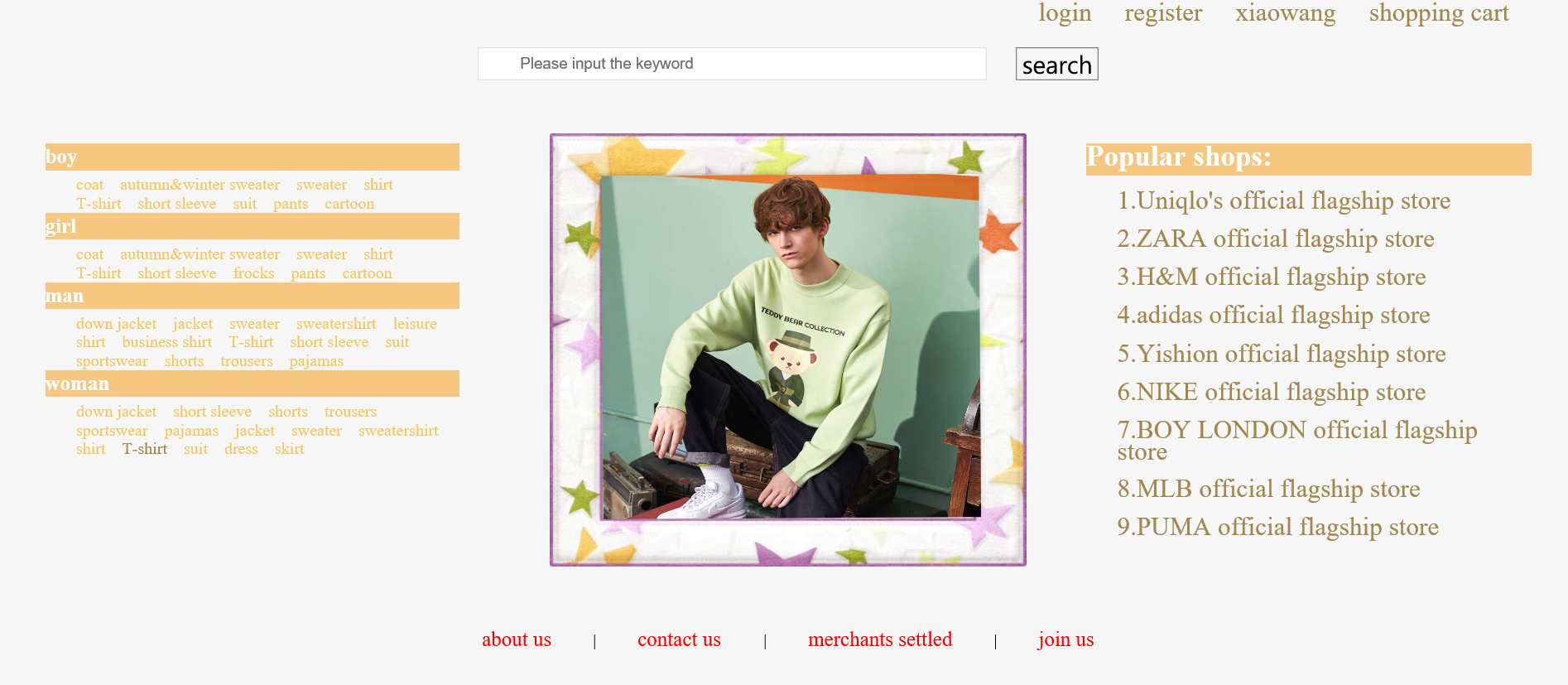
(4) After filling in the user registration interface, click the "Register" button to go back to the Login interface to Login (if you have already registered, ignore this step)



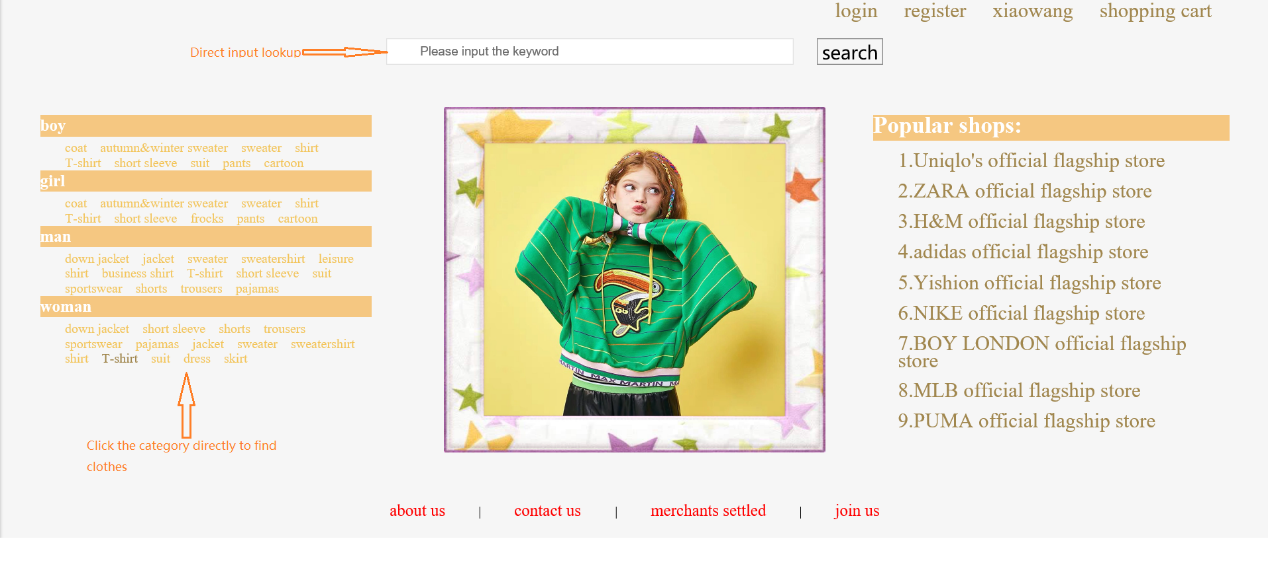
(5) Enter the account and password, then click "Login" button to enter the website.



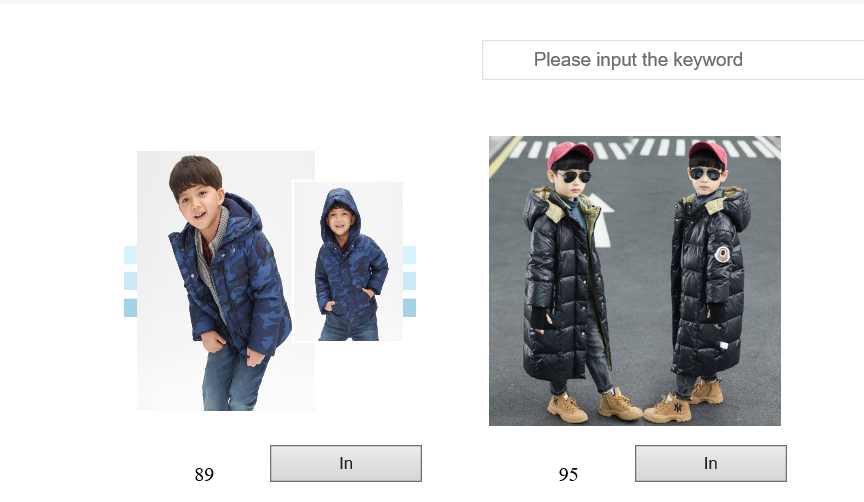
1. The following page appears after entering this website:



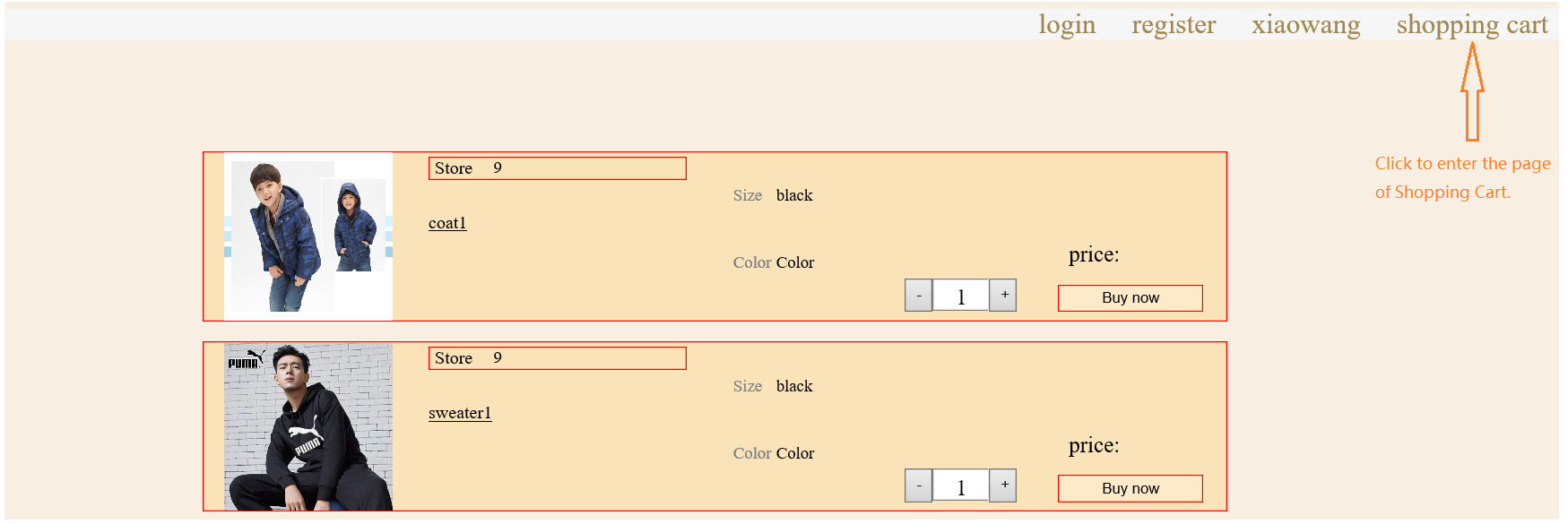
1. There are two ways to find products in this website:



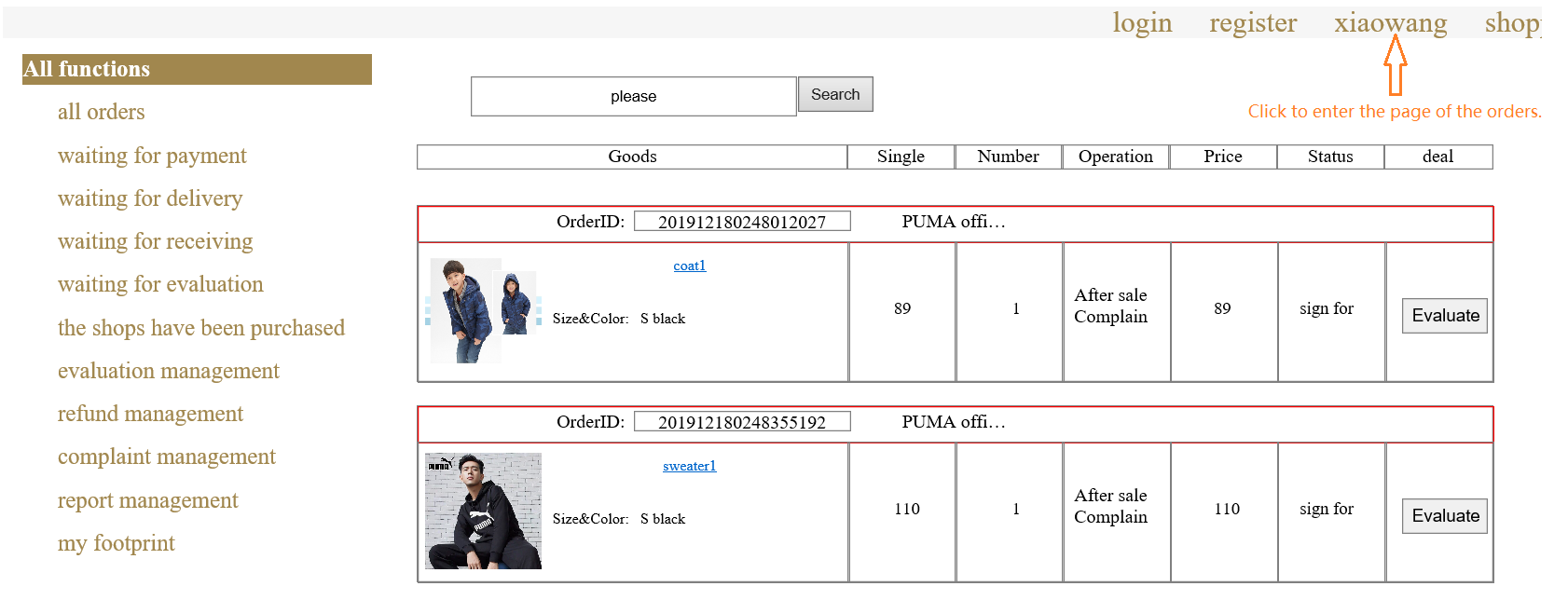
1. Enter the product interface to browse the product



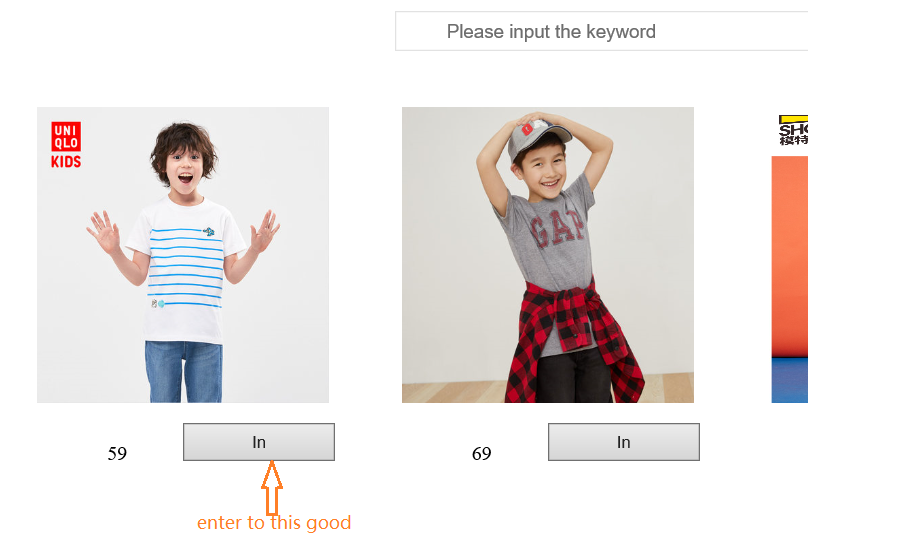
1. Go to the shopping cart to view the items that have been added to the shopping cart



1. View purchased items (unreceived and received)



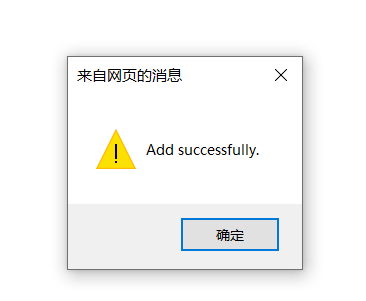
1. Enter the product page from the browse product page



(12) Click add to cart



(13) Add to cart successfully, click ok to return



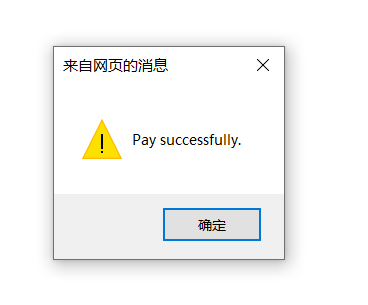
1. Click to pay



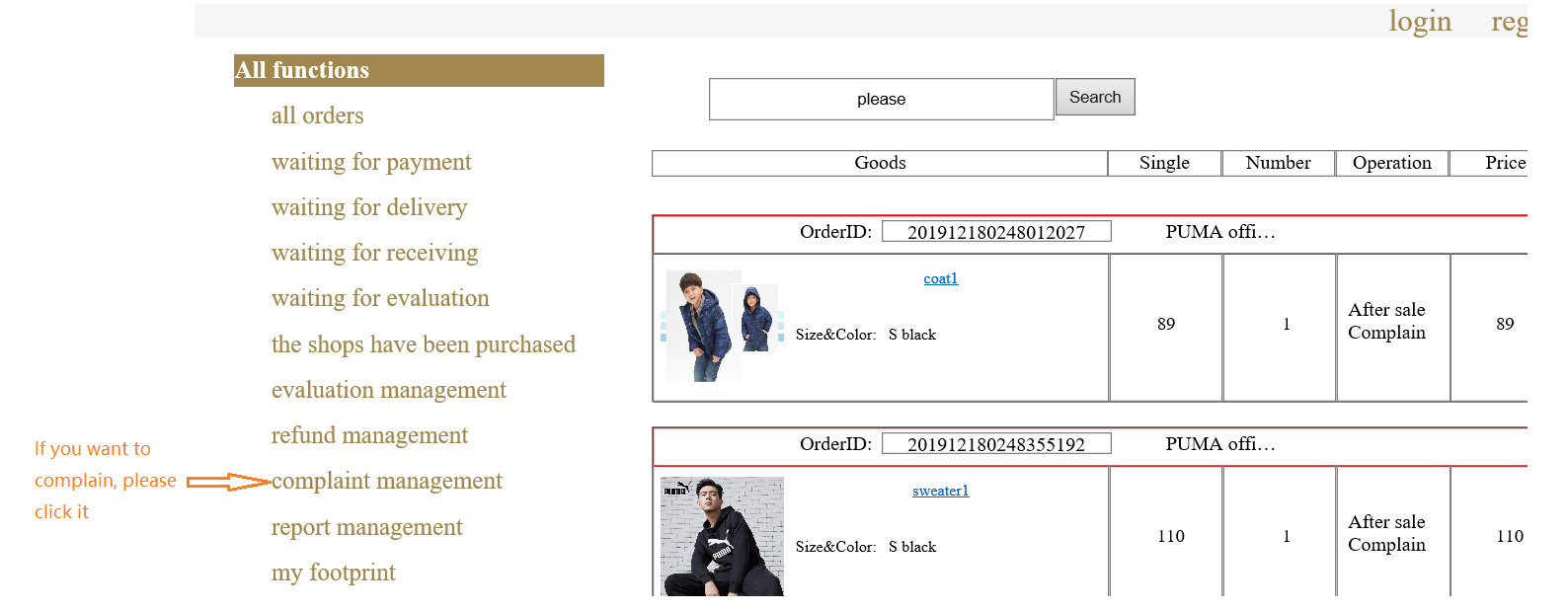
1. Pay for



(16) Purchase successful, click "ok" to return

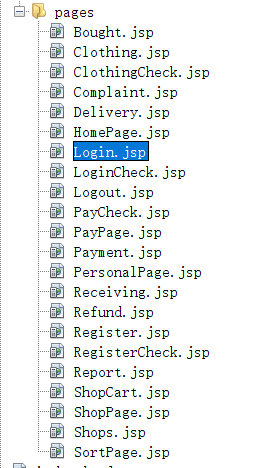


(17) If you want to complain, please click the button in the page



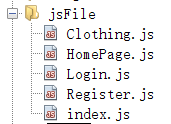
**5. Code directory structure description**

(1) The directory used to display the page



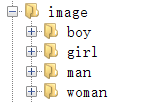
The above is the directory used to display the page, where the Bought. JSP is the order interface, which displays the information of each order Bought by the user in detail.Clothing. JSP is the product interface that displays the details of an item;Clothingcheck.jsp carried on clothing.jsp, logging the actions of the user on the item, such as adding to the shopping cart, buying directly, etc.;Complaint.jsp displays the user's historical comments or Suggestions about a product or the site;The delivery.jsp displays the user's order status;HomePage. JSP as the user's HomePage;Login.jsp is the Login interface;Logincheck.jsp for login.jsp to undertake, through the log of the user Login status, including the user Login success, Login failure details;PayCheck. JSP is the payment confirmation interface, which logs the information of the user's payment success and payment failure.Paypage.jsp is the output interface of the payment password, which displays the necessary information such as the amount of payment;The payment.jsp displays the status of each of our expenditures;Receiving. JSP is the logistics information page, showing the delivery progress of the goods for the user;Refoundation.jsp is the product return interface, showing the user the detailed history of the product return and exchange.Register.jsp for the registration interface, the user must enter the user name, password and other necessary information can be logged in successfully;Registercheck.jsp for register.jsp to do the following, through the log record the user registration success of the information;Report.jsp is the user feedback interface, showing each feedback of the user to this website;Shopcart.jsp for the shopping cart interface, detailed display of the user added to the shopping cart information;Shoppage.jsp for the store page, showing all the store's treasures;Sortpage.jsp for browsing the product page, through the log record users for each product browsing.

(2) The directory used to decorate the display page with events



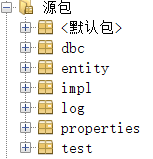
Above is the catalog used to decorate the display page with events, where clothes.js is the collection of click events for clothes.jsp, including the increase or decrease of the number of items, the selection of size and color, and the addition to the shopping cart and direct purchase operations;HomePage. Js is the collection of click events for HomePage. JSP;Login.js is the collection of click events for login. JSP, which includes immediate registration, forgetting the password and successfully logging into the next page;Register.js is the collection of click events of register.jsp, including the blank of user name and password, wrong mobile phone number, and successfully jump to the login page;Index.js is the set of the operation of the first page picture rotation.

(3) The catalog used to store the pictures of the products



The above figure is a collection of commodity pictures, all concentrated under the image package.

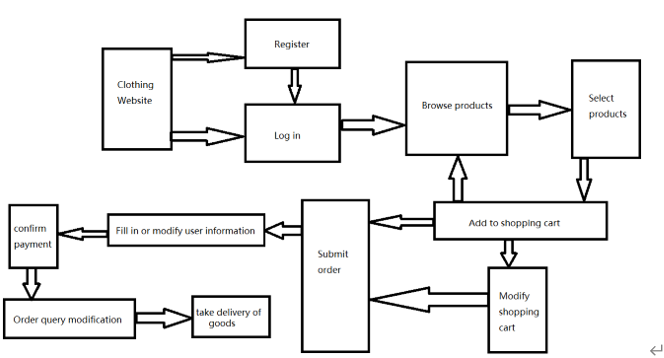
(4) The directory used to beautify the display page



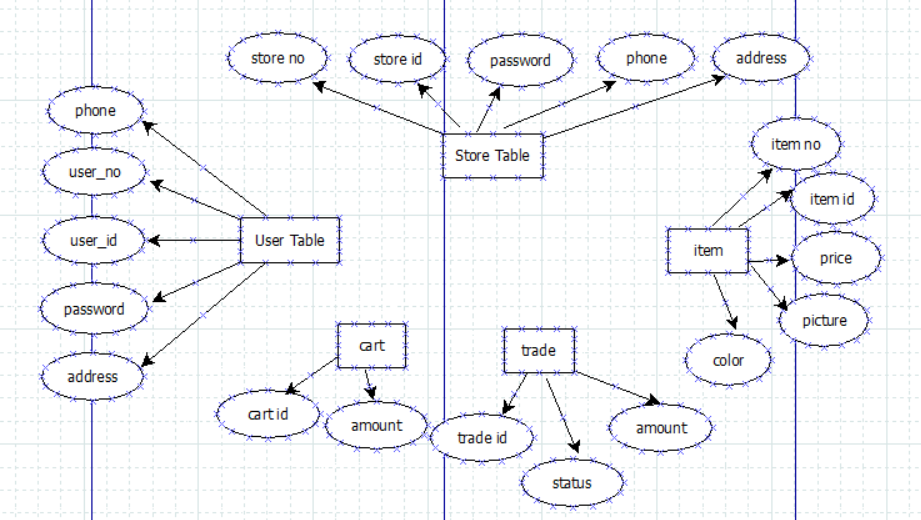
The above figure shows the directory used to connect to the database and call the collection of class packages displayed on the website of the database data.Entity is used for adding, deleting, modifying and checking the data of a single commodity, order, shopping cart and store.The impl package is used to search for the overall merchandise, order, shopping cart, store, and input the overall data;The log package is used to hold log file information;The properties package is used to hold information about database users, passwords, and so on.The test package contains the code used to test the site.

**Flow Chart**

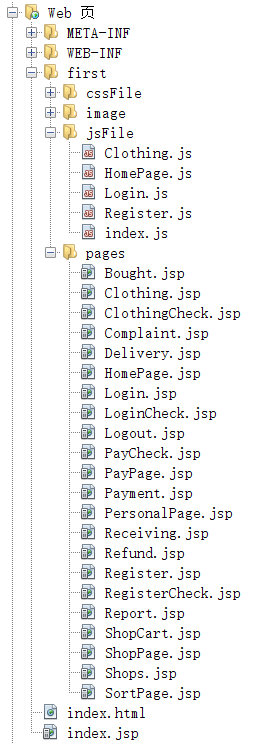
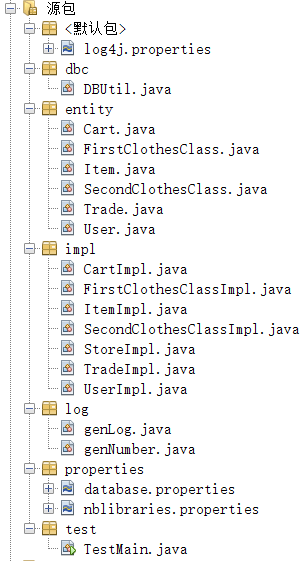
1. **Client function**



1. **ER diagram for database Design**



**Class Diagram**



**Individual Milestone Schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Start Time** | **Finish Time** | **Developer Name** |
| SRS | 2019/11/02 | 2019/11/04 | TengChang、WangNan |
| Find web logs | 2019/11/06 | 2019/11/08 | YanBingjie |
| Web Design | 2019/11/06 | 2019/11/25 | TengChang、WangNan |
| DataBase | 2019/11/10 | 2019/11/25 | NiuCong |
| JDBC | 2019/11/12 | 2019/11/25 | YanBingjie |
| Log Analysis | 2019/12/01 | 2019/12/17 | YanBingjie |
| Project Report | 2019/12/17 | 2019/12/18 | TengChang、WangNan |
| PPT | 2019/12/18 | 2019/12/18 | NiuCong |

**Important Code**

/\*Log file\*/

public static void Log(String user\_id, String ip\_address, String action\_id, String category\_id1, String category\_id2,

String item\_id, String seller\_id) { // timestamp, user\_id, ip\_address, action\_id, first\_category\_id, second\_category\_id, item\_id, seller\_id

Logger log = Logger.getLogger(genLog.class.getName());

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String time = df.format(new Date());

log.info(time + "," + user\_id + "," + ip\_address + "," + action\_id + "," + category\_id1 + "," + category\_id2 + "," + item\_id + "," + seller\_id);

}

/\*User operation file\*/

public class UserImpl {

public static int add(User user) throws Exception {

String sql = "insert into user(user\_no,user\_id,user\_passwd,user\_phone, user\_addr) values(?,?,?,?,?)";

if(user.getUser\_no() == null || user.getUser\_no().length() == 0) {

user.setUser\_no(genNumber.getno());

}

try (Connection conn = getConnection();) {

return exceuteUpdate(conn, sql, new Object[]{user.getUser\_no(), user.getUser\_id(), user.getUser\_passwd(), user.getUser\_phone(), user.getUser\_addr()});

}

}

public static User findIdUser(String name) throws Exception {

String sql = "select \* from user where user\_id=?";

User use = null;

try (Connection conn = getConnection();

PreparedStatement pst = conn.prepareStatement(sql)) {

pst.setString(1, name);

try (ResultSet rs = pst.executeQuery()) {

if (rs.next()) {

use = new User(rs.getString("user\_no"), rs.getString("user\_id"), rs.getString("user\_passwd"), rs.getString("user\_phone"), rs.getString("user\_addr"));

}

return use;

}

}

}

public static User findNoUser(String name) throws Exception {

String sql = "select \* from user where user\_no=?";

User use = null;

try (Connection conn = getConnection();

PreparedStatement pst = conn.prepareStatement(sql)) {

pst.setString(1, name);

try (ResultSet rs = pst.executeQuery()) {

if (rs.next()) {

use = new User(rs.getString("user\_no"), rs.getString("user\_id"), rs.getString("user\_passwd"), rs.getString("user\_phone"), rs.getString("user\_addr"));

}

return use;

}

}

}

}

/\*PayCheck file\*/

<body>

<%

String pass = request.getParameter("pass");

User u = (User) session.getAttribute("user");

genLog log = new genLog();

// genIp ip = new genIp();

String item\_id = (String)session.getAttribute("item\_id");

String store = StoreImpl.findById(item\_id);

Cookie[] co = request.getCookies();

if (!u.getUser\_passwd().equals(pass)) {

log.Log(u.getUser\_id(), InetAddress.getLocalHost().getHostAddress(), "08", co[0].getName(), co[1].getName(), item\_id, store);

out.println("<script>alert(\"Password is incorrect.\")</script>");

out.println("<script> window.history.back(-1); </script>");

} else {

log.Log(u.getUser\_id(), InetAddress.getLocalHost().getHostAddress(), "09", co[0].getName(), co[1].getName(), item\_id, store);

Item item = ItemImpl.findIdItem(item\_id);

String item\_no = item.getItem\_no();

List<Cart> c = CartImpl.findCartUser(u.getUser\_no()); //通过item\_no找到该cart

int i = 0;

while (i < c.size()) {

if (c.get(i).getItem\_no().equals(item\_no)) {

break;

} else {

i++;

}

}

Trade trade = new Trade(null, c.get(i).getItem\_amount(), item\_no, c.get(i).getUser\_no(), c.get(i).getItem\_size(), c.get(i).getItem\_color());

int t = TradeImpl.add(trade);

if(c.get(i).getCart\_no()!=null){

CartImpl.deleteCartId(c.get(i).getCart\_no());

}

out.println("<script>alert(\"Pay successfully.\")</script>");

out.println("<script> window.location.href=\"HomePage.jsp\"; </script>");

}

%>

</body>

/\*connect database\*/

public static void init() throws IOException {

Properties params = new Properties();

String configFile = "properties/database.properties";

InputStream is = DBUtil.class.getClassLoader().getResourceAsStream(configFile);

params.load(is);

driver = params.getProperty("driver");

url = params.getProperty("url");

username = params.getProperty("username");

password = params.getProperty("password");

}

/\*Update database\*/

public static int exceuteUpdate(Connection conn, String preparedSql, Object[] param) throws SQLException {

int num;

try (Connection connection = conn;

PreparedStatement pstmt = connection.prepareStatement(preparedSql);) {

if (param != null) {

for (int i = 0; i < param.length; i++) {

pstmt.setObject(i + 1, param[i]);

}

}

num = pstmt.executeUpdate();

return num;

}

}

/\*BestSellMapReduce\*/

static class MapDistinctTask extends Mapper<LongWritable, Text, Text, Text> {

@Override

protected void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] ss = value.toString().split(",");

// 3 action\_id 6 item\_id 7 store\_id

if("1".equals(ss[3])) { // buy

context.write(new Text("1"), new Text(ss[6]));

}

}

}

static class ReduceDistinctTask extends Reducer<Text, Text, Text, Text> {

TreeMap<Integer, String> tree = new TreeMap<Integer, String>();

@Override

protected void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

tree.clear();

List<String> s = new ArrayList();

for (Text val : values) {

String ss = val.toString();

s.add(ss);

}

Collections.sort(s);

for (int i = 0; i < s.size(); i++) {

int j = i;

for (; j < s.size() && s.get(j).equals(s.get(i)); j++) ;

tree.put(j - i + 1, s.get(i));

if (tree.size() > 5) {

tree.remove(tree.firstKey());

} i=j-1;

}

Iterator<Integer> it1 = tree.keySet().iterator();

while (it1.hasNext()) {

int kk = it1.next();

String vv = tree.get(kk);

context.write(new Text(vv), new Text(Integer.toString(kk)));

}

}

}

/\*BuyRateMapReduce\*/

static class MapDistinctTask extends Mapper<LongWritable, Text, Text, Text> {

@Override

protected void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] ss = value.toString().split(",");

// 4 action\_id 6 item\_id 7 store\_id

context.write(new Text(ss[6]), new Text(ss[3]));

}

}

static class ReduceDistinctTask extends Reducer<Text, Text, Text, Text> {

TreeMap<Double, String> tree = new TreeMap<Double, String>();

@Override

protected void cleanup(Context context) throws IOException, InterruptedException{

Iterator<Double> it1 = tree.keySet().iterator();

while (it1.hasNext()) {

Double kk = it1.next();

String vv = tree.get(kk);

context.write(new Text(vv), new Text(kk.toString()));

}

}

@Override

protected void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum0 = 0, sum1 = 0;

for(Text t: values) {

if("1".equals(t.toString())) sum1 ++;

else sum0 ++;

}

tree.put(1.0\*sum1/(sum1+sum0), key.toString());

}

}

/\*CombineSoldTop5\*/

static class MapDistinctTask extends Mapper<LongWritable, Text, Text, Text> {

@Override

protected void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] s = value.toString().split(",");

if ("1".equals(s[3])) { // buy

context.write(new Text(s[1]), new Text(s[6]));

}

}

}

static class ReduceDistinctTask extends Reducer<Text, Text, Text, Text> {

TreeMap<Integer, String> tree = new TreeMap<Integer, String>();

@Override

protected void cleanup(Context context) throws IOException, InterruptedException{

Iterator<Integer> it1 = tree.keySet().iterator();

while (it1.hasNext()) {

Integer kk = it1.next();

String vv = tree.get(kk);

context.write(new Text(vv), new Text(kk.toString()));

}

}

@Override

protected void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

List<String> ls = new ArrayList();

List<String> res = new ArrayList();

// tree.clear();

for (Text t : values) {

ls.add(t.toString());

}

int tot = ls.size();

for (int i = 0; i < tot; i++) {

for (int j = i + 1; j < tot; j++) {

String ss = ls.get(i) + "+" + ls.get(j);

if (ls.get(i).compareTo(ls.get(j)) > 0) {

ss = ls.get(j) + "+" + ls.get(i);

}

res.add(ss);

}

}

Collections.sort(ls);

for (int i = 0; i < res.size(); i++) {

int j;

for (j = i; j < res.size() && res.get(i).equals(res.get(j)); j++) ;

tree.put(j - i + 1, res.get(i));

if (tree.size() > 5) {

tree.remove(tree.firstKey());

} i=j-1;

}

}

}

/\*MostPopularItem\*/

static class MapDistinctTask extends Mapper<LongWritable, Text, Text, Text> {

@Override

protected void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] ss = value.toString().split(",");

// 4 action\_id 6 item\_id 7 store\_id

if("0".equals(ss[3])) { // click

context.write(new Text("1"), new Text(ss[6]));

}

}

}

static class ReduceDistinctTask extends Reducer<Text, Text, Text, Text> {

TreeMap<Integer, String> tree = new TreeMap<Integer, String>();

@Override

protected void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

tree.clear();

List<String> s = new ArrayList();

for (Text val : values) {

String ss = val.toString();

s.add(ss);

}

Collections.sort(s);

for (int i = 0; i < s.size(); i++) {

int j = i;

for (; j < s.size() && s.get(j).equals(s.get(i)); j++) ;

tree.put(j - i + 1, s.get(i));

if (tree.size() > 5) {

tree.remove(tree.firstKey());

} i=j-1;

}

Iterator<Integer> it1 = tree.keySet().iterator();

while (it1.hasNext()) {

int kk = it1.next();

String vv = tree.get(kk);

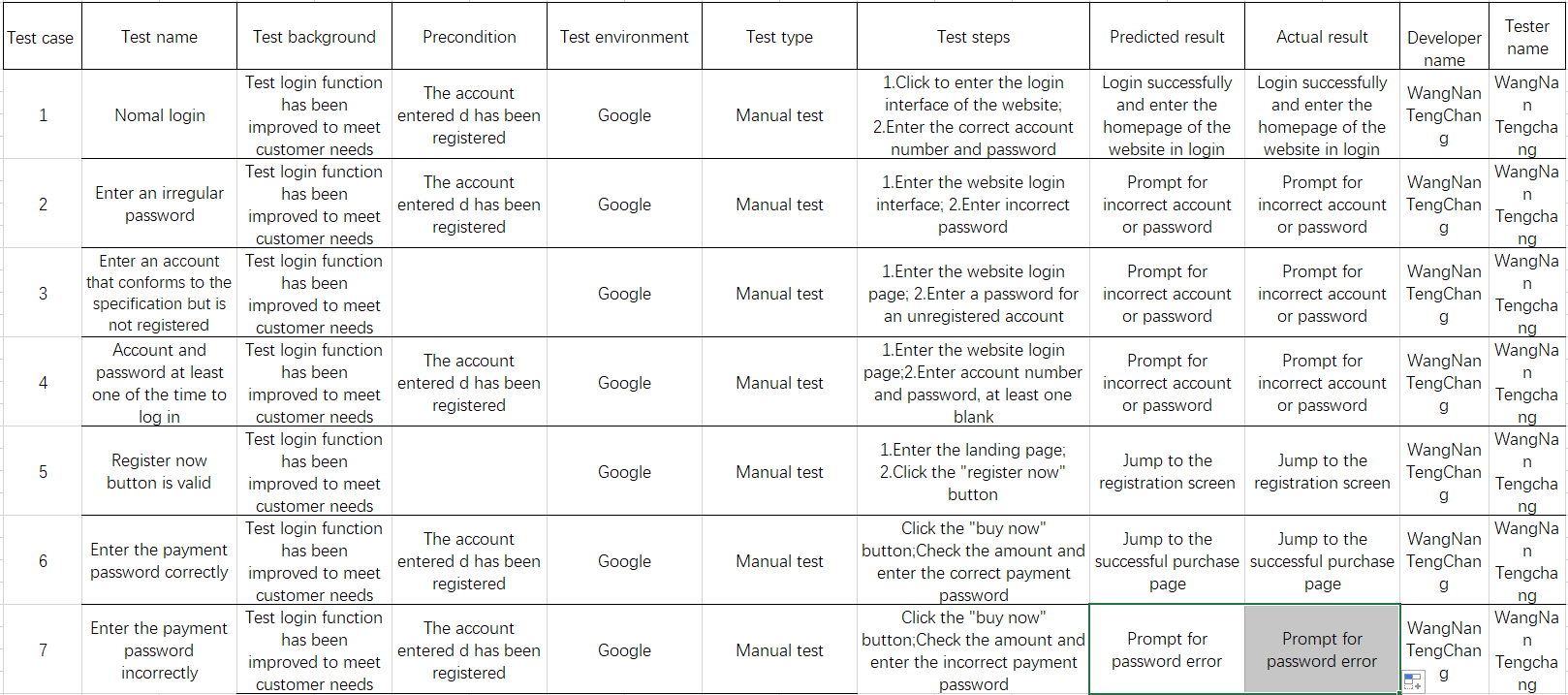
context.write(new Text(vv), new Text(Integer.toString(kk)));

}

}

}

**Test**

****

**Potential Improvements**

**1. The security**

In the current system, even if the user USES the appropriate user name and password to enter the bank Intranet site, it is not completely secure, we can use session to prevent this problem.

**2. Data validation**

In the current system, all the number and date fields of the added page have data validation, but the update page does not.If the user makes an error during the update process, you should add data validation to all update pages.

**3. Limited historical memory**

Users sometimes log off their accounts by mistake. However, the accounts that have been logged off by mistake contain some data that the users have not been able to move out. At this time, there should be a period of logout.

**4. Page index**

Page index function if the table view page has a page index function, the behavior of searching from the drop-down menu would be inappropriate.For example, when the user views all the products and goes to the second page, the user cannot select any other options from the drop-down box.With this problem, the page index function is removed from the current system.