Split Credit Card Payment System.  
(Integration and Architecture)

Submitted to the  
Department of Computer Science  
College of Computing Sciences  
New Jersey Institute of Technology

in Partial Fulfillment of  
the Requirements for the Degree of  
Master of Science in Computer Science

By

Yongbin Gao
&
Nitesh Phatnani

Project Advisor: Dr. James Geller

APPROVALS

Proposal Numbers: _______________________________

Agree to Advise: ________________________________
( Project Advisor)

Date Submitted: ________________________________

Approved by: ________________________________
(MS in CS Committee)

Date Approved: ________________________________

New Jersey Institute of Technology
ABSTRACT
This project deals with internet credit card payments, specifically we allow a customer to split a purchase over several cards. The customers may face a complex utility optimization problem on each purchase, namely, which card (or which combination of cards, if allowed) would be the best one to use among multiple cards for this particular purchase.

A typical example can be found in a tradeoff decision based on a customer’s preferences. Let us suppose that a customer has two cards, A and B, and is interested in the following card features: interest rate and bonus miles. For simplicity, assume both cards have enough credit to cover the desired potential purchase. Suppose Card A offers a lower interest rate and one mile per dollar spent, while Card B offers a higher interest rate and two miles per dollar. Depending on how much the customer prefers avoiding interest payments or getting airline miles, the customer may be better off using one card over another or a combination of both cards.

So, in our system, a Virtual card (V-card) is created using fuzzy technology and local optimization algorithm each time the customer wants to use a combination of several credit cards. Also, to support the divisible card payments, the Virtual Card Manager (VCM) is added to the Payment Agent module. In all, there are three major modules from which the system is built, named Merchant module, Payment agent module and the Bank module.
# TABLE OF CONTENTS

1. INTRODUCTION ................................................................. 4

2. PREVIOUS WORK (ARCHITECTURE AND SYSTEM) ...................... 7

   2.1 MODULE A DETAILS ....................................................... 10

   2.2 MODULE B DETAILS ....................................................... 14

   2.3. MODULE C DETAILS ...................................................... 16

3. CURRENT SYSTEM ............................................................. 17

4. OUR CONTRIBUTION (MERCHANT WEBSITE) ................................. 20

5. MERCHANDISE RETURN .................................................... 28

6. DATABASE CONNECTIVITY .................................................. 35

7. EQUIPMENT CONFIGURATION ............................................ 36

8. CONCLUSION ...................................................................... 37

9. FUTURE WORK .................................................................. 38

10. REFERENCES ...................................................................... 39

APPENDIX A: ALLOCATION OF WORK .................................... 41

APPENDIX B: USER MANUAL .................................................. 43

APPENDIX C: SOURCE CODE .................................................. 45
1. INTRODUCTION

Credit cards are the payment choice in e-commerce. Despite the on-going development efforts on various kinds of new payment systems for e-commerce, online shoppers use credit cards for a majority of their purchases. Research shows that 85% of all Internet transactions are done with online credit card payments and that customers are more comfortable with and feels more secure about using credit cards over the Internet [1].

When people use credit cards, they expect functionalities different from, say, cash transactions. Credit cards, although not providing anonymity, offer the balance carryover functionality such that the purchase amounts on a credit card can be carried over to the future and be paid in installments with interest. Many credit cards offer additional features, such as cash-back on a percentage of total purchases made, travel protection, additional warranty, or airline frequent flier miles. In such a myriad of choices and features, a customer may be better off using a particular card, depending on his/her preferences and spending habits. For example, a customer who carries a large balance may prefer a card with a lower interest rate, while another customer who does not carry a balance, but likes traveling, may prefer to use a card affiliated with an airline company to receive free airline miles. Furthermore, customers are sometimes better off if they can use a combination of credit cards for a single purchase [15].

This project describes an infrastructure that supports the divisible payment of a single purchase [2]. In the new infrastructure, a Virtual card (V-card) is created and used each time the customer wants to use a combination of cards. This new infrastructure modifies the existing systems in two ways. First, to support the divisible card payments, the Virtual Card Manager (VCM) is added to the merchant side. The VCM handles the
divisible card approval process between the merchant and the respective credit card issuers. Second, to support the customer’s card-usage decisions, the new infrastructure provides the customer with a V-card Agent (VA). As which card to use is a complex decision, an optimization model is built into the VA. Based on the customer’s preferences, the VA generates the best option that may include using multiple cards for a single purchase [2].

It is believed that the proposed infrastructure is well suited for online purchases. The creation of the V-card does not create a physical card but only a valid card number, and thus this is well suited for Web purchases where no physical card needs to be handled. The VA’s optimization decision needs computing power, and therefore online purchases that use computers in the first place are a good fit for the divisible card payment infrastructure. It is also expected that this infrastructure will be effective in the emerging mobile commerce domain. The VA residing at the customer’s mobile device, for example, may assist the customer’s decisions at runtime [15].

The increased use of credit cards on the Internet has brought increased credit card fraud. Thus, the majority of research on credit card payments for e-commerce focuses on the security issues [17]. One study relevant to this work is the payment with single-use credit card numbers [10]. In order to reduce fraud with permanent card numbers, the card issuing banks, such as Visa, Mastercard, American Express, Discover, may issue a one-time use credit card number instead. For example, American Express’ Private Payments program allows consumers to obtain single-use numbers from American Express directly to be used for purchases. A card number expires after a purchase is made or after approximately 30 days from the date of issue. Although the one-time use credit card
number is primarily designed for protecting against card fraud, it is applicable to this
divisible card payment. When generating a virtual card, the Virtual Agent may use this
method to create a one-time use Virtual-card number [15].

There have been studies on divisible e-cash payment protocols [2,8]. These
studies focus on payment solutions that ensure anonymity and unlinkability while
allowing electronic coins to be divisible. That is, each coin can be spent incrementally as
long as the total purchases are limited to the monetary value of the coin. These projects
look at multiple purchases and multiple merchants, while our work is about one purchase
with one merchant but with multiple credit card issuers. The solutions devised for
divisible e-cash are therefore not directly transferable.

Most of studies on credit card payment security do not focus on the credit card
user’s practical decision-making problem. Users may face a complex utility optimization
problem on each purchase, namely, which card would be the best one to use among
multiple cards for this particular purchase. The user’s perspective of credit card uses and
payments based on her preferences or goals, however, has not been addressed in the
literature. The security and protection against fraud are of paramount importance, but as
technologies advance, capturing the user’s preferences and goals and customizing the use
of credit cards should also be an important issue in an electronic payment system [15].
2. PREVIOUS WORK

The previous architecture for the credit card payment system [15] is depicted in Figure 1 and described as follows:

1. When the customer finds a desired product on a website, he/she decides to buy the product online. The customer has to pay the bill in the end. If the merchant website has secure page, then the customer can enter the billing information on that page. If the merchant website does not have a secure page, the customer will be directed to the merchant’s secure payment gateway.

2. The billing information will be send to the payment gateway.
3. The payment gateway sends the billing information (such as customer address, credit card number, credit card holder name, expiry date) to the merchant account that has been set up by the online merchant account provider.

4. As the merchant account provider provides these services to number of merchants, it maintains a number of merchant accounts for all merchants.

5. The merchant account provider provides the credit card information processing to each merchant account such as transferring the billing information to the issuing bank of the credit card and sending back responses from issuing banks to merchants. Each issuing bank checks if the credit card information is valid and if the credit card has sufficient funds to cover the purchase. If so, it sets aside the amount of the purchase from the customer’s account for the merchant. If the credit card is invalid or the credit limit has been reached, the issuing bank sends back a denial code.

6. The issuing bank sends back an approval or denial code to the merchant account.

7. The approval (or denial) code is passed by the payment gateway back to the secure merchant site.

8. The approval code is passed to the customer. Usually, the payment gateway emails the customer the payment receipt.

9. At the end of the day, the merchant requests to settle all the transactions of the day. The merchant account settling service sends the request to capture funds to the acquiring bank.

10. The acquiring bank forwards the request to the issuing bank.
11. The card issuing bank pays funds to the acquiring bank and the funds are deposited to the merchant’s bank account. The actual funds reach the merchant’s checking account in approximately two business days.
2.1 MODULE A DETAILS

The Virtual Agent software is developed using JSP [16]. Below mentioned is the sequence of events in this module which does the function of creating a V-card.

1. The first page shows the Login to VA.

2. The registration screen appears next.

3. Next is the main menu. When the customer logs into the VA the main menu of the V-card system comes up. The initial window consists of three sub-menus: ‘my card list,’ ‘my preference,’ and ‘create a V-card.’

4. The VA menu screen appears next. The My Card List screen is shown next. By clicking ‘My Card List’ the customer sees the list of his cards. The form displays the card nickname and its card number. Clicking the card nickname reveals more detailed information about that card. The interface also provides two functions to edit and delete the existing card information in the list.

5. The customer may add an additional card by pressing the ‘Add a New Card’ link. In the prototype, the VA receives the information regarding the detailed features of the card from the customer, but we are exploring a way to automate the acquisition of such information. Once a new card is added, the VA can access the Website of that card to automatically receive the existing balance information (and therefore the available credit limit). This feature of automatically receiving the existing transactions has already been
used in commercial personal finance software, such as Quicken and Money.

6. Next window is for capturing the ‘My preferences’. At present, the utility function is computed by approximation based on a series of simple questions.

7. Next part shows the process of ‘Creating a V-card’. When the customer wants to purchase a product, she enters the purchase amount and pushes the ‘Go Optimization’ button. At present, the customer enters the purchase amount, but the VA should be able to receive this information automatically from the merchant site in the future. Then, the VA performs the optimization. For example, the customer enters the purchase amount of $500. This information, in conjunction with the previously entered information about credit cards and preferences, is used to calculate the optimization result.

8. Next is the result of the optimization. The optimized solution shows a list of cards to be used and the charge against each card.

9. The final step is to create a V-card. When the customer follows the suggestion (by selecting “Yes” to the question of “Create a V-Card” asked in the previous window), the VA creates a one-time use V-card number. The expiration date is set to be the next year from today at present.

10. When the user clicks on ‘Confirm’, the V-Card information is sent to the Bank. The Bank Simulation JSP checks with the database and returns the Approval Code. The Approval Code can be ‘Approved’ or ‘Denied’ based on the data and preferences.
There have been some improvements and additions to this module by another student.

The ‘My Card List’ showing available credit and the credit limit on each card has been displayed in Figure 2.

![Figure 2: My Card List](image)

The Bank approval has been captured and displayed to the user as shown in Figure 3.

![Figure 3: Bank Approval](image)
Below in Figure 4 is the screenshot after clicking “Capture funds.”

Also, you can look up your history information via click “View Transaction” button.

The function of “Capture funds” should be added to merchant site, and the information shown here is about merchant bank.

![Figure 4: Capture Funds](image-url)
2.2 MODULE B DETAILS

Module B is an enhanced, better and modern version of module A. The GUI for this module is very attractive and user friendly. The system roughly looks like as shown below in Figure 5:

![System Architecture Diagram]

**Figure 5: System Architecture [15]**

Module B has all the features shown in the Figure 5 implemented successfully. The system maintains a balance between closing date, interest rate and bonus rate based on weighted fuzzy decisions. The GUI is also pretty easy, simple and attractive. It follows all the 10 steps as in module A implemented correctly.

Here is how the GUI looks like as shown in Figure 6 and 7 for steps 7 and 8.
Figure 6: Optimization Start, Enter weights on criteria [17]

Figure 7: The Optimization Result [17]
2.3 MODULE C DETAILS

The module C is a simulation of the merchant’s website where a customer can order his items. When the item is selected and added to the cart then the system will prompt the user to create a V-card for the payment.

The system works as follows:

1. It shows the list of items on the merchants’ website that a customer comes to buy to this website.

2. Second screen is where the item is selected along with its quantity and price so that it can be added to the cart as shown in Figure 8.

3. When you click on checkout after the item being selected, it will take you to VA (Virtual Agent).

4. Then comes the functionality of Virtual Agent (VA).

![Figure 8: Merchant website showing ‘My Cart’](image)
3. CURRENT SYSTEM

Our contribution to this project has been playing a major role in redesigning the system for better usability. Earlier, the customers’ details like billing information and credit card details used to reside on the merchants’ website or the merchants’ secured payment gateway. But we conducted a survey and that survey indicated that the majority of the customers would hesitate to reveal their credit card details when it comes to saving them on the merchants’ database. So we redesigned the system to give it better features from a security perspective. So, to resolve the security issue, we added a new module to the current system. We added a secured ‘Payment Agent’ that keeps all the customer information in a database including the confidential credit card details. This ‘Payment Agent’ gives out all the necessary details upon request by the merchant. A customer has to register for an account with this ‘Payment Agent.’ Also, the merchant has to be registered with this ‘Payment Agent’ for approval of all the transactions.

So now the system has three modules namely:

1. Merchant Module.
2. Payment Agent.
3.1 MERCHANT MODULE

1. Customer registers with the Merchant website, provides all his contact details and logs in (This registration is a one time event).
2. Customer does his shopping there online.
3. Customer selects the items and adds them to the cart.
4. Customer goes to check out.
5. Customer selects payment method after which he/she is directed to the secured ‘Payment Agent’ page.

3.2 PAYMENT AGENT

1. Customer goes to Payment agent and registers (This registration is a one time event).
2. Customer gets a registration number and customer number for reference.
3. Payment Agent requests the customer to enter in all his card details.
4. Agent requests input of the customer preferences like mileage points, cash back, interest rates, payments deadlines, etc.
5. Customer enters all the different cards he has with the details and preferences.
6. Customer clicks on submit and all the details are stored in the database and he logs out. This is the end of registration.
7. The Payment Agent synchronizes the customer information with the bank to get the current balance on his cards.
8. Enter the amount of final transaction.
9. Payment Agent then using the fuzzy decision system, decides which cards to use to make the payment and generates a virtual card number.
10. The Payment Agent then requests the Customer for confirmation the above
payment details generated, as to which cards to use which amount.

11. Customer sends confirmation to the Payment Agent.

12. Payment Agent makes the payment to the merchant account providing the virtual
card number that was generated.

13. Customer gets confirmation of the payment being made with details of all the
cards used in the form of a receipt.

3.3 BANK MODULE

This module is prepared to simulate the bank. The bank has to reveal the balance
remaining on the cards upon request from the Payment Agent. The respective credit card
issuing banks have to transfer the money from the credit card accounts of the customers
to the respective merchant account. In case of returns they have to deposit the amount
back to the customers’ account from the merchant account.

The current system architecture is shown below in Figure 9

Figure 9: Current System Design
4. OUR CONTRIBUTION

Our contribution has been the development the merchant module.

Previous work had a lot of overlap in this module. We designed this module keeping the new structure in mind and removing all the overlap.

Below is the description of the Merchant module:

The screen below in Figure 10 shows the home page for the E-ShoppingMall.com (simulation). This page contains the login menu for the existing users. If the user is new then there is a link to register for this website.

![Login screen to the merchant website.](image-url)

**Figure 10: Login screen to the merchant website.**
The next screen, in Figure 11, shows the registration window for a new user. He has to enter all his details like User ID, password, personal information and address details.

![Figure 11: Registration Window](image)

Figure 12 shows the catalog for shopping from the E-ShoppingMall.com. Currently it contains computer accessories like CPUs, memory, hard discs and motherboards.

![Figure 12: Catalog screen](image)
Next screen in Figure 13 shows all the products in a particular category. The prices of the products are also listed. It also gives an option as to what quantity to select and then adds the selection to the cart. We give the quantity option as integer values only to avoid errors. This was due the feedback we received when we gave a demonstration of the project.

Figure 13: Products in a particular category
Figure 14 shows how the shopping cart looks like after an item has been added to a cart. It shows the product details, unit price and quantity. It also shows the grand total including the tax. After that it gives two options, either to go back and shop more or move towards the check out.

Figure 14: Shopping Cart
The next screen in Figure 15 shows the order history of the particular user that is logged in. It shows the list of all the products he has purchased upto this date with their respective amounts and order numbers. It then gives an option to view order details and payment details.

Figure 15: Order History

Figure 16 shows the order details of a particular order with the product description, price, quantity and shipping status.
Figure 16: Order Details

Next screen (Figure 17) shows the payment details that includes card types, card numbers, card expiration dates and the total amount on a particular order.

Figure 17: Payment Details

Next is the check out screen. It shows the details of the current order, like product description, unit price, quantity and the final payable amount. Then it gives the various payment options like Visa card, Mastercard, Discover, Diners, Debit or the Virtual credit card. So, the customer can pay directly from one of the cards if he has enough balance on a single card or else our system gives him an additional feature to use the Virtual credit card system and create a virtual card using various cards keeping his preferences in mind. This decision is taken by our system that uses fuzzy logic. This is the point where the
bank module comes into picture. The payment has to go to the bank. Or else, when you click on the virtual card, a new window pops up that goes to the virtual card payment agent. Simultaneously, two windows remain open so that when the virtual card number is generated that number can be used to make a payment in the merchant window. One can always come back to this window to complete the transaction. This scenario is depicted in Figure 18, shown below.

Figure 18: Payment Method
Figure 19 shows the process flow from the merchant website to the payment agent when payment agent is called to create a virtual credit card.
5. MERCHANDISE RETURN

A return option is provided on the main page after the customer logs into the merchant website. There are two further choices – Request a Return Reference Number and Check Return History shown in Figure 20 – after the customer clicks on the return option from the top main menu.

![Image of Merchandise Return](image.png)

**Figure 20: Merchandise Return**

For the ease of the customer there is a help page added to this menu which explains to the user the correct procedure for a merchandise return. It explains all the cases when there is a return possible.
The help page is shown in Figure 21.

![Help Menu](image)

**Figure 21: Help Menu**

5.1 Request a Return Reference Number

The “Request a return reference number” link will list all of orders which have been already placed and meet the condition of requesting a return. The table contains records on Order Number, Payment Method, Card Number, Amount, Order Date, and Action shown in Figure 22.
Figure 22: Return Request

A return reference number will be generated sequentially and sent back to the front-end interface from the database for the order which the customer selected to be returned to the merchant, shown in Figure 23. At this point, no refund is issued, only a return reference number is provided until all of the products are shipped back to the merchant along with that number. The customer is responsible to ship all products related to the order back to the merchant with the return reference number.
Once the merchant receives the returned products, an inspection of the products will be performed in order to determine whether a full or partial refund will be issued to the customer, depending on the product conditions and the return reference number.

5.2 Check Return History

The check return history enumerates all orders which have been requested to be returned by the customer (Figure 24). It includes both current and past requests. The table is generated with the records on Return Reference Number, Amount, Request Date, and Status.
There are four possible statuses – Pending, Processing, Approved and Failed. ‘Pending’ indicates that the customer has already requested a return and the merchant did not yet receive the products from the customer. ‘Processing’ implies that the products have been received by the merchant and an inspection is being performed on the returned products. ‘Approved’ informs the customer that the returned products have been accepted and a refund has been issued depending on the method through which the customer paid. ‘Failed’ is a sign that the customer should call the customer service center of the merchant for further assistance. No refund will be issued with seeing ‘Failed’ in the status column. The refund can happen only in the condition that the word Approved is presented to the customer. All other statuses do not guarantee a final refund decision.
5.3 Database Usage on Merchandise Return Module

This module involves PAYMENTINFO, ORDERHISTORY, RETURNHISTORY and a sequential number generator RETURN_ID_SEQ. The way to locate a record among them primarily uses the fields on a USERID, ORDERID and RETURNID.

The function DISTINCT() is used to eliminate all ORDERIDs which belong to the same order with different products. ORDER BY is used to list records in an ascending order.

TO_CHAR() is used to format the output of the date in an mm/dd/yyyy form.

CREATE TABLE PAYMENTINFO ( 
  USERID VARCHAR(20), 
  ORDERNUMBER NUMBER, 
  PAYMETHOD VARCHAR(20), 
  CARDNUMBER VARCHAR(20), 
  EXPIREDATE VARCHAR(10), 
  AMOUNT NUMBER(10, 2), 
  PDATE DATE 
);

CREATE TABLE ORDERHISTORY ( 
  OID NUMBER, 
  PID NUMBER, 
  QTY NUMBER, 
  SHIPSTATUS NUMBER, 
  ODATE DATE 
);

CREATE TABLE RETURNHISTORY ( 
  OID NUMBER, 
  PAYMETHOD VARCHAR(20), 
  CARDNUMBER VARCHAR(20), 
  REQUESTDATE DATE, 
  REFID NUMBER, 
  STATUS VARCHAR(20) 
);

CREATE SEQUENCE RETURN_ID_SEQ
MINVALUE 100
START WITH 100
INCREMENT BY 1;

5.4 Problems Encountered

With thinking more deeply and working on the merchant module, I found that there should be, for the merchant customer services, an interface through which the merchant can retrieve the information on the customers, like payment method and card number, and update the status of a return request. Since the customer must not be able to do a refund for himself, an additional interface for the Virtual Agent will be necessary for all of the merchants who need to do refunds to customers. The Virtual Agent will process and provide refunds to the customer, whenever the merchant accesses the interface with providing the card number and the amount to which a refund will go.
6. DATABASE CONNECTIVITY

The Oracle 9i release 2 is used to store the data in this project. The overall database design obeys the database management theory. To access and retrieve data, a Java Database Connectivity (JDBC) and JavaBean mechanisms are used to avoid security issues which could be caused by entering the database from a JSP file directly. A JavaBean wraps the data from the database and is passed to a JSP file which can access the data inside the bean using the interfaces from the bean.

The connection to the database is created by a compiled Java class file which contains the information required to access the database, like username, password and connection identifier. This Java source file is implemented using the JDBC interfaces. A JSP file which wants to access the database needs to create an object as an instance of the connection class. The connection class provides methods to open a connection to the database, send queries, return result sets, and close the connection with the database. It also handles errors and exceptions which may happen during the execution time.

The connection class file has to be stored in the directory under “WEB-INF/classes” which is restricted by the Tomcat HTTP server. A database driver is loaded when a connection is needed by a JSP file. To program in JSP is similar to core Java. Once a connection is created, a user can send queries to the database and retrieve data from the database.
7. EQUIPMENT CONFIGURATION

The Front-end of the project is done using JSP and HTML. The Back-end, which is the database of the project, is done using Oracle. This system was tested on a Pentium M (1.6 GHz) machine under Windows XP professional. We used to test the system on the NJIT Tomcat server and also we had Tomcat installed on the local machine. Theoretically, any machine that has a fast Internet connect, a JRE (Java Runtime Environment) 1.5 or above (comes with major operating systems), and sufficient memory should be able to test this project. For the end user, all that is required is an Internet connection and a Web browser.

This project has been tested on the following machine configurations:

**Machine Configuration**

- Pentium M – 1.6GHz, Win XP, 512MB RAM
- High Speed Internet
- Tomcat server
- Java JRE 1.5 or above
- Internet Explorer 6.
- Windows XP Professional.
8. CONCLUSION

We created the Merchant module to remove the redundancy. We also made certain changes to the existing system to fit to the new structure we proposed. We added some security features like sessions, access restrictions and automatic logout user that will allow only the valid users to log into the system. Also the user won’t be able to open any Web page in the system unless he has logged in first. We have built the actual database for the system. The user’s primary information is stored in the Database of the system so that the user doesn’t have to enter his primary information again when he logs into the system. We decided to keep the important details of the customers in the database of the payment agent to keep these details secured. The other relevant details are also stored in the System’s Database.

We succeeded in removing the all the overlap that existed in previous modules. Also, we made the GUls of the modules look dissimilar. Another thing we achieved is that all the modules are now independent of each other though the whole system is integrated. Once the bank module is completely developed the system will be complete and ready for deployment.
9. FUTURE WORK

There is an immediate need to create a simulation for the bank to make this system complete. The bank simulation should calculate and update balances on credit cards of the customers. The bank module will also be used to settle the merchant and the customer accounts after the transactions. After the merchandise return the bank will need to credit the money back to the customer account from the merchant account. Thus, developing the bank module is the prime future work. Some other improvements can be made are:

- The current system has customer registration in the payment agent system. But we also need the merchant to be registered with the payment agent to settle all his transactions and keep a track of transactions made by his customer.

- It is possible to work on improving the front end of the merchant site by adding new functionality to it and make it more attractive and marketable as currently it is too simple. For example, the user should be logged out automatically if he is inactive for a certain amount of time on the Payment agent module.

- One can enhance the system by adding additional features like Cookies, Sessions etc. to the Payment Agent module. This will enhance the system and we would be able to keep track of the customers visiting the site. Since the Payment Agent will keep all the credit card details and other secret information of the customers, it needs high quality security.

- A notification email should be sent to the customers about their purchase.

- There should be a specific logic to create a Virtual card number that is 16 digits long following a distinct format that is acceptable to banks.

- The system should prompt the user to print his order details and invoice.
10. REFERENCES


APPENDIX A: ALLOCATION OF WORK:

Since the project was large, we decided to do the work together and both of us worked together as a team in accomplishing this project. Nitesh Phatnani was mainly responsible for proposing and presenting the new structure of the project where as Yongbin Gao was responsible for designing the merchant module of the system. Yongbin Gao concentrated on the merchant module and Nitesh Phatnani worked on its appearance and Graphical User Interface.

Nitesh Phatnani also worked on some problems that existed on the virtual card agent module. There were some flow errors on the payment agent side that Nitesh Phatnani resolved. He also worked on the aesthetic part of the payment agent side. Yongbin Gao worked on planning of the merchandise return.

Division of work in terms of coding is as below in Table 1:

Table 1:

<table>
<thead>
<tr>
<th>Nitesh Phatnani</th>
<th>Yongbin Gao</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login.jsp</td>
<td>Cart.jsp</td>
</tr>
<tr>
<td>Login Action.jsp</td>
<td>BillDetail.jsp</td>
</tr>
<tr>
<td>LoginFail.jsp</td>
<td>Catalog.jsp</td>
</tr>
<tr>
<td>Logout.jsp</td>
<td>Checkout.jsp</td>
</tr>
<tr>
<td>Registration.jsp</td>
<td>CheckoutAction.jsp</td>
</tr>
<tr>
<td>RegistrationAction.jsp</td>
<td>Details.jsp</td>
</tr>
<tr>
<td>TopMenu.jsp</td>
<td>History.jsp</td>
</tr>
<tr>
<td>Style.css</td>
<td>MyAccount.jsp</td>
</tr>
<tr>
<td>Table Structures</td>
<td>PurchaseAction.jsp</td>
</tr>
<tr>
<td>Background Structure</td>
<td>PurchaseAmount.jsp</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>Return.jsp</td>
</tr>
<tr>
<td>Menu Styles</td>
<td>ReturnAction.jsp</td>
</tr>
<tr>
<td>Inserting Images</td>
<td>ReturnHelp.jsp</td>
</tr>
<tr>
<td>New System Structure</td>
<td>ReturnHistory.jsp</td>
</tr>
<tr>
<td>Minor corrections on Payment Agent</td>
<td>ReturnRequest.jsp</td>
</tr>
</tbody>
</table>
The division of work for report writing is shown below:

1. INTRODUCTION - Nitesh Phatnani

2. OVERVIEW – Nitesh Phatnani

3. PREVIOUS WORK – Yongbin Gao

4. CURRENT SYSTEM - Yongbin Gao

5. OUR CONTRIBUTION (MERCHANT WEBSITE) - Nitesh Phatnani, Yongbin Gao

6. MERCHANDISE RETURN - Yongbin Gao

7. EQUIPMENT CONFIGURATION - Nitesh Phatnani, Yongbin Gao

8. CONCLUSION - Yongbin Gao

9. FUTURE WORK – Nitesh Phatnani

10. REFERENCES – Nitesh Phatnani
APPENDIX B: USER MANUAL

All Java Server Page (JSP) files about merchant site are stored in AFS system and the full path to those files is “/afs/cad/research/p/2/Creditcard/Shreeshah/credit_card/yg8/jsp”. The source Java files are stored under “/afs/cad/research/p/2/Creditcard/Shreeshah/credit_card/yg8/src”. The virtual agent interface JSP files are stored under AFS system with the access path “afs/cad.njit.edu/research/p/2/Creditcard/fuzzy_opt/nkp3”.

A NJIT user ID and password are needed to get a full access to the above directories. The database used to build the system is Oracle 9i release 2 which can be accessed from “http://seer.njit.edu/isqlplus”. The username, password and connection identifier all have to be specified in order to gain full access to the database.

The system is tested on Tomcat HTTP server. There are two crucial directories under WEB-INF, classes and lib. The directory classes is the place at which all Java Servlet files and JavaBean class files stay. All Java JAR files and compressed package class files are located in the directory lib. Tomcat is very restricted in the file structure hierarchy which means each kind of files have to be put into the corresponding directories. Otherwise, the Tomcat HTTP server can not find the related files to response the client’s request. Therefore, an additional caution may be needed in deploying files among directories.

The JSP file is compiled by the Tomcat server when the server receives a request toward that file. The file is actually converted into a Java Servlet first by the Tomcat server without user’s consciousness and then rendered to the user for viewing. As a result, the
front users can not see the JSP source files from a web browser. What they will see is just HTML files returned by Tomcat server.

To run a JSP file, users need to specify the target file by its name in the URL along with the directory hierarchy from a web browser. After the server receives the request, the server will convert the target JSP file into a servlet file and then compile the servlet file into a class file. The compiled servlet file will be again converted into an HTML file when it is returned to the web browser.

The following figure 25 gives a visual illustration on how a Tomcat server works.

![Diagram of Web Browser and Tomcat Server](image)

**Figure 25: Web Browser and Tomcat Server**
APPENDIX C: SOURCE CODE

billdetail.jsp

```html
<html>
<%@page import='java.sql.*,java.text.*,common.*'%>
<head><title>Payment Details</title></head>
<body BACKGROUND= "yellow_paper.jpg">
<% if (session.getAttribute("userID") == null){
    response.sendRedirect("Login.jsp");
} %>
<jsp:include page="topmenu.jsp" />
<% DBConnection myConn = null;
    myConn = new DBConnection();
    Connection conn = myConn.getConnection();
    Statement stmt = null;
    ResultSet rs = null;
    String orderid = request.getParameter("orderid");
    // ResultSet rs2 = null;
    // Statement stmt2 = null;
    // NumberFormat nf = NumberFormat.getNumberInstance();
    // nf.setMaximumFractionDigits(2);
    // nf.setMinimumFractionDigits(2);
%>
<br><br><center>
<TABLE>
    <TR>
    <TH><font size="6">Payment Details</font></TH>
</TR>
    <TR><TH><br></TH></TR>
    <TR>
    <TH><font size="5">Order Number: <%=orderid%></font></TH>
</TR>
</TABLE></center>
<table BACKGROUND= "bg.jpg" cellspacing=1 cellpadding=2 width=100%>
    <TR>
    <TH bgcolor="#333399"><font color="ffffff"> Pay Method</font></TH>
    <TH bgcolor="#333399"><font color="ffffff">Description</font></TH>
    <TH bgcolor="#333399"><font color="ffffff">Expiration Date</font></TH>
    <TH bgcolor="#333399"><font color="ffffff">Amount</font></TH>
</TR>
<% try {
    stmt = conn.createStatement();
    rs = stmt.executeQuery("select * " +
        "from paymentinfo " +
        "where ordernumber='" +
        orderid + "'");
    while(rs.next()) { %>
    <TR>
```
```
<TD valign=top align="left"><font size="3"><%=rs.getString(3)%></font></TD>
<TD valign=top align="left"><font size="3"><%=rs.getString(4)%></font></TD>
<TD valign=top align="left"><font size="3"><%=rs.getString(5)%></font></TD>
<TD valign=top align="left"><font size="3">$<%=rs.getString(6)%></font></TD>
</TR></table>
<%     } %>
</%>
rs.close() ;
stmt.close() ;
try {
    catch(Exception ex2){

    }
</catch>
</%>
</body></html>
nf.setMinimumFractionDigits(2);

String pid = request.getParameter("pid");
String act = request.getParameter("act");
String qty = request.getParameter("qty");
if (act != null) {
    if (act.equals("ADD"))
        order.addOrderItem(pid, qty);
    else if (act.equals("REMOVE"))
        order.removeOrderItem(pid);
}%>

<BR><BR>
<form action="cart.jsp">
<table width=100% background="bg.jpg">
<tr>
<th><font size="6">Shopping Cart</font></th>
</tr>
<tr>
<td>
<table cellspacing=1 cellpadding=2 width=100%>
<tr>
<th bgcolor="#333399"><font color="ffffff">Product</font></th>
<th bgcolor="#333399" width="100"><font color="ffffff">Unit Price</font></th>
<th bgcolor="#333399"><font color="ffffff">Quantity</font></th>
<th bgcolor="#333399" width="100"><font color="ffffff">Price</font></th>
</tr>
<%try{
    Hashtable orderItems = order.getOrder();
    Enumeration keys = orderItems.keys();
    float total = 0;
    while (keys.hasMoreElements()) {
        pid = (String) keys.nextElement();
        qty = (String) orderItems.get(pid);
        stmt = conn.createStatement();
        rs = stmt.executeQuery("SELECT p.products_id, p.products_name, p.products_description, p.products_price FROM products p, products_description pd WHERE p.products_id = pd.products_id AND p.products_id = "+pid+" order by products_id";
        if (rs.next()) {
            <TR>
            <input type=hidden name="pid" value="<%= rs.getString(1) %>">
            <input type=hidden name="act" value="REMOVE">
            <TD valign=top><STRONG><%= rs.getString(2) %></STRONG><BR><%= rs.getString(3) %>
        String price = rs.getString(4);
float stotal = 
Integer.parseInt(qty)*Float.parseFloat(price);
total += stotal;

<TD valign=top align=left>$<%=price%></TD>
<TD valign=top align=left><%=qty%></TD>
<input type=submit value="Remove from Cart">
<TD valign=top align=right>$<%=nf.format(stotal)%></TD>
</TR>

try{
    rs.close() ;
    stmt.close() ;
}catch(Exception ex2){}

<% if(total > 0) { %>
    <TR>
        <TD align=right colspan=4>&nbsp;</TD>
    </TR>
    <TR>
        <TD align=right colspan=3><STRONG>Total Amount</STRONG></TD>
        <TD align=right>$<%=nf.format(total)%></TD>
    </TR>
    <TR>
        <TD align=right colspan=3><STRONG>Tax (6%)</STRONG></TD>
        <TD align=right>$<%=nf.format(total*0.06)%></TD>
    </TR>
    <TR>
        <TD align=right colspan=3><STRONG>Grand Total</STRONG></TD>
        <TD align=right>$<%=nf.format(total+total*0.06)%></TD>
    </TR>
<% } else { %>
    <TR><TD align="center"><BR><BR><strong><font size="3">No items in the cart! Please shop from <a href="catalog.jsp">Catalog</a>.</font></strong> </TD></TR>
<% } %>
</TABLE>
catalog.jsp
<html>
<%@ page import='java.sql.*,common.*'%>
<head><title>Catalog</title></head>
<body BACKGROUND= "yellow_paper.jpg">

```jsp
if (total > 0) {
    a href="catalog.jsp"Continue Shopping
    a href="checkout.jsp"Checkout
}
```

```jsp
catch (Exception ex) {
    Error: <%=ex.getMessage()%>
}
```

</body></html>
<jsp:include page="topmenu.jsp" />
<br><br>

<TABLE width=100% BACKGROUND= "bg.jpg">
  <TR>
    <TH><font size="6">Catalog</font></TH>
  </TR>
  <tr><td bgcolor="#333399"><font color="ffffff"size="5">Catalog List:</font></td></tr>
<%>
  DBConnection myConn = null;
  ResultSet rs = null;
  myConn = new DBConnection();
  Connection conn = myConn.getConnection();
  Statement stmt = conn.createStatement();
  rs = stmt.executeQuery("SELECT categories_id,categories_name FROM categories order by categories_id");
  while ( rs.next() ) {
    String url = "category.jsp?cid=" + rs.getString(1);
    %>
    <TR>
      <TD width=50% valign="top">&gt;&nbsp;<A href="" + rs.getString(2) + "><font size="5">" + rs.getString(2) + ">
    </TR>
  <%
  } %>
</TABLE>
<%
  rs.close() ;
  stmt.close() ;
%>
</body></html>

**Category.jsp**

```html
<html>
<%@ page import='java.sql.*,common.*'%>
<head><title>Category</title></head>
<body BACKGROUND= "yellow_paper.jpg">
<jsp:include page="topmenu.jsp" />
<%
  String cid = request.getParameter("cid");
  DBConnection myConn = null;
  ResultSet rs = null;
  myConn = new DBConnection();
  Connection conn = myConn.getConnection();
  Statement stmt = conn.createStatement();
  rs = stmt.executeQuery("SELECT categories_name FROM categories where categories_id="+cid);%
>
<br><br>
```
```html
<TABLE width=100% BACKGROUND="bg.jpg">
  <TR>
    <TH><font size="6">Products in</font></TH>
  </TR>
  <% if (rs.next()) { %>
    <%=rs.getString(1)%>
  <% } %>
  </TR>
</TABLE>

<% try { %>
  rs.close();
  stmt.close();
<% } catch (Exception x) {} %>

<% try { %>
  stmt = conn.createStatement();
  rs = stmt.executeQuery("SELECT p.products_id," +
    "pd.products_name," +
    "pd.products_description," +
    "p.products_price " +
    "FROM products p," +
    "products_description pd," +
    "products_to_categories pc" +
    "where pc.products_id=p.products_id and " +
    "pc.categories_id=" + cid +
    " order by products_id") ;
  while ( rs.next() ) { %>
    <TR><FORM action="cart.jsp">
      <input type=hidden name="pid" value="<%=rs.getString(1)%>">
      <input type=hidden name="act" value="ADD">
      <TD valign=top><STRONG><%=rs.getString(2)%></STRONG><BR><%=rs.getString(3)%></TD>
      <TD valign=top align=left>$<%=rs.getString(4)%></TD>
    </FORM></TR>
  <% } %>
</%>
```
checkout.jsp

<html>
<%@page import='java.sql.*,java.text.*,common.*,java.util.*'%>
<head><title>Checkout</title></head>
<body BACKGROUND= "yellow_paper.jpg">
<% if (session.getAttribute("userID") == null){
    response.sendRedirect( "Login.jsp");
} %>
<jsp:useBean id="order" scope="session" class="common.Order" />
<jsp:include page="topmenu.jsp" />
<% DBConnection myConn = null;
    ResultSet rs = null;
    myConn = new DBConnection();
    Connection conn = myConn.getConnection();
    Statement stmt = null;
    NumberFormat nf = NumberFormat.getNumberInstance();
    nf.setMaximumFractionDigits(2);
    nf.setMinimumFractionDigits(2);
    String pid = "";
    String qty = "";
    float gtotal = 0;
    float total = 0;
%>
<TABLE width=100% BACKGROUND= "bg.jpg">
<TR><TH><font size="6">Check Out</font></TH></TR></table>
<TABLE cellspacing=1 cellpadding=2 width=100%>
<TR>
<TH bgcolor="#333399"><font color="ffffff">Product</font></TH>
<TH bgcolor="#333399" width="100"><font color="#ff0000">Unit Price</font></TH>
</TR>
</TABLE>
</body></html>
<TH bgcolor="#333399"><font color="ffffff">Quantity</font></TH>
<TH bgcolor="#333399" width="100"><font color="ffffff">Price</font></TH>
</TR>
<% try {
    Hashtable orderItems = order.getOrder();
    Enumeration keys = orderItems.keys();
    while (keys.hasMoreElements()) {
        pid=(String)keys.nextElement();
        qty=(String)orderItems.get(pid);
        stmt = conn.createStatement();
        rs = stmt.executeQuery("SELECT
p.products_id,pd.products_name,pd.products_description,p.products_price
FROM products p, products_description pd where
p.products_id=pd.products_id and p.products_id="+pid+" order by
products_id") ;
        while(rs.next()) {
<TR><TD valign=top><STRONG><%=rs.getString(2)%></STRONG><BR><%=rs.getString(3)%></TD>
    String price = rs.getString(4);
    float stotal =
    Integer.parseInt(qty)*Float.parseFloat(price);
    total += stotal;
<% } %>
    <TD valign=top align=right>$<%=price%></TD>
<% String price = rs.getString(4);
    total += stotal;
<% } %>
    <TD valign=top align=right>$<%=total%></TD>
%>
<table>
  <tr>
    <td align=right>$<%=nf.format(total*0.06)%></td>
  </tr>
  <tr>
    <td align=right colspan=3><strong>Grand Total</strong></td>
    <td align=right>$<%=nf.format(total+total*0.06)%></td>
  </tr>
</table>

<% }
else {%>
  <tr><td><br><br><strong>There are no items in the Cart</strong></td></tr>
<% } %>

<!-- <form action="dbtest.jsp?txtPurchaseAmount=<%=gtotal%>" method="post"> -->
<form action="checkoutaction.jsp?txtPurchaseAmount=<%=gtotal%>" method="post">
  <table>
    <tr><td colspan="2"><br></td></tr>
    <tr><td><strong>Payment Method</strong></td>
      <td><select name="paymethod" size="1"
        onchange='if(selectedIndex==3)
        {window.open("http://web.njit.edu/fuzzy_opt/Index1.jsp");}'>
        <option>Select One...</option>
        <option>MasterCard</option>
        <option>Visa</option>
        <option>VirtualCard</option>
        </select></td></tr>
    <tr><td><strong>Card Number</strong></td>
      <td><input type="text" name="cardnumber"
        maxlength="16" size="14"></td></tr>
    <tr><td><strong>Expiration Date</strong></td>
      <td>
        <select name="expmon" size="1">
          <option>Month</option>
        </select>
        <select name="expyr" size="1">
          <option>Year</option>
        </select>
        <% for (int k=2000; k<=2020; k++) { %>
          <option><%= k %></option>
        <% } %></select>
      </td></tr>
    <tr><td><strong>Amount Due</strong></td>
      <td><%= "$" + nf.format(total+total*0.06) %></td></tr>
    <tr><td colspan="2"><br></td></tr>
    <tr><td><input type="submit" Value="Submit"></td>
      <td><input type="reset" value="Reset"></td></tr>
  </table>
</form>
checkoutaction.jsp
<html>
<%@ page language="java" import="java.sql.*,common.*,java.util.*"%>
<jsp:useBean id="order" scope="session" class="common.Order" />
<head>
<title>Check Out Action</title>
</head>
<body BACKGROUND= "yellow_paper.jpg">
<jsp:include page="topmenu.jsp" />
<%>
Hashtable cart = order.getOrder();
String usr = (String)session.getAttribute("userID");
String amt = request.getParameter("txtPurchaseAmount");
String method = request.getParameter("paymethod");
String cardNum = request.getParameter("cardnumber");
String expMo = request.getParameter("expmon");
String expYr = request.getParameter("expyr");
DBConnection myConn = new DBConnection();
Connection conn = myConn.getConnection();
Statement stmt = null;
ResultSet rs = null;
String sqlQuery = null;
String expiryDate = expMo + "/" + expYr;
</%>
```java
float purchaseAmt = 0;
int purchaseId = 0;
int rowsUpdated = 0;

try{
    stmt = conn.createStatement();
    if(amt != null){
        purchaseAmt = Float.parseFloat(amt);
    }
    sqlQuery = "SELECT PURCHASEID_SEQ.NEXTVAL PURCHASE_ID FROM DUAL";
    rs = stmt.executeQuery(sqlQuery);
    while(rs.next()){
        purchaseId = rs.getInt("PURCHASE_ID");
    }
    rs.close();
    sqlQuery = "INSERT INTO paymentinfo VALUES('" + usr + "," + purchaseId + "," + method + "," + cardNum + "," + expiryDate + "," + purchaseAmt + ",sysdate,sysdate)";
    rowsUpdated = stmt.executeUpdate(sqlQuery);
    Enumeration keys = cart.keys();
    while(keys.hasMoreElements()) {
        String charProid = (String)(keys.nextElement());
        String charQty = (String)(cart.get(charProid));
        int pid = Integer.parseInt(charProid);
        int qty = Integer.parseInt(charQty);
        rowsUpdated = stmt.executeUpdate("INSERT INTO orderhistory VALUES(" + purchaseId + "," + pid + "," + qty + "," + 0 + "," + sysdate")");
    }
    cart.clear();
}
catch(Exception ex){ }
stmt.close();
myConn.closeConnection();
%
<BR><BR>
<center><font size="6">Check Out Successfully</font><BR><BR>
<h3>Your Order Number is <%=purchaseId%></h3>
<BR><font size="3">We appreciate your business.</font><BR><BR>
<font size="3">Please go to <a href="history.jsp">Order History</a> to review your order history.</font><BR><BR>
<h5><%= cart.size() %&gt; item(s) left in the cart.</h5>
</center>
</body>
</html>
```
details.jsp

<html>
<%@page import='java.sql.*,java.text.*,common.*'%>
<head><title>Order Details</title></head>
<body BACKGROUND= "yellow_paper.jpg">

<jsp:include page="topmenu.jsp" />

<% DBConnection myConn = null;
    myConn = new DBConnection();
    Connection conn = myConn.getConnection();
    Statement stmt = null;
    ResultSet rs = null;
    String orderid = request.getParameter("orderid");
    // ResultSet rs2 = null;
    // Statement stmt2 = null;
    // NumberFormat nf = NumberFormat.getNumberInstance();
    // nf.setMaximumFractionDigits(2);
    // nf.setMinimumFractionDigits(2);
%>

<BR><BR><center>
<TABLE>
<TR><TH><font size="6">Order Details</font></TH></TR>
<TR><TH><BR></TH></TR>
<TR>
<TH><font size="5">Order Number: <%=orderid%></font></TH>
</TR>
</TABLE></center>

<TABLE BACKGROUND= "bg.jpg" cellspacing=1 cellpadding=2 width=100%>
<TR>
    <TH bgcolor="#333399"><font color="ffffff">Product</font></TH>
    <TH bgcolor="#333399"><font color="ffffff">Description</font></TH>
    <TH bgcolor="#333399"><font color="ffffff">Unit Price</font></TH>
</TR>
</table>
<TH bgcolor="#333399"><font color="ffffff">Quantity</font></TH>
<TH bgcolor="#333399"><font color="ffffff">Ship Status</font></TH>

<%try{
    stmt = conn.createStatement();
    rs = stmt.executeQuery("select * "+
        "from orderhistory oh, "+
        "products p, "+
        "products_description pd "+
        "where oh.oid='" + orderid+
        "' and " +
        "oh.pid=p.products_id and " +
        "p.products_id=pd.products_id");
    while(rs.next()) { %>
        <TR>
            <TD valign=top align="left"><font size="3"><%=rs.getString(20)%></font></TD>
            <TD valign=top align="left"><font size="3"><%=rs.getString(21)%></font></TD>
            <TD valign=top align="left"><font size="3">$<%=rs.getString(10)%></font></TD>
            <TD valign=top align="left"><font size="3"><%=rs.getString(3)%></font></TD>
            <TD valign=top align="left"><font size="3"><%
            if((rs.getString(4)).equalsIgnoreCase("y")){
            <%="Yes"%>
            } else {
            <%="No"%>
            }%
        </font></TD>
        </TR>
    <%} %>
</TR></table>
<%rs.close() ;
stmt.close() ;}
</%>

rs.close() ;
stmt.close() ;
} catch(Exception ex2){
%
</body></html>

history.jsp
<html>
<%@page import='java.sql.*,java.text.*,common.*,java.util.*'%>
<head><title>History</title></head>
<body  BACKGROUND= "yellow_paper.jpg">
<% if (session.getAttribute("userID") == null){
            response.sendRedirect("Login.jsp");
} %>
<jsp:include page="topmenu.jsp" />
<%
DBConnection myConn = null;
ResultSet rs = null;
myConn = new DBConnection();
Connection conn = myConn.getConnection();
Statement stmt = null;
String usr = (String)session.getAttribute("userID");
%>
<br>
<BR><BR>
<TABLE width=100% BACKGROUND= "bg.jpg">
<TR>
<TH><font size="6">Order History</font></TH>
</TR>
<TR>
<td>
<TABLE cellspacing=1 cellpadding=2 width=100%>
<TR>
<TH bgcolor="#333399"><font color="ffffff">Order Number</font></TH>
<TH bgcolor="#333399"><font color="ffffff">Date</font></TH>
<TH bgcolor="#333399"><font color="ffffff">Amount</font></TH>
<TH bgcolor="#333399"><font color="ffffff">&nbsp;</font></TH>
</TR>
<%
try{
    stmt = conn.createStatement();
    rs = stmt.executeQuery("select ordernumber, TO_CHAR(pdate, 'mm/dd/yyyy'), amount " +
                           "from paymentinfo " +
                           "where userid='" + usr +
                           "' order by ordernumber");
    if(rs.next()) {
        String orderid=rs.getString(1);
        String ordertime=rs.getString(2);
        String orderamount=rs.getString(3);
        <TR>
        <TD valign=top align="left"><STRONG><font size="3">orderid</font></STRONG></TD>
        <TD valign=top align="left"><font size="3">ordertime</font></TD>
        <TD valign=top align="left"><font size="3">orderamount</font></TD>
        <TD valign=top align="left"><a href="details.jsp?orderid="+orderid">Order Details</a>&nbsp;&nbsp;&nbsp;
    </TR>
    }
<%}
</TR>
</TABLE>
</td>
</TR>
</TABLE>
</body>
<a href="billdetail.jsp?orderid=<%=orderid%>">Payment Details</a></font></TD>
</TR>
<% while(rs.next()) {
  orderid=rs.getString(1);
  ordertime=rs.getString(2);
  orderamount=rs.getString(3);
%>
<tr>
  <TD valign=top align="left"><STRONG><font size="3"><%=orderid%></font></STRONG></TD>
  <TD valign=top align="left"><font size="3"><%=ordertime%></font></TD>
  <TD valign=top align="left"><font size="3">$<%=orderamount%></font></TD>
  <TD valign=top align="left"><font size="3"><a href="details.jsp?orderid=<%=orderid%>">Order Details</a> &nbsp;&nbsp;&nbsp;<a href="billdetail.jsp?orderid=<%=orderid%>">Payment Details</a></font></TD>
</TR>
<% } %>
</TABLE>
</TD></TR>
</TABLE>
<%} catch(Exception ex) { %>
Error: <%=ex.getMessage()%>
<%} %>
</body></html>
login.jsp

```html
<html>
<head><title>Login Page</title></head>
<body BACKGROUND= "yellow_paper.jpg" >
<jsp:include page="<%=response.encodeURL("topmenu.jsp")%>" />
<% if (((String)session.getAttribute("status"))!=null) { %>
  <% if (((String)session.getAttribute("status")).equals("loggedin")) { %>
    <jsp:forward page='<%=response.encodeURL("catalog.jsp")%>' />
  <% } %>
<% } %>
<center>
<form name="loginForm" method="post"
action='<%=response.encodeURL("LoginAction.jsp")%>'>
<table BACKGROUND= "bg.jpg" width="35%" border="0" align="center"
cellpadding="2" cellspacing="2">
  <tr><td colspan="2" width="50%"></td></tr>
  <tr><td colspan="2" width="50%"></td></tr>
  <p>
    <td align="center" size="3"><b>Please Login to Shop</b></td>
  </p>
  <tr>
    <td align="right"><font color="#000000" size="2" face="Arial, Helvetica">User ID:</font></td>
    <td><input name="usrname" type="text" maxlength="25" value="test" size="20"></td>
  </tr>
  <tr>
    <td align="right"><font color="#000000" size="2" face="Arial, Helvetica">Password:</font></td>
    <td><input name="passwd" type="password" maxlength="25" value="test" size="22"></td>
  </tr>
</table>
</form>
</center>
</body>
</html>
```
<table align="center">
<tr>
<td><input value="Login" type="submit" size="30"></td>
<td><input type="reset" value="Reset" size="30"></td>
</tr></table>
<br>
<table>
<tr><td align="right"><font color="#000000" size="2" face="Arial, Helvetica">Please click</font></td></tr>
<tr><td><a href='<%=response.encodeURL("Registration.jsp")%>here</a> to register.</td></tr>
</table>
</center>
</html>

LoginAction.jsp
<html>
<%@ page language = "java" import = "java.sql.*,common.*"%>
<head><title>Login Action</title></head>
<body>
<%  DBConnection myConn = null;
   ResultSet rs = null;
   String sqlQuery = null;
   boolean loginflag = false;
   String usr = request.getParameter( "usrname");
   String pwd = request.getParameter( "passwd");
   String cusName = null;
   try{
      myConn = new DBConnection();
      Connection conn = myConn.getConnection();
      Statement stmt = conn.createStatement();
      if((!usr.equals("") && (!pwd.equals(""))) {  
         sqlQuery = "SELECT COUNT(*) FROM USERS WHERE USER_ID = " +
                   usr + "' AND PASSWD = " +
                   pwd + ";";
         rs = stmt.executeQuery(sqlQuery);
         if(rs.next()) {
            loginflag = true;
            session.setAttribute("userID", usr);
            session.setAttribute("status", "loggedin");
         } else {
            loginflag = false;
         }
      }
      sqlQuery = "select first_name, last_name from users where user_id=":" +
               usr + ";";
      rs = stmt.executeQuery(sqlQuery);
      if(rs.next()) {
         cusName = rs.getString(1) + " " + rs.getString(2);
         session.setAttribute("RealName", cusName);
      }
      rs.close();
      stmt.close();
      myConn.closeConnection();
   } catch(Exception ex){  }
   if(loginflag) {  <%}
      <jsp:forward page="catalog.jsp" />
   %>
   else {  <%}
      <jsp:forward page="loginfail.jsp" />
   %>
</body>
</html>
loginfail.jsp

<html><%@ page language = "java" import = "java.sql.*,common.*"%>
<head><title>Login Page</title>
<link rel="stylesheet" href="style.css" type="text/css"></head>
<body BACKGROUND= "yellow_paper.jpg">
<jsp:include page="topmenu.jsp" />
<center><BR><BR><BR>
<font size="3" color="red">Login failed, please try again.</font>
<form name="loginForm" method="post" action="LoginAction.jsp">
<table width="35%" border="0" align="center" cellpadding="2"
cellspacing="2">
<tr><td colspan="2" width="50%"></td></tr>
<tr>
<td align="right"><font color="#000000" size="2" face="Arial, Helvetica">User ID:</b></font></td>
<td><input name="usrname" type="text" maxlength="25" value="test" size="20"></td>
</tr>
<tr>
<td align="right"><font color="#000000" size="2" face="Arial, Helvetica">Password:</b></font></td>
<td><input name="passwd" type="password" maxlength="25" value="test" size="22"></td>
</tr>
</table>
</form>
</center>
</body>
</html>
Logout.jsp

```html
<html>
<%@ page language = "java" import = "java.sql.*,common.*" %>
<head><title>Logout</title>
<link rel="stylesheet" href="style.css" type="text/css">
<body BACKGROUND= "yellow_paper.jpg">
<% session.invalidate();%>
<center>
<h2><font FACE="Century Gothic" size="5"><B>E-Shopping Mall.com</B></font><BR><BR></h2>
<TABLE cell-spacing=5 cellPadding=10 width=800 border="1">
<tr>
<td align="middle"><a href="Login.jsp">Home</a></td>
<td align="middle"><a href="catalog.jsp">Catalog</a></td>
<td align="middle"><a href="cart.jsp">My Cart</a></td>
<td align="middle"><a href="Login.jsp">My Account</a></td>
</tr>
</table>
<br><br>
<table width="100%" border="0" align="left" cellpaddning="0" cellspacing="0">
<tr>
<td width="100%" align="center">
<font size="4" face="Arial, Helvetica">Thank you for shopping with us!</font></td>
</tr>
</table>
</center><br><br>
</body></html>
```
myacct.jsp

<html>
<%@ page language = "java" import = "java.sql.*,common.*"%>
<!-- jsp that collects the login details-->
<head><title>My Account</title>
<link rel="stylesheet" href="style.css" type="text/css">
<script language="JavaScript">
    function evalAll() {
        test = true;
        if((document.all) || (document.getElementById)){
            var userID = document.regForm.txtUserID.value;
            var passwd = document.regForm.txtPassword.value;
            var passwd_retype = document.regForm.txtPasswordRetype.value;
            var firstName = document.regForm.txtFirstName.value;
            var lastName = document.regForm.txtLastName.value;
            var homePhone = document.regForm.txtHomePhone.value;
            var city =document.regForm.txtCity.value;
            var streetAddress = document.regForm.txtStreetAddress.value;
            var myIndex = document.regForm.cboState.selectedIndex;
            var selStateValue = document.regForm.cboState.options[myIndex].value;
            var zipCode =document.regForm.txtZipCode.value;
            var cityb =document.regForm.txtbCity.value;
            var streetAddressb = document.regForm.txtbStreetAddress.value;
            var myIndexb = document.regForm.cbobState.selectedIndex;
            var selStateValueb = document.regForm.cbobState.options[myIndexb].value;
            var zipCodeb =document.regForm.txtbZipCode.value;
        }
        if (passwd!=null && passwd_retype!=null) {
            if (passwd!=passwd_retype){
                alert('Password and Retype Password should be same');
                document.regForm.txtPassword.value="";
                document.regForm.txtPasswordRetype.value="";
                document.regForm.txtPassword.focus();
                test = false;
            }
        }
    }
</script>
</head>
<body>
</body>
</html>
if (userID == ""){
    alert("User ID cannot be blank");
document.regForm.txtUserID.focus();
test = false;
return test;
} else if(passwd == ""){
    alert("Password cannot be blank");
document.regForm.txtPassword.focus();
test = false;
return test;
} else if(passwd_retype == ""){
    alert("Password Retype cannot be blank");
document.regForm.txtPasswordRetype.focus();
test = false;
return test;
} else if(firstName == ""){
    alert("First Name cannot be blank");
document.regForm.txtFirstName.focus();
test = false;
return test;
} else if(lastName == ""){
    alert("Last Name cannot be blank");
document.regForm.txtLastName.focus();
test = false;
return test;
} else if(homePhone == ""){
    alert("Home Phone cannot be blank");
document.regForm.txtHomePhone.focus();
test = false;
return test;
} else if(streetAddress == ""){
    alert("Street Address cannot be blank");
document.regForm.txtStreetAddress.focus();
test = false;
return test;
} else if(city == ""){
    alert("City cannot be blank");
document.regForm.txtCity.focus();
test = false;
return test;
} else if(selStateValue == ""){
    alert("State cannot be blank");
document.regForm.cboState.focus();
test = false;
return test;
} else if(zipCode == ""){
    alert("Zip Code cannot be blank");
document.regForm.txtZipCode.focus();
test = false;
return test;
}
return test;

function evalAddress(){
```javascript
var city = document.regForm.txtCity.value;
var streetAddress = document.regForm.txtStreetAddress.value;
var myIndex = document.regForm.cboState.selectedIndex;
var selStateValue = document.regForm.cboState.options[myIndex].value;
var zipCode = document.regForm.txtZipCode.value;

//alert(document.regForm.chkAddrSame.checked);
if (document.regForm.chkAddrSame.checked) {
    document.regForm.txtbCity.value = city;
    document.regForm.txtbStreetAddress.value = streetAddress;
    document.regForm.cbobState.selectedIndex = myIndex;
    document.regForm.cbobState.options[myIndex].value = selStateValue;
    document.regForm.txtbZipCode.value = zipCode;
} else {
    document.regForm.txtbCity.value = "";
    document.regForm.txtbStreetAddress.value = "";
    document.regForm.cbobState.selectedIndex = "";
    document.regForm.cbobState.options[myIndex].value = "";
    document.regForm.txtbZipCode.value = "";
}

function validateInt() {
    var iString = document.regForm.txtZipCode.value
    alert(iString);
    return ("" + parseInt(iString)) == iString;
}

function AllowNumericOnly(){
  Expression = '0123456789';
  var ch = String.fromCharCode(window.event.keyCode);
  ch = ch.toLowerCase();
  var a = Expression.indexOf(ch);
  if (a == -1)
    window.event.keyCode = 0;
}
```
String phone_res = "";
String phone_off = "";
String street = "";
String cty = "";
int stid = 0;
String st = "";
int zip = 0;

// gets the state information
DBConnection myConn = null;
ResultSet rs = null;
String sqlQuery = null;
try{
    myConn = new DBConnection();
    Connection conn = myConn.getConnection();
    Statement stmt = conn.createStatement();
    sqlQuery = "select * from users where user_id='" + usrid + "'";
    rs = stmt.executeQuery(sqlQuery);
    while (rs.next()) {
        pwd = rs.getString("passwd");
        repwd = rs.getString("passwd_retype");
        pfix = rs.getString("prefix");
        fname = rs.getString("first_name");
        lname = rs.getString("last_name");
        mname = rs.getString("middle_name");
        phone_res = rs.getString("phone_num_res");
        phone_off = rs.getString("phone_num_off");
    }
    sqlQuery = "select * from user_address where user_id='" + usrid + "'";
    rs = stmt.executeQuery(sqlQuery);
    while (rs.next()) {
        street = rs.getString("street_address");
        cty = rs.getString("city");
        stid = Integer.parseInt(rs.getString("state_id"));
        zip = Integer.parseInt(rs.getString("zip_code"));
    }
    sqlQuery = "SELECT STATE_DESC FROM STATE where state_id=" + stid;
    rs = stmt.executeQuery(sqlQuery);
    while (rs.next()) {
        st = rs.getString("STATE_DESC");
    }
    rs.close();
    stmt.close();
}
myConn.closeConnection();

catch(Exception ex){ }

%>
<br><br>
</br>
<h1><font color="#08246B" size="5" face="arial,helvetica">Registration Information</font></h1>
<form name="regForm" method="post" action="updateAction.jsp">
<table BACKGROUND= "bg.jpg" border="0" align="left" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF">
<tr>
<td colspan="3" align="left" bgcolor="#333399"> <font color="#FFFFFF" size="2" face="Arial, Helvetica">Login Information:</font></td>
</tr>
</table>
<br><br><br><br><br><br><br><br><br>
<table BACKGROUND= "bg.jpg" border="0" align="left" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF">
<tr>
<td colspan="3" align="justify" bgcolor="#333399">
<font color="#FFFFFF" size="2" face="Arial, Helvetica">Personal Information:</font></td>
</tr>
</table>
<br>
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First Name:</strong></td>
<td> </td>
</tr>
<tr>
<td><strong>Middle Name:</strong></td>
<td> </td>
</tr>
<tr>
<td><strong>Last Name:</strong></td>
<td> </td>
</tr>
</tbody>
</table>
<td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica">Home Phone:</font></td>
<td>&nbsp;</td>
<td><input name="txtHomePhone" type="text" value="<%=phone_res%>" maxlength="15" size="40"></td>
</tr>
<tr>
<td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica">Office Phone:</font></td>
<td>&nbsp;</td>
<td><% if(phone_off != null) { %>
<td><input name="txtOfficePhone" type="text" value="<%=phone_off%>" maxlength="15" size="40"></td>
<% } else { %>
<td><input name="txtOfficePhone" type="text" value="" maxlength="15" size="40"></td>
<% } %></td></tr></table>
</body></html>
<select name="cboState" size="1">
    <option>&nbsp;</option>

    <%try{
        myConn = new DBConnection();
        Connection conn = myConn.getConnection();
        Statement stmt = conn.createStatement();
        sqlQuery = "SELECT STATE_ID, STATE_DESC
        FROM STATE";
        rs = stmt.executeQuery(sqlQuery);
        while (rs.next()) {
            String stateID = rs.getString("STATE_ID");
            String stateDesc = rs.getString("STATE_DESC");
            if (st.equals(stateDesc)) {
                <option value="<%=stateID%>" selected><%=stateDesc%></option>
            } else {
                <option value="<%=stateID%>"><%=stateDesc%></option>
            }
        }
        rs.close();
        stmt.close();
        myConn.closeConnection();
    } catch(Exception ex){
    }
    %>
</select>
Registration.jsp

<html>
<%@ page language = "java" import = "java.sql.*, common.*" %>
<!--java script for client side validations-->
<!--jsp that collects the registration information-->
<head><title>Registration</title>
<link rel="stylesheet" href="style.css" type="text/css">
<script language="JavaScript">
  function evalAll() {
    test = true;
    if((document.all) || (document.getElementById)){
      var userID = document.regForm.txtUserID.value;
      var passwd = document.regForm.txtPassword.value;
      var passwd_retype =
        document.regForm.txtPasswordRetype.value;
      var firstName = document.regForm.txtFirstName.value;
      var lastName = document.regForm.txtLastName.value;
      var homePhone = document.regForm.txtHomePhone.value;
      var city = document.regForm.txtCity.value;
      var streetAddress =
        document.regForm.txtStreetAddress.value;
      var myIndex =
        document.regForm.cboState.selectedIndex;
      var selStateValue =
        document.regForm.cboState.options[myIndex].value;
      var zipCode =
        document.regForm.txtZipCode.value;
      var myIndexb =
        document.regForm.cbobState.selectedIndex;
      var selStateValueb =
        document.regForm.cbobState.options[myIndexb].value;
      var zipCodeb =
        document.regForm.txtbZipCode.value;
    }
  }
</script>
</head>
<body>
</body>
</html>
if (passwd!=null && passwd_retype!=null){
    if (passwd!=passwd_retype){
        alert('Password and Retype Password should be same');
        document.regForm.txtPassword.value="";
        document.regForm.txtPasswordRetype.value="";
        document.regForm.txtPassword.focus();
        test= false;
    }
}

if(userID == ""){
    alert("User ID cannot be blank");
    document.regForm.txtUserID.focus();
    test = false;
    return test;
} else if(passwd == ""){
    alert("Password cannot be blank");
    document.regForm.txtPassword.focus();
    test = false;
    return test;
} else if(passwd_retype == ""){
    alert("Password Retype cannot be blank");
    document.regForm.txtPasswordRetype.focus();
    test = false;
    return test;
} else if(firstName == ""){
    alert("First Name cannot be blank");
    document.regForm.txtFirstName.focus();
    test = false;
    return test;
} else if(lastName == ""){
    alert("Last Name cannot be blank");
    document.regForm.txtLastName.focus();
    test = false;
    return test;
} else if(homePhone == ""){
    alert("Home Phone cannot be blank");
    document.regForm.txtHomePhone.focus();
    test = false;
    return test;
} else if(streetAddress == ""){
    alert("Street Address cannot be blank");
    document.regForm.txtStreetAddress.focus();
    test = false;
    return test;
} else if(city == ""){
    alert("City cannot be blank");
    document.regForm.txtCity.focus();
    test = false;
    return test;
} else if(selStateValue == ""){
    alert("State cannot be blank");
    document.regForm.cboState.focus();
    test = false;
    return test;
```javascript
} else if(zipCode == ""){
    alert("Zip Code cannot be blank");
document.regForm.txtZipCode.focus();
test = false;
return test;
}
return test;

function evalAddress(){
    var city =document.regForm.txtCity.value;
    var streetAddress =
document.regForm.txtStreetAddress.value;
    var myIndex =
document.regForm.cboState.selectedIndex;
    var selStateValue =
document.regForm.cboState.options[myIndex].value;
    var zipCode =document.regForm.txtZipCode.value;
    //alert(document.regForm.chkAddrSame.checked);
if (document.regForm.chkAddrSame.checked) {
document.regForm.txtbCity.value = city;
document.regForm.txtbStreetAddress.value =
streetAddress;
document.regForm.cbobState.selectedIndex =
myIndex;
    document.regForm.cbobState.options[myIndex].value =
    selStateValue;
    document.regForm.txtbZipCode.value = zipCode;
} else {
    document.regForm.txtbCity.value = "";
    document.regForm.txtbStreetAddress.value = "";
document.regForm.cbobState.selectedIndex = "";
    document.regForm.cbobState.options[myIndex].value = "";
    document.regForm.txtbZipCode.value = "";
}
}

function validateInt() {
    var iString = document.regForm.txtZipCode.value
    alert(iString);
    return ("" + parseInt(iString)) == iString);
}

function AllowNumericOnly(){
    Expression = '0123456789';
    var ch = String.fromCharCode(window.event.keyCode);
    ch = ch.toLowerCase();
    var a = Expression.indexOf(ch);
    if (a == -1)
        window.event.keyCode = 0;
}//end function
</script>
</head>
```
<body BACKGROUND= "yellow_paper.jpg">
<center>
<font FACE="Century Gothic" size="5"><B>E-Shopping Mall.com</B></font></center>

<TABLE cellSpacing=5 cellPadding=10 width=800 border="1">
<tr>
<td align="middle"><a href="Login.jsp">Home</a></td>
</tr>
</table>
<h1><font color="#08246B" size="5" face="arial,helvetica">Registration</font></h1>
<form name="regForm" method="post" action="RegistrationAction.jsp">
<table BACKGROUND= "bg.jpg" border="0" align="left" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF">
<tr>
<td colspan="3" align="middle" bgcolor="#333399"> <font color="#FFFFFF" size="2" face="Arial, Helvetica">
<b>Login Information</b></font>
</td>
</tr>
<tr><td colspan="3">&nbsp;</td></tr>
<tr>
<td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica"><b>*</b></font>
<font color="#000000" size="2" face="Arial, Helvetica"><b>User ID:</b></font>&nbsp;</td>
<td>&nbsp;</td>
<td><input name="txtUserID" type="text" maxlength="25" size="25"></td>
</tr>
<tr>
<td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica"><b>*</b></font>
<font color="#000000" size="2" face="Arial, Helvetica"><b>Password:</b></font>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;<input name="txtPassword" type="password" maxlength="25" size="27"></td>
</tr>
<tr>
<td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica"><b>*</b></font>
<font color="#000000" size="2" face="Arial, Helvetica"><b>Retype Password:</b></font>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;<input name="txtPasswordRetype" type="password" maxlength="27" size="27"></td>
</tr>
</table>
<br><br><br><br><br><br><br><br><br>
<table BACKGROUND= "bg.jpg" border="0" align="left" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF">
<tr>
<td colspan="3" align="justify" bgcolor="#333399">
<font color="#FFFFFF" size="2" face="Arial, Helvetica">
Personal Information</font>
</td>
</tr>
</table>
<br><br><br><br><br><br><br><br><br>
</body>
<tr><td align="right"><font color="#000000" size="2" face="Arial, Helvetica">Prefix:</font></td><td>&nbsp;</td><td><select name="cboPrefix" size="1">
  <option selected>&nbsp;</option>
  <option>Mr.</option>
  <option>Mrs.</option>
  <option>Miss.</option>
</select></td></tr>
<tr><td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica"><b>*</b> First Name:</font></td><td>&nbsp;</td><td><input name="txtFirstName" type="text" maxlength="40" size="40"></td></tr>
<tr><td align="right"><font color="#000000" size="2" face="Arial, Helvetica">Middle Name:</font></td><td>&nbsp;</td><td><input name="txtMiddleName" type="text" maxlength="40" size="40"></td></tr>
<tr><td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica"><b>*</b> Last Name:</font></td><td>&nbsp;</td><td><input name="txtLastName" type="text" maxlength="40" size="40"></td></tr>
<tr><td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica"><b>*</b> Home Phone:</font></td><td>&nbsp;</td><td><input name="txtHomePhone" type="text" maxlength="15" size="40"></td></tr>
<tr><td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica">Office Phone:</font></td><td>&nbsp;</td><td><input name="txtOfficePhone" type="text" maxlength="15" size="40"></td></tr>
<table>
<thead>
<tr>
<th>Address</th>
<th>Street Address:</th>
<th>City:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```java
// gets the state information
DBConnection myConn = null;
ResultSet rs = null;
String sqlQuery = "SELECT STATE_ID, STATE_DESC FROM STATE";
try {
    myConn = new DBConnection();
    Connection conn = myConn.getConnection();
    Statement stmt = conn.createStatement();
    sqlQuery = "SELECT STATE_ID, STATE_DESC FROM STATE";
    rs = stmt.executeQuery(sqlQuery);
    while (rs.next()) {
        String stateID = rs.getString("STATE_ID");
        String stateDesc = rs.getString("STATE_DESC");
    }
} finally {
    myConn.close();
    rs.close();
}
```
<option value="<%=stateID%>"<%=stateDesc%></option>

rs.close();
stmt.close();
myConn.closeConnection();
} catch (Exception ex) {
} %>
</td>
</tr>
<tr>
<td align="right"><font color="#FF0000" size="2" face="Arial, Helvetica"><b>*</b></font>
<font color="#000000" size="2" face="Arial, Helvetica"><b>Zip Code:</b></font></td>
<td>&nbsp;</td>
<td><input name="txtZipCode" type="text" maxlength="5" size="5"
onKeyPress="AllowNumericOnly();"></td>
</tr>
</table><br><br><br><br><br><br><br><br><br>
<table BACKGROUND= "bg.jpg" border="0" align="left" cellpadding="0"cellspacing="0" bgcolor="#FFFFFF">
<tr>
<td align="center">
<input name="pagemode" type="hidden" value="submit">
<input type="submit" onKeyDown='return evalAll();' value="Submit" onClick='return evalAll();'>
</td>
<td>&nbsp;&nbsp;</td>
<td><input type="reset" value="Reset"></td>
</tr>
</table>
</form>
</body>
</html>
<form name="regAction" action="Login.jsp">
<%
    //stores the registration information in the database
    DBConnection myConn = null;
    Connection conn = null;
    ResultSet rs = null;
    String sqlQuery = null;
    int rowsAffected = 0, rowsAffectedUA = 0, rowsAffectedBA = 0;
    Statement stmt = null;
    String userID = request.getParameter("txtUserID");
    String passwd = request.getParameter("txtPassword");
    String passwd_retype = request.getParameter("txtPasswordRetype");
    String userPrefix = request.getParameter("cboPrefix");
    String firstName = request.getParameter("txtFirstName");
    String middleName = request.getParameter("txtMiddleName");
    String lastName = request.getParameter("txtLastName");
    String homePhone = request.getParameter("txtHomePhone");
    String officePhone = request.getParameter("txtOfficePhone");
    String streetAddress = request.getParameter("txtStreetAddress");
    String city = request.getParameter("txtCity");
    String state = request.getParameter("cboState");
    String zipCode = request.getParameter("txtZipCode");
    String addrSame = request.getParameter("chkAddrSame");
    String streetbAddress = request.getParameter("txtbStreetAddress");
    String cityb = request.getParameter("txtbCity");
    String stateb = request.getParameter("cbobState");
    String zipbCode = request.getParameter("txtbZipCode");
    boolean userFound = false;

    try{
        myConn = new DBConnection();
        conn = myConn.getConnection();
        stmt = conn.createStatement();
        sqlQuery = "SELECT COUNT(*) FROM USERS WHERE USER_ID = '" + userID + ";"
        rs = stmt.executeQuery(sqlQuery);
        while(rs.next()){
            if ( !rs.getString(1).equals("0") ){
                userFound = true;
                session.setAttribute("USER_ID",(String) userID);
            }
        }
    } catch(Exception e) {
        alert("Error: Could not connect to database.
                 Please try again later.
                 If the problem persists,
                 please contact your system administrator.");
    }

<%>
    <script language="JavaScript">
        alert( "This User ID ALREADY EXISTS! Please enter different User ID" );
        document.regAction.action="Registration.jsp";
        document.regAction.submit();
    </script>
</%
</}%>
	
rs.close();

if (userFound == false) {

- 81 -
sqlQuery = "INSERT INTO USERS (USER_ID, PASSWD,
PASSWD_RETYPE, PREFIX, FIRST_NAME, "+ "LAST_NAME, MIDDLE_NAME, PHONE_NUM_RES, PHONE_NUM_OFF) "+ "VALUES('"+ userID + '", "+ passwd + ","+ passwd_retype + ","+ "+ + + firstName + ","+ lastName + ","+ "+ " + middleName + "," + homePhone + ","+ officePhone + ");";
rowsAffected = stmt.executeUpdate(sqlQuery);
sqlQuery = "INSERT INTO USER_ADDRESS(USER_ID,
STREET_ADDRESS, CITY, STATE_ID, "+ "ZIP_CODE) VALUES
( "+ userID + ","+ streetAddress + ","+ city + ","+ 
state + ","+ zipCode + ");";
rowsAffectedUA = stmt.executeUpdate(sqlQuery);
if (rowsAffected == 1 && rowsAffectedUA==1) {
  <h3> Hello <%=firstName%><%=lastName%></h3>
  <input type="submit">
  <script language="JavaScript">
    document.regAction.action= "Login.jsp";
    document.regAction.submit();
  </script>
} else {
  <h3> There is an error in the application </h3>
} //end of if rowsAffected
//end of if userFound
} //catch(Exception ex){
userFound = false;
stmt.close();
myConn.closeConnection();

"
return.jsp

<html>
<%@ page import='java.sql.*,common.*'%>
<head><title>Return</title></head>
<body BACKGROUND= "yellow_paper.jpg">
<jsp:include page="topmenu.jsp" />
<br><br>
<TABLE width=100% BACKGROUND= "bg.jpg">
<TR><TH><font size="6">Merchandise Return</font></TH></TR>
<tr><td><br></td></tr>
<tr><td><a href="returnrequest.jsp"><font size="4">Request a Return Reference Number</font></a></td></tr>
<tr><td><a href="returnhistory.jsp"><font size="4">Check Return History</font></a></td></tr>
</table>
</body></html>
```html
<html>
<%@page import='java.sql.*,java.text.*,common.*,java.util.*'%>
<head><title>Return Action</title></head>
<body BACKGROUND="yellow_paper.jpg">

  <% if (session.getAttribute("userID") == null){
    response.sendRedirect("Login.jsp");
  } %>

  <%
    DBCConnection myConn = new DBCConnection();
    Connection conn = myConn.getConnection();
    Statement stmt = null;
    ResultSet rs = null;
    String sqlQuery = null;
    String usr = (String)session.getAttribute("userID");
    int rowsUpdated=0;
    String returnid = null;
  %>

  <TABLE width=100% BACKGROUND="bg.jpg">
    <TR><TH><font size="6">Return Request</font></TH></TR>
    <tr><td><br></td></tr>
    <tr><td align="center"><font size="5">
      <%
        try {
          stmt = conn.createStatement();
          sqlQuery = "insert into returnhistory values(" +
          request.getParameter("orderid") +
          ", " + request.getParameter("paymethod") +
          ", " + request.getParameter("cardnumber") + ",
          sysdate, return_id_seq.NextVal, 'Pending');
          rowsUpdated = stmt.executeUpdate(sqlQuery);
          if(rowsUpdated!=0) {
            rs = stmt.executeQuery("select refid from returnhistory where oid=" +
            request.getParameter("orderid") + ");
            if(rs.next()) {
              out.println("Your return request reference number is");
              out.println(rs.getString(1));
            } else
              out.println("Please contact customer service center for help.");
          }
        } catch(Exception ex2) {} 
        } catch(Exception ex) { }
      %>
      </font></td></tr></TABLE>

  </body></html>
```
returnhistory.jsp
<html>
<%@page import='java.sql.*,java.text.*,common.*,java.util.*'%>
<head><title>Return Request History</title></head>
<body BACKGROUND= "yellow_paper.jpg">
<% if (session.getAttribute("userID") == null){
     response.sendRedirect("Login.jsp");
} %>
<jsp:include page="topmenu.jsp" />
<%
    DBConnection myConn = null;
    ResultSet rs = null;
    myConn = new DBConnection();
    Connection conn = myConn.getConnection();
    Statement stmt = null;
    String usr = (String)session.getAttribute("userID");
%>
</jsp:include page="topmenu.jsp" />
<table width=100% BACKGROUND= "bg.jpg">
<tr>
    <th><font size="6">Return Request History</font></th>
</tr>
</table>
<table>
<thead>
<tr>
<th>Return Reference Number</th>
<th>Amount</th>
<th>Request Date</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>returnRefId</td>
<td>returnAmount</td>
<td>returnDate</td>
<td>returnStatus</td>
</tr>
</tbody>
</table>

```coldfusion
try{
stmt = conn.createStatement();
rs = stmt.executeQuery("select refid, amount, 
TO_CHAR(requestdate, 'mm/dd/yyyy'), status " +
"from paymentinfo pi,
returnhistory rh " +
"where userid='" + usr +
"' and pi.ordernumber=rh.oid " +
"order by refid") ;
if(rs.next()) {
String returnRefId=rs.getString(1);
String returnAmount=rs.getString(2);
String returnDate=rs.getString(3);
String returnStatus=rs.getString(4);
}%>
```
```html
<% }
}
else {
    <TR><td colspan="3" align="center"><h3><BR>You have no return requests.</h3></td></TR>
<% }
try{
    rs.close() ;
    stmt.close() ;
}catch(Exception ex2){}
%
</TABLE>
</TD></TR>
</TABLE><%
} catch(Exception ex) { %>
    Error: <%=ex.getMessage()%>
<% }
%
</body></html>
```

**returnrequest.jsp**

<html>
<%@page import='java.sql.*, java.text.*, common.*, java.util.*'%>
<head><title>Return Request</title></head>
<body BACKGROUND= "yellow_paper.jpg">
  <% if (session.getAttribute("userID") == null) {
    response.sendRedirect("Login.jsp");
  } %>
  <jsp:include page="topmenu.jsp" />
  <br><br>
  <table width=100% BACKGROUND= "bg.jpg">
    <tr>
      <th><font size="6">Return Request</font></th>
    </tr>
    <tr><td><br></td></tr>
    <tr>
      <td>
        <table cellspacing=1 cellpadding=2 width=100%>
          <tr>
            <th bgcolor="#333399"><font color="ffffff">Order Number</font></th>
            <th bgcolor="#333399"><font color="ffffff">Payment Method</font></th>
            <th bgcolor="#333399"><font color="ffffff">Card Number</font></th>
            <th bgcolor="#333399"><font color="ffffff">Amount</font></th>
            <th bgcolor="#333399"><font color="ffffff">Order Date</font></th>
            <th bgcolor="#333399"><font color="ffffff">Action</font></th>
          </tr>
          <tr>
            <td></td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
  <br><br>
  <% try {
    stmt = conn.createStatement();
    rs = stmt.executeQuery("select distinct(oh.oid),
      pi.paymethod, pi.cardnumber, " +
      "pi.amount,
      TO_CHAR(oh.odate, 'mm/dd/yyyy') " +
      "from paymentinfo pi,
      orderhistory oh " +
      "'" +
      " and pi.ordernumber=oh.oid and " +
      "rh.oid from returnhistory rh" +
      " order by oh.oid");
    if(rs.next()) {
      String orderid=rs.getString(1);
      String paymethod=rs.getString(2);
      String cardnumber=rs.getString(3);
      String amount=rs.getString(4);
    }
  } %></jsp:include>
</body>
String ordertime = rs.getString(5);

`<TR>
<TD valign=top align="left"><STRONG><font size="3"><%=orderid%></font></STRONG></TD>
<TD valign=top align="left"><STRONG><font size="3"><%=paymethod%></font></STRONG></TD>
<TD valign=top align="left"><STRONG><font size="3"><%=cardnumber%></font></STRONG></TD>
<TD valign=top align="left"><STRONG><font size="3">$<%=amount%></font></STRONG></TD>
<TD valign=top align="left"><font size="3"><%=ordertime%></font></TD>
<TD valign=top align="left"><a href="returnaction.jsp?orderid=<%=orderid%>&paymethod=<%=paymethod%>&cardnumber=<%=cardnumber%>&amount=<%=amount%>"><font size="3">Return Request</font></a></TD>
</TR>

<% while (rs.next()) {
orderid = rs.getString(1);
paymethod = rs.getString(2);
cardnumber = rs.getString(3);
amount = rs.getString(4);
ordertime = rs.getString(5);
%>

`<TR>
<TD valign=top align="left"><STRONG><font size="3"><%=orderid%></font></STRONG></TD>
<TD valign=top align="left"><STRONG><font size="3"><%=paymethod%></font></STRONG></TD>
<TD valign=top align="left"><STRONG><font size="3"><%=cardnumber%></font></STRONG></TD>
<TD valign=top align="left"><STRONG><font size="3">$<%=amount%></font></STRONG></TD>
<TD valign=top align="left"><font size="3"><%=ordertime%></font></TD>
<TD valign=top align="left"><a href="returnaction.jsp?orderid=<%=orderid%>&paymethod=<%=paymethod%>&cardnumber=<%=cardnumber%>&amount=<%=amount%>"><font size="3">Return Request</font></a></TD>
</TR>

<% } %>

<TR><td colspan="3" align="center"><h3>You are currently not able to perform return action.</h3></td></TR>

<% try{
rs.close() ;
stmt.close() ;
}catch(Exception ex2){}%>

</TABLE>
</TD></TR>
<TR><TD align="right">
ship.jsp
<html>
<head><title>Ship</title>
</head>
<body>
   <% response.addHeader("Expires","0"); %>
   <% jsp:include page="topmenu.jsp" />%
   <%
      DBConnection myConn = null;
      ResultSet rs = null;
      myConn = new DBConnection();
      Connection conn = myConn.getConnection();
      Statement stmt = null;
      Statement stmt2 = null;
      NumberFormat nf = NumberFormat.getNumberInstance();
      nf.setMaximumFractionDigits(2);
      nf.setMinimumFractionDigits(2);
      String pid = "";
      String qty = "";

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

      if(vcard == null) {
         vcard = "NA";
         <strong>No Virtual Card for payment is selected. So completed order without using any Card.<BR> <strong>
      }
   %>
   <%
      int orderid = 0;
      try{
         stmt = conn.createStatement();
      } catch(Exception ex) { %>
         Error: <%=ex.getMessage()%>
      %>
   %>
rs = stmt.executeQuery("SELECT order_seq.nextval from dual");

    if(rs.next()) {
       orderid = rs.getInt(1);
    }

    try{
        rs.close();
        stmt.close();
    }catch(Exception ex4){}

}catch(Exception ex5) { %>
Error: <%=ex5.getMessage()%>
<%      }

try{
    String cid = "+"+session.getAttribute("userID");
    String cname = "+"+session.getAttribute("userName");
    String cstreet = "";
    String ccity = "";
    String ccode = "";
    String cstate = "";
    String bname = ""+session.getAttribute("userName");
    String bstreet = "";
    String bcity = "";
    String bcode = "";
    String bstate = "";

    stmt = conn.createStatement();
    rs = stmt.executeQuery("SELECT street_address, city, state_desc, zip_code FROM user_address u, state s where u.state_id=s.state_id and u.user_id="+session.getAttribute("userID")+"'");
    if(rs.next()) {
        cstreet = rs.getString(1);
        ccity = rs.getString(2);
        cstate = rs.getString(3);
        ccode = rs.getString(4);
    }

    try{
        rs.close();
        stmt.close();
    }catch(Exception ex11){}

    stmt = conn.createStatement();
    rs = stmt.executeQuery("SELECT street_address, city, state_desc, zip_code FROM user_address u, state s where u.state_id=s.state_id and u.user_id="+session.getAttribute("userID")+"'");
    if(rs.next()) {
        bstreet = rs.getString(1);
        bcity = rs.getString(2);
        bstate = rs.getString(3);
        bcode = rs.getString(4);
    }

    try{
rs.close();
stmt.close();
} catch (Exception ex12) {

String insstmt = "insert into orders " + "( orders_id " + ", customers_id " + ", customers_name " + ", customers_street_address " + ", customers_city " + ", customers_postcode " + ", customers_state " + ", customers_country " + ", customers_telephone " + ", customers_email_address " + ", billing_name " + ", billing_street_address " + ", billing_city " + ", billing_postcode " + ", billing_state " + ", billing_country " + ", payment_method " + ", orders_status) " + "values(" + " " +"+orderid + ", 0" + ", "+cname+'
' + ", "+cstreet+'
' + ", "+ccity+'
' + ", "+ccode+'
' + ", "+cstate+'
' + ", ' '" + ", ' '" + ", ' '" + ", "+bname+'
' + ", "+bstreet+'
' + ", "+bcity+'
' + ", "+bcode+'
' + ", "+bstate+'
' + ", ' '" + ", '+vcard+'
' + ", 0 )";

stmt = conn.createStatement();
int inscount = stmt.executeUpdate(insstmt) ;

Hashtable orderItems = order.getOrder();
Enumeration keys = orderItems.keys();
float total = 0;
while( keys.hasMoreElements() ) {
    pid=(String)keys.nextElement();
    qty=(String)orderItems.get(pid);
    stmt = conn.createStatement();
    rs = stmt.executeQuery("SELECT p.products_id,p.products_name,p.products_description,p.products_price FROM products p, products_description pd where")
    ...
p.products_id=pd.products_id and p.products_id="+pid+" order by products_id") ;

if(rs.next()) {
    String insstmt2 = "insert into orders_products"
    + " ( orders_products_id"
    + " , orders_id"
    + " , products_id"
    + " , products_name"
    + " , products_price"
    + " , products_quantity) "
    + " values ( order_prod_seq.nextval"
    + " ,"+orderid
    + " ,"+rs.getString(1)
    + " ,"+rs.getString(2)+"
    + " ,"+rs.getString(4)
    + " ,"+qty+");

    stmt2 = conn.createStatement();
    int inscount2 = stmt2.executeUpdate(insstmt2);
}

try{
    rs.close();
    stmt.close();
    stmt2.close();
}catch(Exception ex2){
    
} %>

  <BR><B>Thank you for shopping</B>
  <BR><B>Your Order Number: <%=orderid%></B>
  <BR><B>Your order has been accepted for shipment</B>

<% order.removeOrder(); %>
  }catch(Exception ex) { %>
    Error: <%=ex.getMessage()%> 
  %>
  
</body></html>
style.css

BODY
{
    FONT-SIZE: 12;
    FONT-FAMILY: Arial
}
TABLE.Data
{
}
TABLE.Data TH
{
    FONT-WEIGHT: bold;
    FONT-SIZE: 12;
    COLOR: #ffffff;
    FONT-FAMILY: Arial;
}
TABLE.Data TD
{
    FONT-SIZE: 12;
    FONT-FAMILY: Arial;
}
BODY TABLE
{
}
BODY TABLE TD
{
    FONT-SIZE: 12;
    FONT-FAMILY: Arial
}
BODY TABLE TD A
{
    FONT-WEIGHT: bold;
    FONT-SIZE: 12;
    COLOR: #333399;
    FONT-FAMILY: Arial
}
INPUT.SmallButton
{
    BORDER-RIGHT: medium none;
    BORDER- TOP: medium none;
    FONT-WEIGHT: bold;
    FONT-SIZE: 12;
}
topmenu.jsp

```html
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<link rel="stylesheet" href="style.css" type="text/css">
<title>Top Menu</title>
</head>
<body>
</body>
</html>
```