العنوان:	Handover Algorithm and Traffic
المؤلف الرئيسي:	Abd Allah, Amani
مؤلفين آخرين:	Bilal, Khalid Hamid(super)
التاريخ الميلادي:	2010
موقع:	أم درمان
الصفحات:	1 -73
رقم MD:	560860
نوع المحتوى:	رسائل جامعية
اللغة:	English
الدرجة العلمية:	رسالة ماجستير
الجامعة:	جامعة أم درمان الاسلامية
الكلية:	كلية الدراسات العليا
الدولة:	السودان
قواعد المعلومات:	Dissertations
مواضيع:	الاتصالات، الهاتف المحمول، هندسة البرمجيات، السودان
رابط:	https://search.mandumah.com/Record/560860

Chapter Six
Conclusions & Recommendations

6.1 Conclusion

1. First, the study and the analysis been completed and then the simulation of the handoff algorithm have been done using JAVA software program.

2. The simulation program consider the signal strength of the service cell, and the signal strength of the neighbor cell, T-AD, T-DROP which are the parameters, from the simulation we observe that results of classification of cells, handover success, handover failure and call drop.

3. Also failure handover of real data measured for WAD ALAMEEN cell have been listed and measured using drive test and calculating the forward receive power, reserved transmitted power, FER and Ec/Ro.

4. Lastly, we perform the handoff algorithm by traffic analysis of the cell of BAGAIR, GIAD AND UMDAWAN BAN by considering the traffic, drop rate, handover success rate and handover failure rate.The analysis results been shown in tables and graphs

6.2 Limitations

1. The real data used in this analysis was old because of the difficulty of obtaining an up to date data from the service providers due to the confidentiality and security between telecommunications companies as a result of competition.

2. The real data analysis and test for WAD ELAMEEN through the service provider is very expensive in order to provide a means of movements, optimization Team and the specialized tools.

3. Time for simulation to run the program will increase if the number for sample is increased which need a computer with higher specifications

6.3 Recommendation

1. The analysis consider internal handoff algorithm, we suggest that to analyze external handoff.

2. The analysis was done for soft handoff for CDMA system, the hard handoff for GSM system need to be analyzed for the sake of comparison.

3. To extend the simulation program to include all effective parameter of handoff algorithm to achieve optimization.

4. The traffic analysis consider only three cells, we suggest that the analysis should deal with at least a cluster size of seven cells to see the effect of signal interference ratio on the handoff algorithm.

5. To extend the performance of the traffic analysis of the handoff algorithm to include the pit error rate (PER) and quality of service (QOS).

العنوان:	Web store front
المؤلف الرئيسي:	Al Ahmari, Saad A.
مؤلفين آخرين:	Martin, Dennis(Super.)
التاريخ الميلادي:	2002
موقع:	سكرانتون، بنسلفانيا
الصفحات:	1 - 41
رقم MD:	617982
نوع المحتوى:	رسائل جامعية
اللغة:	English
الدرجة العلمية:	رسالة ماجستير
الجامعة:	University of Scranton
الكلية:	College of Arts and Sciences
الدولة:	الولايات المتحدة الأمريكية
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، المتاجر الالكترونية، شبكة الانترنت
رابط:	https://search.mandumah.com/Record/617982

2. System Overview

This section describes all the constraints on and functions performed by the system to define the project fully for the reader. It lists all the functions performed by the system and the constraints under which it is to operate. This section consists of the following sections:

Section 2.1: This section provides an overview of the top-level view of the system architecture.

Section 2.2: This section provides an overview of the system functions.

Section 2.3: This section describes all the elements involved in a Web Store Front.

2.1 Top-Level View of System

This project used a three-tier framework to build the e-commerce Website. The first tier consists of multiple clients connected to the enterprise system. The second tier is comprised of the business and program logic of the system, incarnated by a Web Server. The third tier was comprised of an Oracle database server and the database store. The components are described below. The deployment diagram of the Web Store Front shows the physical layout of the system. It shows that the system consists of three major system components or tiers. See Figure 1.

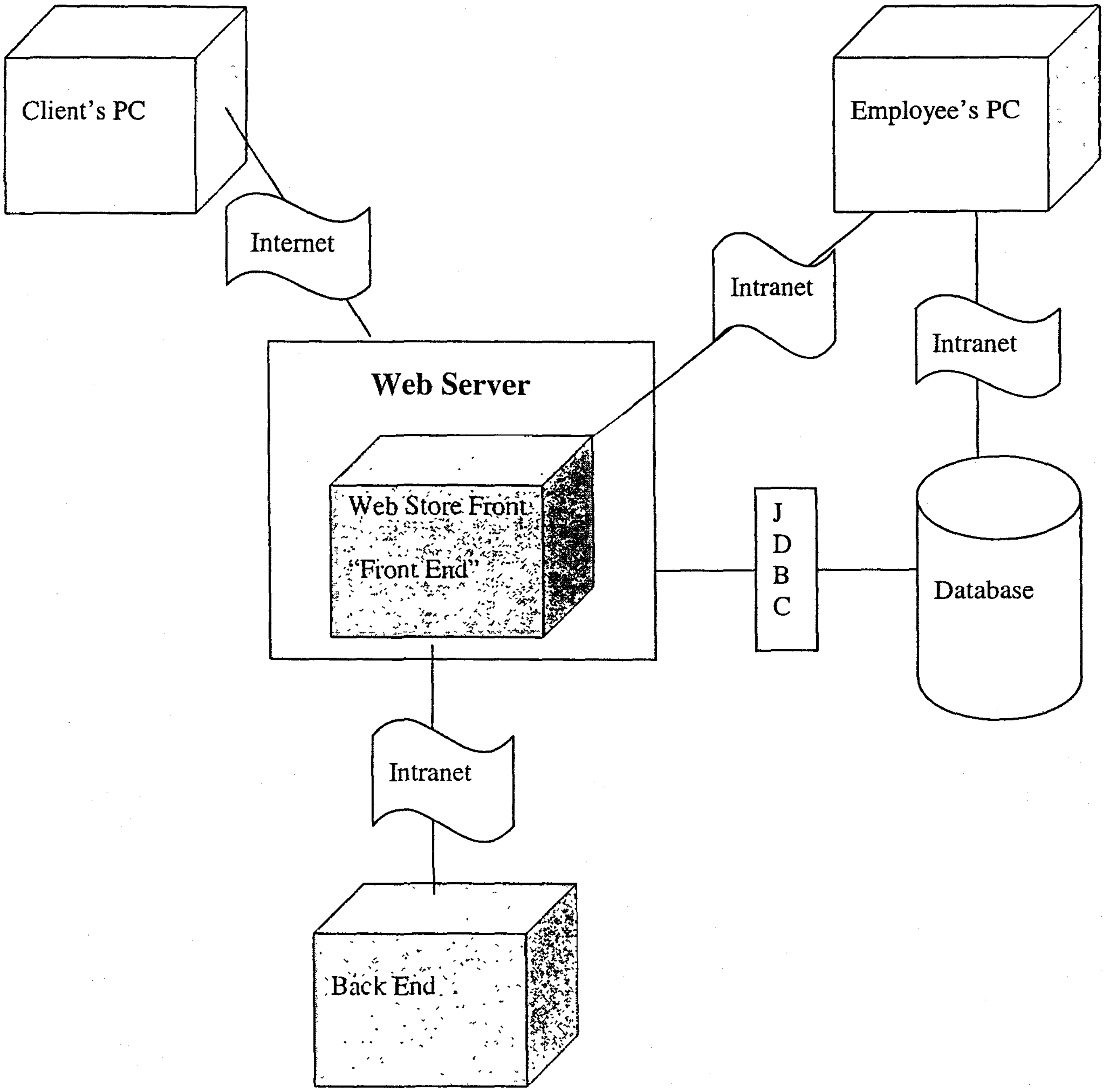

Figure 1- Deployment Diagram

- First Tier

The first tier consists of Web browsers that are located on remote computers. The computers are allowed to connect to the enterprise system through the Internet. This creates a server-client system in which many clients can be served by a single or a cluster of servers. The first tier is connected to the entire system through the second tier. Specifically, Web Browsers make requests to the Web Server by use of the Hyper Text Transmission Protocol. The Web Server receives the requests and returns the requested information. The first tier is for the client, and contains the presentation logic. This includes simple control and user input validation, which is implemented with HTML and JavaScript.

- Second Tier (Middle Tier)

The second tier "Front End" is also known as the application server, and provides the business process logic and the data access. It includes the Web Server business logic. The Web Server, which is a Tomcat Web Server, is responsible for waiting and for answering requests from the Clients (Web Browsers). The Front End can be thought of as containing business logic and programming, that allows a sequence of decisions to be made such as buy, modify account, and view orders. In addition, it is vital in allowing customers to interact (i.e., place an order) with the Web Store database. JDBC API (Java Database Connective, Application Programming Interface) has two parts, an application-level interface used by the Front End component to access a database, and a service provider interface to attach a JDBC driver to the java based. The Front End also interacts with the Back End, which is a simulation for an external system and has an interface within the Front End. The main goals of the Back End are to process the purchased items from the store and ship items to the customer.

- Third Tier

The third tier consists of the Oracle database server and the oracle database. The oracle database and database server are connected to the second tier, and contain permanent information about orders, inventory and customer's information.

The system is built on a three-tier architecture. The advantages of this approach are:

- It is easier to modify or replace any tier without affecting the other tiers.
- Separating the application and database functionality means better load balancing. It is more scalable and easier to control.
- Adequate security policies can be enforced within the server tiers without hindering the clients.
- The middle tier enables the system to handle more client connections.
- It implements better security and provides easier maintenance.
- It is more scalable and easier to control.

2.2 System Functions

The system provides various functions for the customers and store's staff. The system also differentiates between the registered customer and the unregistered customer. Following is a brief description of each implemented use case as the system was initially planned.

Functions of the customers:

- **Sign On:** The registered customer accesses the sign on page at a set URL, provides his/her username and password, and clicks on "submit." This takes customer to the main interface screen for the registered customers.
- **Create Account**: The unregistered customer accesses the create account at a set URL, fills the account form, and clicks on "submit." This takes the customer to the main interface screen for the registered customer.
- **Change Password**: The registered customer accesses the change password at a set URL, provides his/her username and new password, and clicks on "submit." This takes the customer to another interface screen for either confirmation of the new password or to disprove the password.
- **Modify Account**: The registered customer accesses the modify account at a set URL, provides his/her username, password, and new information, and clicks on "submit." This takes the customer to another interface screen for either confirmation of the modification or disproves it.
- **Search Catalog:** The customer accesses the search catalog page at a set URL, provides key word to search for a particular item in the store, and clicks on

"submit." This takes the customer to an interface screen with the list of the items under that key word if the items are found.

- **View Catalog:** The customer accesses the view catalog page at a set URL, selects a catalog from a drop-down list, and clicks on "submit." This takes the customer to an interface screen with the list of the products under that catalog.
- **Buy:** The registered customer accesses the buy items at set a URL, selects items to buy using either search catalog function or view catalog function, places the selected items into the shopping cart, clicks on "submit." order, and provides the necessary information to process order. This takes the customer to another interface screen for either confirmation of the order or to disprove the order.
- **View Orders**: The registered customer accesses the view orders page at a set URL, provides the order number, and clicks on "submit." This takes the customer to another interface screen with the order information.

Functions of the Employee (store's staff):

- **Delete Items**: The employee accesses the database, selects existing items to delete, and clicks on "delete." button. The system deletes the selected items and updates the inventory number on the database.
- **Add Items:** The employee accesses the database, adds new items, and assigns a product number for the new items and clicks on "add." button. The system automatically creates the items records.

- **Update Items:** The employee accesses the database, provides the item's ID and NAME, provides new description on the selected items, and clicks on "update." button. The system updates the item's information in the database.
- **Find Items:** The employee accesses the database, provides the item's ID, and clicks on "find." button. The system searches for the selected item and displays the selected item information.

  View Orders: The employee accesses the database, requests the view orders. The system displays a list of the selling orders from the database.

2.3 Software Components

This section introduces the software components and the architecture involved in the Web Store Front.

2.3.1 Servlets

After examining the different types of architecture available, I decided to use object-oriented Servlets through Java to develop the Web Store. The following is a list of benefits to java Servlets:

- **Object Oriented**

This project is built by using an object-oriented language so the software is reusable, extensible and maintainable. This means that efforts need not be duplicated when writing a function that makes similar calls. Writing in an object-oriented language allows the programmer to develop code that performs the same function on different data sets, by specifying the data set as an argument. In fact, many of the Servlets in this project are based on an initial Servlet "plan". The contrast of object-oriented program is a single, flow-based program. The problem with a single, flow-based program is that the entire program must be loaded and run, and the entire logic must be traversed, to get to a minor function within the entire program. In this sense, object oriented programs are more advantageous.

- **Servlet based**

A Java Servlet is a Java Applet that runs on a server. Java Servlets run better than CGI scripts and programs because Java Servlets are persistent and can handle multiple requests. This means if two similar requests come one after the other, one Java Servlet can handle both of them. A CGI script can only handle one request before it is destroyed under a CGI script, the script must be compiled at run-time, executed, then destroyed. Then it must be compiled, executed and destroyed again for the second request. This shows that Servlets indeed have a better memory performance than CGI scripts so they are more advantageous.

- **Garbage Collection**

By running Java, the Java Virtual Machine automatically includes a garbage collector that sweeps Java's memory for unused memory blocks and returns them to a pool of available memory. This is not the case with CGI scripts or C and C++ programs, which increases the risk of memory leak. By using a Java-based language, garbage collection is advantageous because it increases the memory and thus overall performance of our program. This is important for the scalability of our enterprise model.

- **Efficient**

With traditional CGI, a new process is started for each HTTP request. If the CGI program itself is relatively short, the overhead of starting the process can dominate the execution time. With Servlets, the Java Virtual Machine is always and handles each request using a lightweight java thread instead of a heavyweight operating system process.

- **Portability**

The Servlet API takes advantage of the Java platform. It is a fairly simple API, which is supported by nearly all Web Srvers so that Servlets may be moved from platform to another platform, usually without any modification whatsoever.

2.3.2 Tomcat Server

The system is built on Tomcat 4.0, which is developed in an open and participatory environment and released under Apache. It is the official reference implementation of the Servlets 2.2 and JSP 1.1 specification. It can be used as a small stand-alone server for testing Servlets and JSP pages, or can be integrated into the Apache Web server. Tomcat, like Apache, is very fast and highly reliable.

2.3.3 Oracle 8i Database

The system database is built by using Oracle8i personal edition database server and the Oracle8i personal edition database to connect to the second tier, and contains permanent information about the store. Since this system is a part of the learning process, this release of Oracle8i was efficient, reliable, and secure for the project.

3. Architectural Design

The system architectural design was designed in two components. It specifies the design entities that collaborate to perform the functionality of the system. Each of these entities has an Abstract Specification and an Interface that expresses the services that it provides to the rest of the system. In turn each design entity is expanded into a set of lower-level design units that collaborate to perform its services. For examples classes, please view Appendix C. Also all tables referred to in this section can be found in the Appendix B.

The two main components of the system architectural are:

- The Front End Architecture
- The Back End Architecture

3.1 Front End Architecture

This component describes two separate parts located in the Front End. The first part is the Web form that the client interacts with. The second part is the employee form that the employee uses to interact with the system. See Figure 2, and 3.

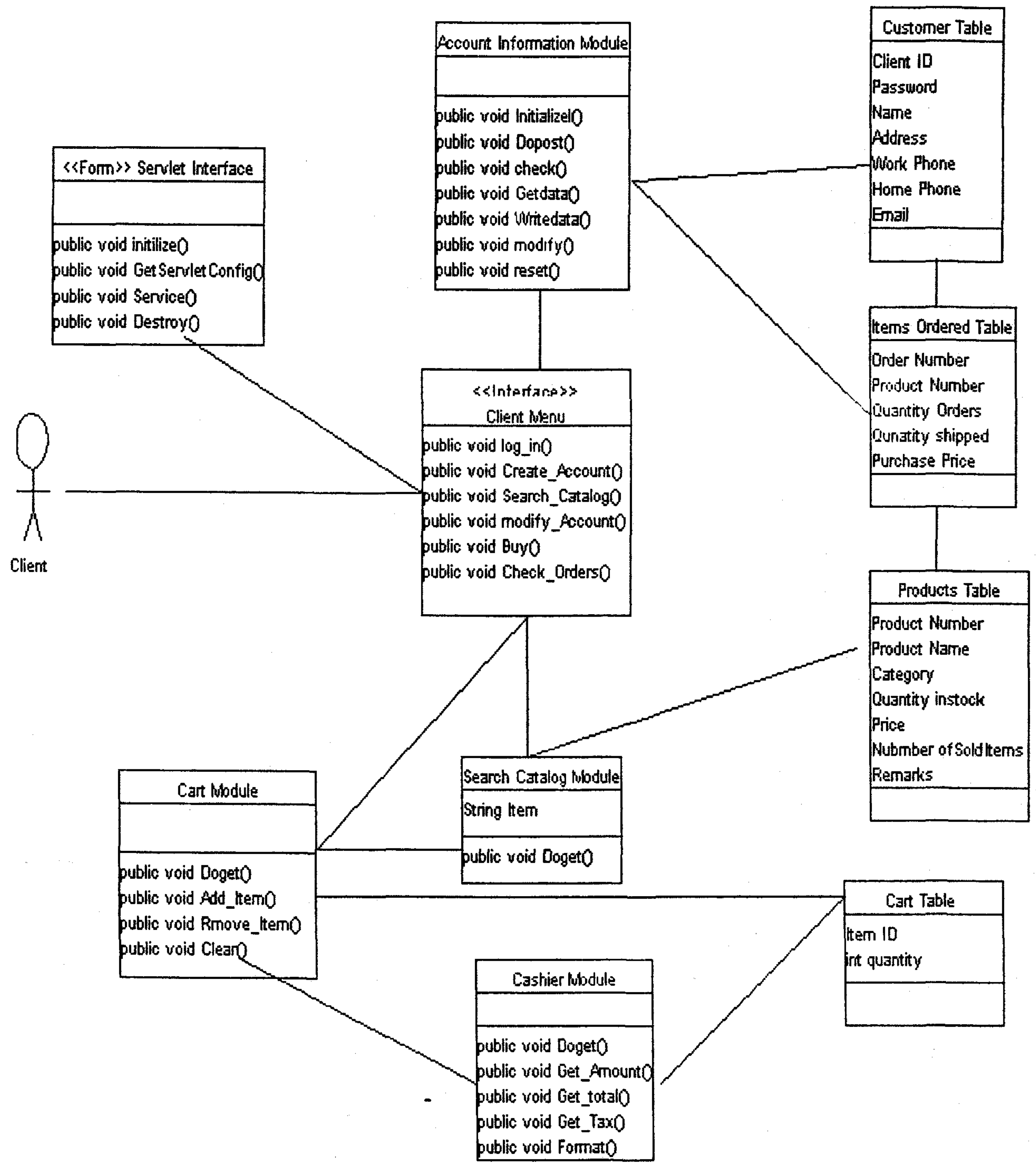

Figure 2 - Front End Architecture Design (Client)

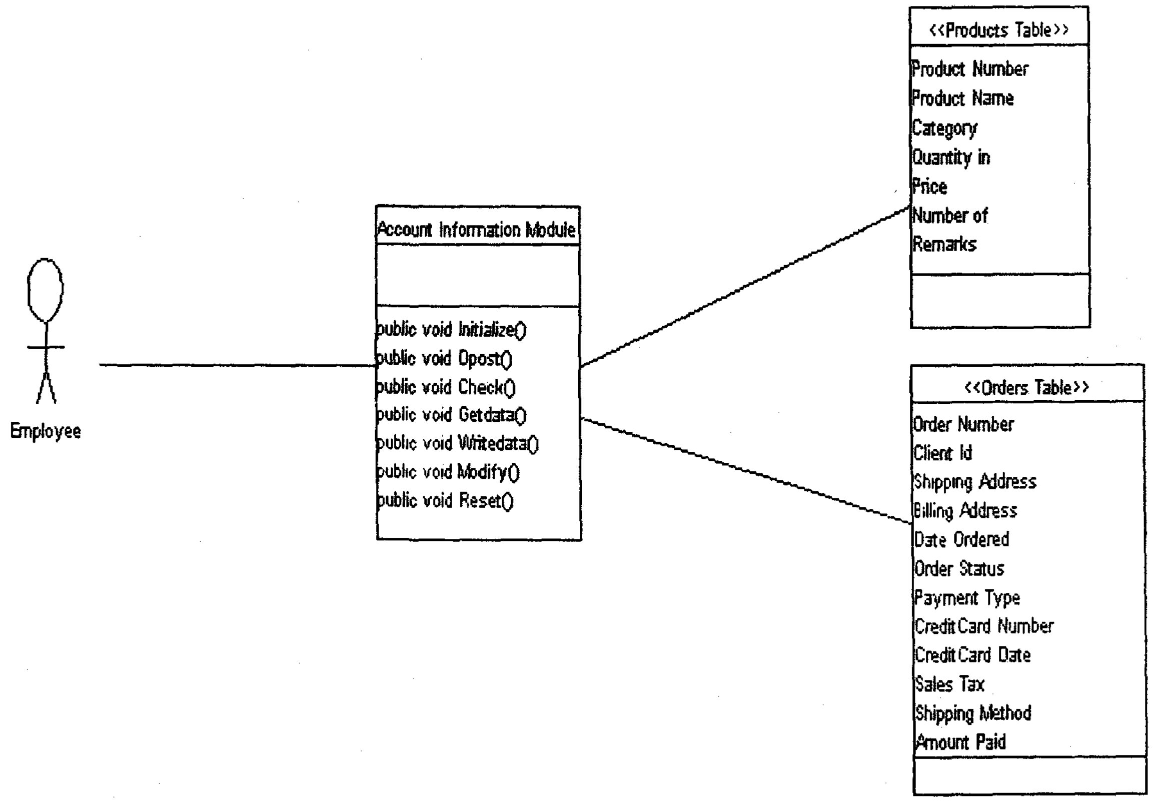

Figure 3 - Front End Architecture Design (Store)

3.2 Back End Architecture

The Front End invokes the Back End to process the purchasing after validation of the client's credit card and the cards expiration date. The Front End sends the clients personal information to the Back End. The Back End then processes the order, remarks that the item is shipped, and notifies the client through the Front End that the order has been shipped. See Figure 4.

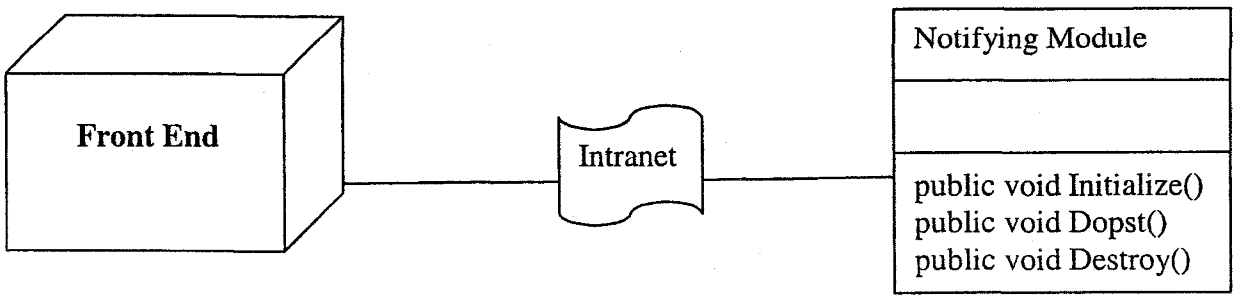

Figure 4 - Back End Architecture Design

3.3 List of Servlets

The following is a list of the Servlets that this system utilizes for the Front End and the Back End.

- **AddToCookie**

The function of this Servlet is to add an identifier id to the user so the commerce system can keep track of his/her shopping cart. This is required because every user has a distinct identifying ID so that items would be placed into the personal shopping cart of each customer.

- **ClientRegServlet**

The function of this Servlet is to register the client. The information is entered from a http form POST. The information is returned to this Servlet and entered in the database.

- **DeleteCookie**

This Servlet deletes the cookie from the user's computer so he can logout and another user can log in or register.

PlaceOrder

This Servlet sends the shopping cart items to the order fulfillment logic. The order fulfillment logic proceeds to insert these items into a table of "placed orders" and sends notification to order fulfillment to pack and ship the orders.

- **SearchPro**

This Servlet is used when the customer wants to view specific catalogs. The Servlet makes a connection with the database and performs a SQL query. The result set is returned to the tomcat server and displayed in a table.

- **ValidateUser**

This Servlet is used when the user registers or when the user checks out. A simple SQL query is performed to make sure the user is registered and valid. If not, the user is returned to a page where he can register.

- **ViewOrderServlet**

This Servlet is used to allow customers to see their current order. The Servlet makes a call to the database and performs a query which returns a result set of items that the customer has ordered. It is displayed by the Java Servlet into a table.

- **ListProdcuts**

This Servlet is used to allow customers to search for a specific item. The Servlet makes a connection with the database and performs a SQL query. The result set is returned to the tomcat server and displayed in a table.

- **ModifyAcc**

This Servlet is used to allow the customer to modify an existing account. A simple SQL query is performed to make the modification in the database. The modified information is returned to the Tomcat server and displayed on the screen for the customer.

- **ChangePassword**

This Servlet is used to allow the customer to change his/her password. A simple SQL query is performed to make the changes in the database. The new password is returned to the Tomcat server and displayed on the screen for the customer.

العنوان:	Web store front
المؤلف الرئيسي:	Al Ahmari, Saad A.
مؤلفين آخرين:	Martin, Dennis(Super.)
التاريخ الميلادي:	2002
موقع:	سكرانتون، بنسلفانيا
الصفحات:	1 - 41
رقم MD:	617982
نوع المحتوى:	رسائل جامعية
اللغة:	English
الدرجة العلمية:	رسالة ماجستير
الجامعة:	University of Scranton
الكلية:	College of Arts and Sciences
الدولة:	الولايات المتحدة الأمريكية
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، المتاجر الالكترونية، شبكة الانترنت
رابط:	https://search.mandumah.com/Record/617982

4. System Code

The code for this system is implemented in two ways, the client side code and the server side code. On the client side are the HTML pages that comprise the user interface and the JavaScripts codes that makes the interface more user-friendly and reliable. On the server side, the Servlets provide a framework for creating applications that implement the request/response between the client "Web Browser" and the Tomcat server.

This code is provided to show the life cycle of Sign On function on client side and on server side. On client side, this code is part of The HTML page for signing on the Web Store Front, which is "login.html." The client provides his/her username and password, and clicks on "submit" button. The Servlet, which is invoked on this HTML page, is ValidateUser. When the user clicks on "submit," the HTML tells the server to do the action "post" on Servlet VaildateUser.

```
(Login.html )
<body>
<font face="verdana" size="2"><b>Online Computer Store</b>
<p>
<H5>If you are a new user, Please</H5><H1><a href="register.html"> Sign
up</a></H1> <u>Login</u>
</p>
<form  action=http://localhost:8080/Servlet/ValidateUscr method="post"
on"submit"="return validate();">
<table cellpadding="1" cellspacing="1" border="0">
<tr><td><font face="verdana" size="2">Name</font></td><td><input type="text"
name="loginid"></td></tr>
<tr><td><font face="verdana" size="2">Password</font></td><td><input
type="password" name="loginpass"></td></tr> </table>
<input type=""submit"" value=""submit"">
<input type="reset" value="Reset">
</form>
</body>
```

On the server side, there is the ValidateUser class. I need to define our class. I also need the javax.Servlet package, which contains the interfaces and the classes intended to be protocol independent ,and the javax.Servlet.http package whichcontains HTTP specific interfaces and classes.

```
import javax.Servlet.*;
import javax.Servlet.http.*;
public class ValidateUser extends HttpServlet {

       Code...

}
```

Servlets initialize the store database with initialization method; the database name is declared as Webstore. A Servlet can read this argument from the ServletConfig at initialization and then later on uses it to open a connection to the database while processing a request. The username and password of the database administrater is “store”

```
private String URL = "jdbc:odbc:WEBSTORE";

public void init( ServletConfig config )
      throws ServletException
     { super.init( config );
         try {
       Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
      connection = DriverManager.getConnection( URL,"store", "store" );
             }
         catch ( Exception e ) {
              e.printStackTrace();
              connection = null;
            }
       }
```

To send response to the client, the dopost method is invoked. This method is declared here.

```
        public void doPost( HttpServletRequest req,
        HttpServletResponse res )
        throws ServletException, IOException
       {
          String  clientid, password ;


          clientid = req.getParameter( "loginid" );
          password = req.getParameter( "loginpass" );
          PrintWriter output = res.getWriter();
          res.setContentType( "text/html" );
                ……. Code
        }
```

To keep track of the customer username and password on the system during using the Web Store Front, cookies are created.

```
        if ( success )
               {

               // Create a [illegible]e
                    Cookie c = new Cookie("storecookie" ,clientid );
                    c.setMaxAge( 3600 );  // seconds until cc [illegible] removed
                    res.addCookie( c );  // must precede getW [illegible]

              }
```

There are two cases here, if the user validation is successful, the customer logs into the system and then the system sends this HTML page, which is included the message, "welcome to the store."

```
// create HTMl page on the sign on successful.
  output.print("<html>");
  output.print("<head>");
  output.print("<BODY><FONT face=verdana size=3><B> Online Computer
Store</B>");
  output.println( "<p align=center><font color=#FF3300> Welcome,
</font>");
  output.print( "<font color=\"#3123FF\"><font size=+1>" + clientid +
"</font></font>");
  output.println( "<p align=center><b> enjoy shopping </b></font>");
  output.print("<p align=center> <b> Now you can buy from the store,
Please view the catalog to see products! </b>");

            ............ .  code



        output.print("<html>");
    output.print("<head>");
    output.close( );
            return;

       }
```

If the user validation fails, the system displays message "please, log in again".

```
        else
          {
       output.print("<html>");
       output.print("<head>");
       output.print("<BODY><FONT face=verdana size=2><B> Online Computer
Store</B>");
       output.println( "<p align=center><font
color=#FF3300><b>Sorry!</b></font>");
       output.println( "<p align=center><font color=#FF3300> <b> Non-
Existance Client,</b></font>");
       output.print("<p align=center> <b> Your username or password is
wrong, please login again or if you forgot your password just contact
us ! </b>");

       output.print("<html>");
       output.print("<head>");
       output.close( );
        return;
          }}
```

As part of this method, the method verifies the username and password in the database. By sending SQL query to the database. The customer's username is defined as ClientID in the customer table in the database. The customer's password is defined as it is in the customer table in the database. Here CheckDB is declared.

```
       private boolean checkDB(String clientid, String password)
      {   boolean fo =false;
         try{
            ResultSet rs;
            statement = connection.createStatement();
            rs= statement.executeQuery(
               "select clientid from customers where clientid= " +  "'"
+ clientid +"'"+ " and password= " + "'" +  password + "'"  );
               code ,,,
             }
```

The Servlet continues to process requests until explicitly unloaded by the servers. I use this method in all Servlets in the system to free system resources such as database. Therefore, in this method the database connection is closed after sending the response to the client

```
  public void destroy()
      {
         try {
            connection.close();
             }
             catch( Exception e ) {
               System.err.println( "Problem closing the database" );
          }
      }
```

5. Testing

5.1 Purpose of Testing

Web Store Front testing is considered successful when an error is detected. Testing is not a distinct phase in system development but is applicable throughout the design, development and maintenance phase. It is concerned with several test stages such as unit testing, module testing, sub-system testing, and system testing. It describes how the system should behave and how the system is validated to meet its functional and non-functional requirements, which are specified in both the Web Store Front software Requirements Specification and Software Design Description.

5.2 Description of Testing

The software developer will test each component as it is built. The system has three separate software components. The Front-End software and the Employee software will be tested by the software developer. I assume that the Back End software has been tested by other developers, since our system only interacts with it, so it is not part of our system. Here I will summarize the testing stages that the system went through.

5.2.1 Unit Testing

The software developer will create a test driver to test the individual units of the system. The test data will be selected to verify specification properties and/or relations between program components. Each class presents a unit, and each unit consists of methods that will be tested by using a Black Box approach. For instance, see the following example for the unit testing on Account Information Unit.

Unit Testing

Please note that any unit that depends upon another unit must be retested after any testing on the other unit.

Name: Account Information Unit

Reference: Software Design Description

Depends on: Nothing

Test Specification:

Test data: set of comprehensive data such as (client name, client username, client address, client phone, client e-m address) provided by the software developer.

Result expected:

- The client 's account is initialized.
- The client 's data is stored in the customer table.
- The client 's data is retrieved from the customer table.
- The client 's data is modified successfully in the customer table.
- The modify account's form is tested.

The test is successful when all of the results are satisfied.

5.2.2 Module Testing

At this stage, units of each module are tested, the objective of module testing is to verify that the modules interact correctly and process work to specification. A WhiteBox approach will be used to accomplish this test. For instance, see the following example for the made Integrative Testing on Search Catalog Module.

Module Name: Search Catalog Module

Module Description: Integration of Search Catalog Unit and Cart Unit

Test Specification:

Test data: a test item is provided by the software developer.

Results expected:

- The test item is found.
- The test item is added to the cart

The test is successful when all of the results are satisfied.

5.2.3 Sub-system Testing

➢ **Front End Sub-system**

This sub-system is tested by using a comprehensive set of data provided by the software developer. The defects will be recorded on the test reports to ensure that the defects are fixed before the system delivered.

➢ Employee Sub-system

This sub-system will be tested by using a test cases approach. The following table shows random test cases.

Test Case	Parameter	Condition	Expected Result	Result
1	INSERT INTO Customers	Customers Table Exists	Success- Table exists	Success
2	INSERT INTO Vendors	Vendors Table Does Not Exist	Failure- Table must exist	Fail
3	INSERT INTO Customers (Name)	Field Name Exists	Success- Field Name exists	Success
4	INSERT INTO Customers (DOB)	Field DOB Does Not Exist	Failure- Field DOB must exist	Fail
5	INSERT INTO Customers (WorkPhone, 2025551234)	Value of WorkPhone Within Range	Success	Success
6	INSERT INTO Customers (WorkPhone, 99x10^99)	Value of WorkPhone Exceeds Positive Range	Failure- Value must be within range	Fail
7	INSERT INTO Customers (WorkPhone, -99x10^99)	Value of WorkPhone Exceeds Negative Range	Failure- Value must be within range	Fail
8	DriverManager.getConnection(URL, “store”, “store”)	Database STORE exists	Success	Success
9	DriverManager.getConnection(URL, “database”, “database”)	Database DATABASE does not exist	Failure- Database must exist	Fail

➢ Back End Sub-system

I assume that other testers test this software since it is out of our system.

5.2.4 System Testing

Subsystems are integrated to make up the entire system. This stage of testing will examine whether or not the system meets its requirements specification. The idea of this phase is to exercise all possible conditions that affect the system as a whole. The developers will fix any defects found. These modules are then reintegrated into the system. Testing continues until all significant defects are fixed. The use cases described in the SRS are listed below. (For Functional Requirements Definition and Requirements Specification see SRS Document, Sections 2.2 and Section 3.2 for each of the following use cases respectively.) (For Use Case Realizations, please refer to SDD, Section 5.1 through 5.11 for each of the following use case respectively.)

- Log In
- Change Password
- Modify Account
- Check Orders
- Search Catalog
- Buy
- Add records
- Delete records
- Update records
- View orders
- Notifying

Each of these use cases was executed by the tester to ensure that each operates according to its Basic Path as described in chapter 3: Requirements Specification in the SRS. Also the tester verified each post-condition as mentioned in the same chapter is true. The validation of all these requirements ensures that the software has successfully met the requirements specification described in the SRS.

العنوان:	Web store front
المؤلف الرئيسي:	Al Ahmari, Saad A.
مؤلفين آخرين:	Martin, Dennis(Super.)
التاريخ الميلادي:	2002
موقع:	سكرانتون، بنسلفانيا
الصفحات:	1 - 41
رقم MD:	617982
نوع المحتوى:	رسائل جامعية
اللغة:	English
الدرجة العلمية:	رسالة ماجستير
الجامعة:	University of Scranton
الكلية:	College of Arts and Sciences
الدولة:	الولايات المتحدة الأمريكية
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، المتاجر الالكترونية، شبكة الانترنت
رابط:	https://search.mandumah.com/Record/617982

6. Conclusion

6.1 Limitation of the System

The mechanism for transferring encryption and secure date between the Tomcat Web server and customers is not available at this time.

- The items that Web Store front offers are not customizable.

6.2 Future Development

This thesis project has built the basic structure for the Web Store Front. I tried to enhance the functions as much as I could, but the project period was critical. Without doubt, there are more software functions that can be added to the Web Store Front. Also, the system should be more secure, which is another major concern in e-commerce. In general, information traveling over the Web is extremely safe if it goes through a secure site. Therefore, the future development should adopt a security approach such as SSL (Secure Sockets Layer), which is a protocol, developed by Netscape for transmitting private documents via the Internet.

Abstract

This thesis focuses on developing Web-business application for an online computer store. The system includes a personalized and easy-to-use Web Site. It helps the store grow into a profitable e-commerce business. The system must use various technologies to build a comprehensive and powerful online computer store. The intended users of this system are the computer store's staff and customers. An Object Oriented Design is used to develop the project with a focus on items such as code re-use and objects. With the provided Web Site that customers interact with, and the employee application that the employees use to interact with the system, I will attempt to incorporate many of the features that are common to most Web order systems.

Table of Contents:

Table of Figures:

THE UNIVERSITY OF
SCRANTON
A JESUIT UNIVERSITY

GRADUATE SCHOOL

UNIVERSITY OF SCRANTON
The Graduate School

The thesis of Saad Al-Ahmari

entitled "Web Store Front"

submitted to the Department of Computing Sciences

in partial fulfillment of the requirements for the degree of MS Software Engineering

in the Graduate School of the University of Scranton has been read and approved by the committee.

Dennis J. Martin
Dr. Dennis Martin

Charles Taylor
Professor Charles Taylor

[illegible]
Dr. Yaodong Bi

April 26, 2002
Date

SCRANTON, PENNSYLVANIA 18510-4632
(570) 941-7600 • FAX: (570) 941-5995

Web Store Front

Thesis

Saad Alahmari

Spring 2002

Submitted in partial fulfillment of the Requirements for the

degree of Master of Science in Software Engineering

University of Scranton

Acknowledgment

The author would like to thank the following people because without the help and co-operation of these wonderful people, this project would have been far from complete.

Dr. Dennis S. Martin – Thesis Advisor and for all the help and advice

Professor Charles Taylor – Second Reader

Dr. Yaodong Bi – For all the help

Last but not least to my family for all the support.

Saad Alahmari

Abstract

This thesis focuses on developing Web-business application for an online computer store. The system includes a personalized and easy-to-use Web Site. It helps the store grow into a profitable e-commerce business. The system must use various technologies to build a comprehensive and powerful online computer store. The intended users of this system are the computer store's staff and customers. An Object Oriented Design is used to develop the project with a focus on items such as code re-use and objects. With the provided Web Site that customers interact with, and the employee application that the employees use to interact with the system, I will attempt to incorporate many of the features that are common to most Web order systems.

Table of Contents:

Table of Figures:

2. System Overview

This section describes all the constraints on and functions performed by the system to define the project fully for the reader. It lists all the functions performed by the system and the constraints under which it is to operate. This section consists of the following sections:

Section 2.1: This section provides an overview of the top-level view of the system architecture.

Section 2.2: This section provides an overview of the system functions.

Section 2.3: This section describes all the elements involved in a Web Store Front.

2.1 Top-Level View of System

This project used a three-tier framework to build the e-commerce Website. The first tier consists of multiple clients connected to the enterprise system. The second tier is comprised of the business and program logic of the system, incarnated by a Web Server. The third tier was comprised of an Oracle database server and the database store. The components are described below. The deployment diagram of the Web Store Front shows the physical layout of the system. It shows that the system consists of three major system components or tiers. See Figure 1.

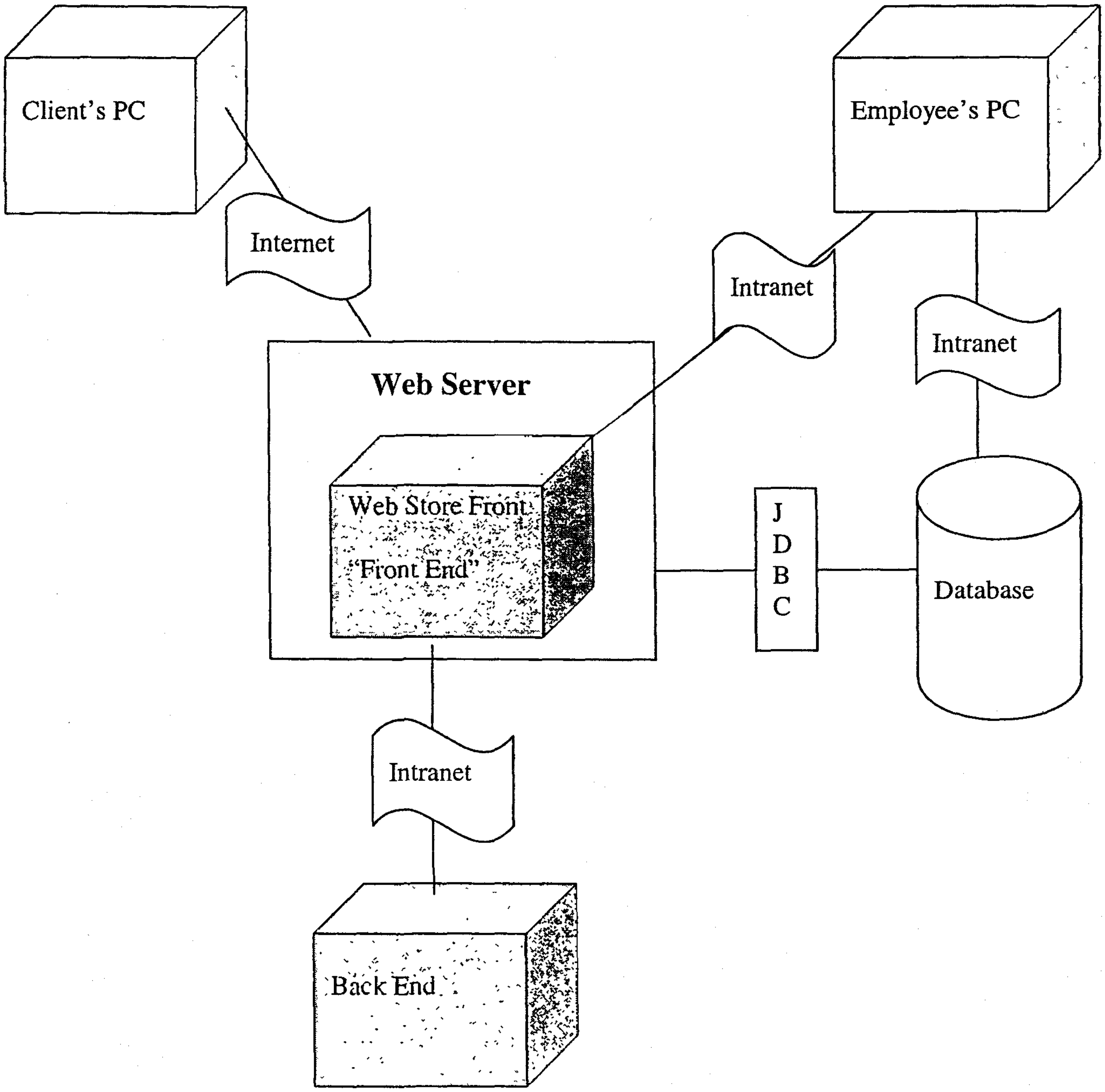

Figure 1- Deployment Diagram

- First Tier

The first tier consists of Web browsers that are located on remote computers. The computers are allowed to connect to the enterprise system through the Internet. This creates a server-client system in which many clients can be served by a single or a cluster of servers. The first tier is connected to the entire system through the second tier. Specifically, Web Browsers make requests to the Web Server by use of the Hyper Text Transmission Protocol. The Web Server receives the requests and returns the requested information. The first tier is for the client, and contains the presentation logic. This includes simple control and user input validation, which is implemented with HTML and JavaScript.

- Second Tier (Middle Tier)

The second tier "Front End" is also known as the application server, and provides the business process logic and the data access. It includes the Web Server business logic. The Web Server, which is a Tomcat Web Server, is responsible for waiting and for answering requests from the Clients (Web Browsers). The Front End can be thought of as containing business logic and programming, that allows a sequence of decisions to be made such as buy, modify account, and view orders. In addition, it is vital in allowing customers to interact (i.e., place an order) with the Web Store database. JDBC API (Java Database Connective, Application Programming Interface) has two parts, an application-level interface used by the Front End component to access a database, and a service provider interface to attach a JDBC driver to the java based. The Front End also interacts with the Back End, which is a simulation for an external system and has an interface within the Front End. The main goals of the Back End are to process the purchased items from the store and ship items to the customer.

- Third Tier

The third tier consists of the Oracle database server and the oracle database. The oracle database and database server are connected to the second tier, and contain permanent information about orders, inventory and customer's information.

The system is built on a three-tier architecture. The advantages of this approach are:

- It is easier to modify or replace any tier without affecting the other tiers.
- Separating the application and database functionality means better load balancing. It is more scalable and easier to control.
- Adequate security policies can be enforced within the server tiers without hindering the clients.
- The middle tier enables the system to handle more client connections.
- It implements better security and provides easier maintenance.
- It is more scalable and easier to control.

2.2 System Functions

The system provides various functions for the customers and store's staff. The system also differentiates between the registered customer and the unregistered customer. Following is a brief description of each implemented use case as the system was initially planned.

Functions of the customers:

- **Sign On:** The registered customer accesses the sign on page at a set URL, provides his/her username and password, and clicks on "submit." This takes customer to the main interface screen for the registered customers.
- **Create Account**: The unregistered customer accesses the create account at a set URL, fills the account form, and clicks on "submit." This takes the customer to the main interface screen for the registered customer.
- **Change Password**: The registered customer accesses the change password at a set URL, provides his/her username and new password, and clicks on "submit." This takes the customer to another interface screen for either confirmation of the new password or to disprove the password.
- **Modify Account**: The registered customer accesses the modify account at a set URL, provides his/her username, password, and new information, and clicks on "submit." This takes the customer to another interface screen for either confirmation of the modification or disproves it.
- **Search Catalog:** The customer accesses the search catalog page at a set URL, provides key word to search for a particular item in the store, and clicks on

"submit." This takes the customer to an interface screen with the list of the items under that key word if the items are found.

- **View Catalog:** The customer accesses the view catalog page at a set URL, selects a catalog from a drop-down list, and clicks on "submit." This takes the customer to an interface screen with the list of the products under that catalog.
- **Buy:** The registered customer accesses the buy items at set a URL, selects items to buy using either search catalog function or view catalog function, places the selected items into the shopping cart, clicks on "submit." order, and provides the necessary information to process order. This takes the customer to another interface screen for either confirmation of the order or to disprove the order.
- **View Orders**: The registered customer accesses the view orders page at a set URL, provides the order number, and clicks on "submit." This takes the customer to another interface screen with the order information.

Functions of the Employee (store's staff):

- **Delete Items**: The employee accesses the database, selects existing items to delete, and clicks on "delete." button. The system deletes the selected items and updates the inventory number on the database.
- **Add Items:** The employee accesses the database, adds new items, and assigns a product number for the new items and clicks on "add." button. The system automatically creates the items records.

- **Update Items:** The employee accesses the database, provides the item's ID and NAME, provides new description on the selected items, and clicks on "update." button. The system updates the item's information in the database.
- **Find Items:** The employee accesses the database, provides the item's ID, and clicks on "find." button. The system searches for the selected item and displays the selected item information.

  View Orders: The employee accesses the database, requests the view orders. The system displays a list of the selling orders from the database.

2.3 Software Components

This section introduces the software components and the architecture involved in the Web Store Front.

2.3.1 Servlets

After examining the different types of architecture available, I decided to use object-oriented Servlets through Java to develop the Web Store. The following is a list of benefits to java Servlets:

- **Object Oriented**

This project is built by using an object-oriented language so the software is reusable, extensible and maintainable. This means that efforts need not be duplicated when writing a function that makes similar calls. Writing in an object-oriented language allows the programmer to develop code that performs the same function on different data sets, by specifying the data set as an argument. In fact, many of the Servlets in this project are based on an initial Servlet "plan". The contrast of object-oriented program is a single, flow-based program. The problem with a single, flow-based program is that the entire program must be loaded and run, and the entire logic must be traversed, to get to a minor function within the entire program. In this sense, object oriented programs are more advantageous.

- **Servlet based**

A Java Servlet is a Java Applet that runs on a server. Java Servlets run better than CGI scripts and programs because Java Servlets are persistent and can handle multiple requests. This means if two similar requests come one after the other, one Java Servlet can handle both of them. A CGI script can only handle one request before it is destroyed under a CGI script, the script must be compiled at run-time, executed, then destroyed. Then it must be compiled, executed and destroyed again for the second request. This shows that Servlets indeed have a better memory performance than CGI scripts so they are more advantageous.

- **Garbage Collection**

By running Java, the Java Virtual Machine automatically includes a garbage collector that sweeps Java's memory for unused memory blocks and returns them to a pool of available memory. This is not the case with CGI scripts or C and C++ programs, which increases the risk of memory leak. By using a Java-based language, garbage collection is advantageous because it increases the memory and thus overall performance of our program. This is important for the scalability of our enterprise model.

- **Efficient**

With traditional CGI, a new process is started for each HTTP request. If the CGI program itself is relatively short, the overhead of starting the process can dominate the execution time. With Servlets, the Java Virtual Machine is always and handles each request using a lightweight java thread instead of a heavyweight operating system process.

- **Portability**

The Servlet API takes advantage of the Java platform. It is a fairly simple API, which is supported by nearly all Web Srvers so that Servlets may be moved from platform to another platform, usually without any modification whatsoever.

2.3.2 Tomcat Server

The system is built on Tomcat 4.0, which is developed in an open and participatory environment and released under Apache. It is the official reference implementation of the Servlets 2.2 and JSP 1.1 specification. It can be used as a small stand-alone server for testing Servlets and JSP pages, or can be integrated into the Apache Web server. Tomcat, like Apache, is very fast and highly reliable.

2.3.3 Oracle 8i Database

The system database is built by using Oracle8i personal edition database server and the Oracle8i personal edition database to connect to the second tier, and contains permanent information about the store. Since this system is a part of the learning process, this release of Oracle8i was efficient, reliable, and secure for the project.

3. Architectural Design

The system architectural design was designed in two components. It specifies the design entities that collaborate to perform the functionality of the system. Each of these entities has an Abstract Specification and an Interface that expresses the services that it provides to the rest of the system. In turn each design entity is expanded into a set of lower-level design units that collaborate to perform its services. For examples classes, please view Appendix C. Also all tables referred to in this section can be found in the Appendix B.

The two main components of the system architectural are:

- The Front End Architecture
- The Back End Architecture

3.1 Front End Architecture

This component describes two separate parts located in the Front End. The first part is the Web form that the client interacts with. The second part is the employee form that the employee uses to interact with the system. See Figure 2, and 3.

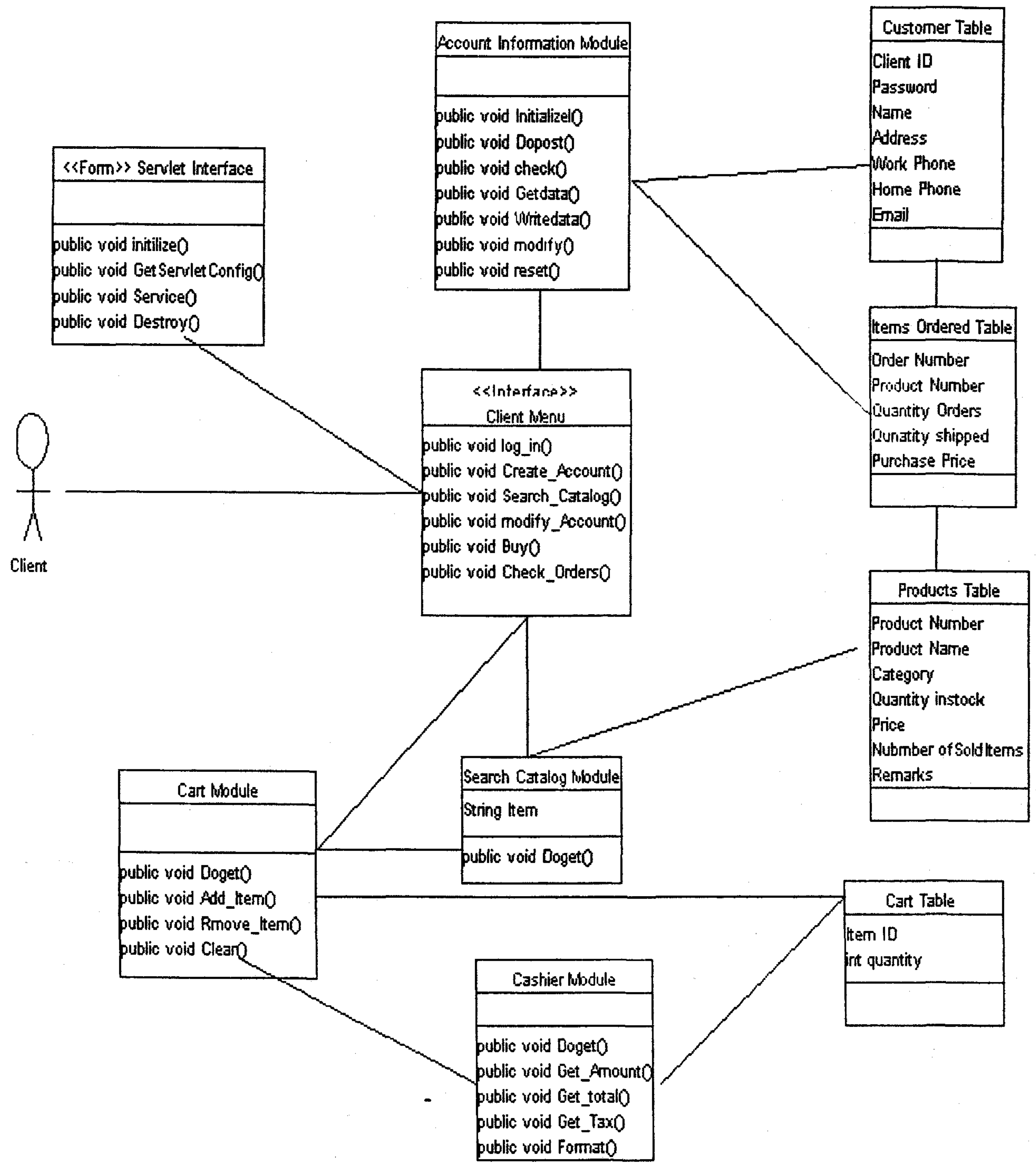

Figure 2 - Front End Architecture Design (Client)

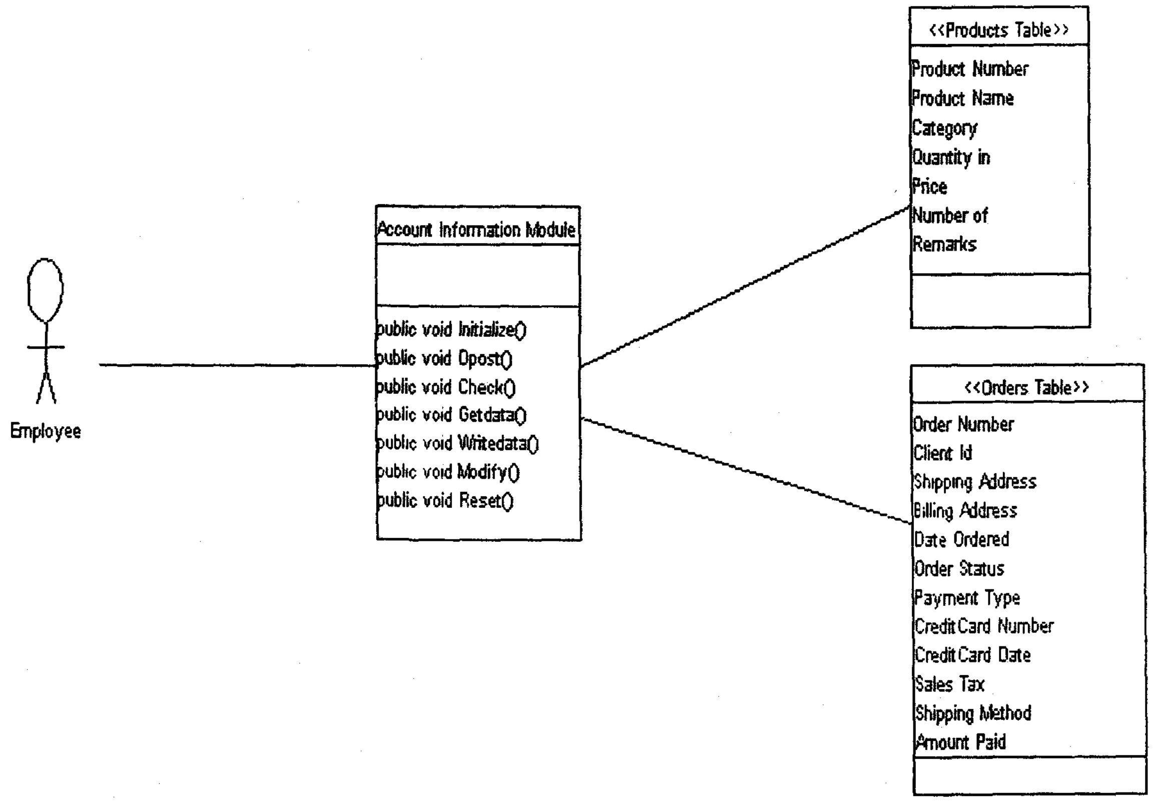

Figure 3 - Front End Architecture Design (Store)

3.2 Back End Architecture

The Front End invokes the Back End to process the purchasing after validation of the client's credit card and the cards expiration date. The Front End sends the clients personal information to the Back End. The Back End then processes the order, remarks that the item is shipped, and notifies the client through the Front End that the order has been shipped. See Figure 4.

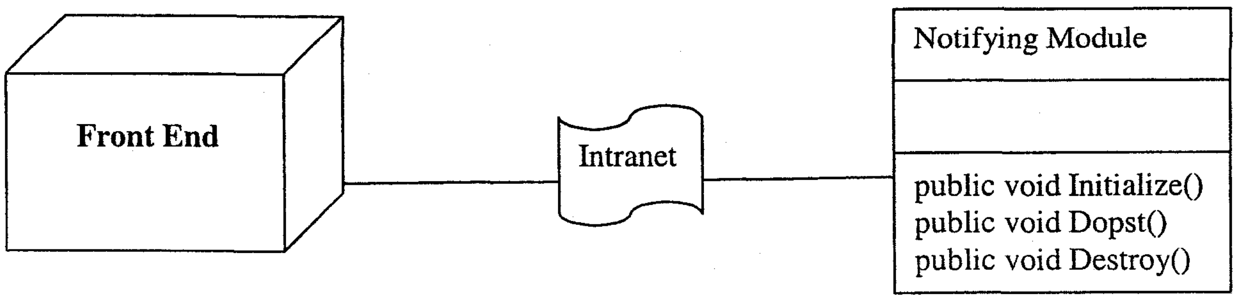

Figure 4 - Back End Architecture Design

3.3 List of Servlets

The following is a list of the Servlets that this system utilizes for the Front End and the Back End.

- **AddToCookie**

The function of this Servlet is to add an identifier id to the user so the commerce system can keep track of his/her shopping cart. This is required because every user has a distinct identifying ID so that items would be placed into the personal shopping cart of each customer.

- **ClientRegServlet**

The function of this Servlet is to register the client. The information is entered from a http form POST. The information is returned to this Servlet and entered in the database.

- **DeleteCookie**

This Servlet deletes the cookie from the user's computer so he can logout and another user can log in or register.

PlaceOrder

This Servlet sends the shopping cart items to the order fulfillment logic. The order fulfillment logic proceeds to insert these items into a table of "placed orders" and sends notification to order fulfillment to pack and ship the orders.

- **SearchPro**

This Servlet is used when the customer wants to view specific catalogs. The Servlet makes a connection with the database and performs a SQL query. The result set is returned to the tomcat server and displayed in a table.

- **ValidateUser**

This Servlet is used when the user registers or when the user checks out. A simple SQL query is performed to make sure the user is registered and valid. If not, the user is returned to a page where he can register.

- **ViewOrderServlet**

This Servlet is used to allow customers to see their current order. The Servlet makes a call to the database and performs a query which returns a result set of items that the customer has ordered. It is displayed by the Java Servlet into a table.

- **ListProdcuts**

This Servlet is used to allow customers to search for a specific item. The Servlet makes a connection with the database and performs a SQL query. The result set is returned to the tomcat server and displayed in a table.

- **ModifyAcc**

This Servlet is used to allow the customer to modify an existing account. A simple SQL query is performed to make the modification in the database. The modified information is returned to the Tomcat server and displayed on the screen for the customer.

- **ChangePassword**

This Servlet is used to allow the customer to change his/her password. A simple SQL query is performed to make the changes in the database. The new password is returned to the Tomcat server and displayed on the screen for the customer.

4. System Code

The code for this system is implemented in two ways, the client side code and the server side code. On the client side are the HTML pages that comprise the user interface and the JavaScripts codes that makes the interface more user-friendly and reliable. On the server side, the Servlets provide a framework for creating applications that implement the request/response between the client "Web Browser" and the Tomcat server.

This code is provided to show the life cycle of Sign On function on client side and on server side. On client side, this code is part of The HTML page for signing on the Web Store Front, which is "login.html." The client provides his/her username and password, and clicks on "submit" button. The Servlet, which is invoked on this HTML page, is ValidateUser. When the user clicks on "submit," the HTML tells the server to do the action "post" on Servlet VaildateUser.

```
(Login.html )
<body>
<font face="verdana" size="2"><b>Online Computer Store</b>
<p>
<H5>If you are a new user, Please</H5><H1><a href="register.html"> Sign
up</a></H1> <u>Login</u>
</p>
<form  action=http://localhost:8080/Servlet/ValidateUser method="post"
on"submit"="return validate();">
<table cellpadding="1" cellspacing="1" border="0">
<tr><td><font face="verdana" size="2">Name</font></td><td><input type="text"
name="loginid"></td></tr>
<tr><td><font face="verdana" size="2">Password</font></td><td><input
type="password" name="loginpass"></td></tr> </table>
<input type=""submit"" value=""submit"">
<input type="reset" value="Reset">
</form>
</body>
```

On the server side, there is the ValidateUser class. I need to define our class. I also need the javax.Servlet package, which contains the interfaces and the classes intended to be protocol independent ,and the javax.Servlet.http package whichcontains HTTP specific interfaces and classes.

```
import javax.Servlet.*;
import javax.Servlet.http.*;
public class ValidateUser extends HttpServlet {

        Code...

}
```

Servlets initialize the store database with initialization method; the database name is declared as Webstore. A Servlet can read this argument from the ServletConfig at initialization and then later on uses it to open a connection to the database while processing a request. The username and password of the database administrater is “store”

```
private String URL = "jdbc:odbc:WEBSTORE";

public void init( ServletConfig config )
      throws ServletException
     { super.init( config );
         try {
       Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
      connection = DriverManager.getConnection( URL,"store", "store" );
              }
         catch ( Exception e ) {
              e.printStackTrace();
              connection = null;
            }
        }
```

To send response to the client, the dopost method is invoked. This method is declared here.

```
        public void doPost( HttpServletRequest req,
        HttpServletResponse res )
        throws ServletException, IOException
       {
          String  clientid, password ;


          clientid = req.getParameter( "loginid" );
          password = req.getParameter( "loginpass" );
          PrintWriter output = res.getWriter();
          res.setContentType( "text/html" );
               ……. Code
        }
```

To keep track of the customer username and password on the system during using the Web Store Front, cookies are created.

```
        if ( success )
              {

              // Create a [illegible]e
                   Cookie c = new Cookie("storecookie" ,clientid );
                   c.setMaxAge( 3600 );  // seconds until co[illegible] removed
                   res.addCookie( c );  // must precede getW[illegible]

              }
```

العنوان:	Web store front
المؤلف الرئيسي:	Al Ahmari, Saad A.
مؤلفين آخرين:	Martin, Dennis(Super.)
التاريخ الميلادي:	2002
موقع:	سكرانتون، بنسلفانيا
الصفحات:	1 - 41
رقم MD:	617982
نوع المحتوى:	رسائل جامعية
اللغة:	English
الدرجة العلمية:	رسالة ماجستير
الجامعة:	University of Scranton
الكلية:	College of Arts and Sciences
الدولة:	الولايات المتحدة الأمريكية
قواعد المعلومات:	Dissertations
مواضيع:	هندسة البرمجيات، المتاجر الالكترونية، شبكة الانترنت
رابط:	https://search.mandumah.com/Record/617982

Bibliography

1) Saad Alahmari, "Software Requirements Specification Document for Web Front Store."

2) Saad Alahmari, "Software Design Description for Web Front Store."

3) Saad Alahmari, "Test Design Document for Web Front Store."

4) Project Documentation Standards [online]. Dr. Dennis S. Martin http://www.cs.scranton.edu/~dmartin/dsindex.html.

5) Inside Servlets, Server-Side Programming for the Java Platform, by Dustin R. Callaway, second edition, 2001.

6) Core Servlets and Java Server Page, by Marty Hall, Java 2 Platform, Enterprise Edition Series, 2000.

7) Core Java volume I-Fundamentals, by Cay S. Horstmann & Gary Cornell, 1999 Sunmicrosystems, Inc.

8) The J2EE Tutorial [online]. http://java.sun.com/j2ee/tutorial/.

9) Documentation bundle for the Tomcat 4 Servlet/JSP container [online]. http://jakarta.apache.org/tomcat/tomcat-4.0-doc/index.html.

Index

Appendix A: Definitions

Browser: A Client program (software) that is used to look at various kinds of Internet resources.

CGI -- (Common Gateway Interface): A set of rules that describe how a Web Server communicates with another piece of software on the same machine, and how the other piece of software (the CGI program) talks to the Web server. Any piece of software can be a CGI program if it handles input and output according to the CGI standard.

Client-Side: A term used to define technology or processing that occurs on the "client" or user side of the network connection

Cookie: The most common meaning of "Cookie" on the Internet refers to a piece of information sent by a Web Server to a Web Browser that the Browser software is expected to save and to send back to the Server whenever the browser makes additional requests from the Server.

HTML -- (HyperText Markup Language): The coding language used to create Hypertext documents for use on the World Wide Web. HTML looks a lot like old-fashioned typesetting code, where you surround a block of text with codes that indicate how it should appear

Intranet:A private network inside a company or organization that uses the same kinds of software that you would find on the public Internet, but that is only for internal use. Compare with extranet.

Internet (Upper case I): The vast collection of inter-connected networks that are connected using the TCP/IP protocols and that evolved from the ARPANET of the late 60's and early 70's.

Java: Java is a network-friendly programming language invented by Sun Microsystems. Java is often used to build large, complex systems that involve several different computers interacting across networks, for example transaction processing systems.

Java Database Connectivity (JDBC): An application programming interface (API) developed by Sun Microsystems, Inc. and various partners and vendors. JDBC possesses the same attributes as Open Database Connectivity (ODBC) but is specifically developed for use by Java database programs. Furthermore, for databases that do not have a JDBC driver, JDBC includes a JDBC-to-ODBC bridge for converting JDBC to ODBC; the bridge presents the JDBC API to Java database applications and converts this to ODBC.

JVM: An acronym for Java Virtual Machine, which behaves as an interface between compiled Java binary code and the microprocessor (or "hardware platform") that actually executes the application's instructions. Once a platform has been provided with a Java virtual machine, any Java application (which, after compilation, is called bytecode) can operate on that platform.

Server: A computer, or a software package, that provides a specific kind of service to client software running on other computers. The term can refer to a particular piece of software, such as a WWW server, or to the machine, on which the software is running, e.g. "Our mail server is down today, that's why e-mail isn't getting out."

Server-side: A term used to define technology or processing that occurs on the "server" or nonuser end of the network connection.

SQL -- (Structured Query Language): A specialized language for sending queries to databases. Most industrial-strength and many smaller database applications can be addressed using SQL. Each specific application will have its own slightly different version of SQL implementing features unique to that application, but all SQL-capable databases support a common subset of SQL.

SSL -- (Secure Socket Layer): A protocol designed by Netscape Communications to enable encrypted, authenticated communications across the Internet.

URL -- (Uniform Resource Locator): The term URL is basically synonymous with URI. URI has replaced URL in technical specifications

SRS: Software Requirement Specification

SDD: Software Design Description

Appendix B: Database

Orders

Field Name	Data type	Length	Primary key	Foreign Key	Value Range
Order Number	Integer	15	YES	NO	Not Null
Client Id	String	10	NO	YES	Not Null
Shipping Address	String	500	NO	NO	Not Null
Billing Address	String	500	NO	NO	Not Null
Date Ordered	Date	10	NO	NO	Not Null
Order Status	String	15	NO	NO	"not-shipped", "shipped", "pending"
Payment Type	String	10	NO	NO	Not Null , "cod", "creditcard"
Credit Card Type	String	20	NO	NO	"Mastercard", "Visa", "AmericanExp"
Credit Card Number	Integer	20	NO	NO	
Credit Card Date	Integer	6	NO	NO	Year, Month ie:200104
Sales Tax	Double	6	NO	NO	Not Null
Shipping Method	String	20	NO	YES	Not Null
Amount Paid	Double	10	NO	NO	Not Null

Products

Field Name	Data type	Length	Primary key	Foreign Key	Value Range
Product Number	String	15	YES	NO	Not Null
Name	String	150	NO	NO	Not Null
Category	String	15	NO	NO	Not Null
Quantity in Stock	Integer	3	NO	NO	Not Null
Price	Integer	10	NO	NO	Not Null
Number of Sold items	String	15	NO	NO	Not Null
Remarks	String	500	NO	NO	

Items Ordered

Field Name	Data type	Lengt h	Primary key	Foreign Key	Value Range
Order Number	Integer	15	YES	YES	Not Null
Product Number	String	15	YES	YES	Not Null
Quantity Ordered	Integer	3	NO	NO	Not Null
Quantity Shipped	Integer	3	NO	NO	Not Null
Purchase Price for each item.	Integer	10	NO	NO	Not Null

Customers

Field Name	Data type	Length	Primary Key	Foreign key	Value Range
Client Id	String	10	YES	NO	Not Null
Password	String	10	NO	NO	Not Null
Name	String	50	NO	NO	Not Null
Address	Integer	500	NO	NO	Not Null
Work Phone	String	20	NO	NO	
Home Phone	String	20	NO	NO	
Email	String	50	NO	NO	

Shipping

Field Name	Data type	Lengt h	Key	Value Range
Shipping Method	String	20	Primary	Not Null
Shipping Costs for the order.	Integer	10	No	Not Null

Appendix C: Architectural Design

These are examples for Servlets in the project.

Name	Servlets Interface
Type	This module is required as System Software for the Servlets initialization
Description	This interface defines methods to initialize a Servlet, to service requests, and to remove a Servlet from the server. When the client accesses to the URL, the client menu invokes the Interface Servlets to initialize the Servlets
Operations	***Initialize*** Arguments: Config - a ServletConfig object containing the Servlet's configuration and initialization parameters. **No Return value** Pre-condition: the Servlets are constructed. Post-condition: The Servlet is initialized and ready to receive any requests. Exception: Throws a ServletException, defines an exception that a Servlet or filter throws to indicate that it is permanently or temporarily unavailable. Also if the initialization does not return within a time period defined by the Web server. Flow of Events: 1. The client requests a service form the Web Store front. 2. The Servlet container calls the Servlet to indicate that the Servlet is being placed into service. 3. The initialization takes place. This is class must be complete before any requests from the client. This software is required for every Servlets container. ***GetServletConfig*** No Arguments Return: the ServletConfig object that initializes this Servlet. Pre-condition: the Servlets are initialized. Post-condition: the Servlets are configured. No Exception Flow of Events: 1. The initialize method invokes the GetServeletConfig. 2. The GetServeletConfig method invokes the GenericServlet class

in the Servlets library.
3. The GenericServlet class, which implements this interface, already does this

This class is required for every Servlets container. It is outside the module.

Service
Arguments: req - the ServletRequest object that contains the client's request
res - the ServletResponse object that contains the Servlet's response
No Return value
Pre-condition: the Servlets are initialized.
Post-condition: the Servlet is able to respond to a request.
Exception: ServletException - if an exception occurs that interferes with the Servlet's normal operation. Also java.io.IOException - if an input or output exception occurs
Flow of Events:

1. The client uses the URL to access the Web site.
2. The Servlets are initialized.
3. The client requests a service form the Web site.
4. The Servlet container invokes the service method.

This class is required for every Servlets container.

Destroy
No Arguments
No Return value
Pre-condition: the Servlets are initialized.
Post-condition: the Servlet is being taken out of service.
Exception: This method gives the Servlet an opportunity to clean up any resources that are being held (for example, memory, file handles, threads) and make sure that any persistent state is synchronized with the Servlet's current state in memory
Flow of Events:

1. The Servlet container invokes the Destroy method.
2. The Destroy method checks if there are held resources.
3. The Destroy method destroys the initialized Servlets

Name	**Client Menu**
Type	**Form.**
Description	This is the Client Menu From provides the client within the primary functions to navigate the Web site. It allows the client to log into the system if the client has an account, to research the catalog, to create account, to buy, to check orders, to modify account and to exit the Web site.
Operations	**Log in** The client logs into the system. The client has full privileges to search catalog, to buy, to check orders, to modify personal account. Exception: None Flow of Events: See Log In Use Case Realizations, section **Create Account** No arguments. No return value. Pre-condition: None. Post-condition: the client has an account. Exception: None Flow of Events: See Create Account Use Case Realizations, section **Search catalog** No arguments. Return value: found items. Pre-condition: None. Post-condition: the client finds the requested item. Exception: if the item is not available in the database, the system raises an exception that reveals the item is not found. Flow of Events: See Search Catalog Use Case Realizations, section **Modify Account** No arguments. No return value. Pre-condition: the client has an account. Post-condition: the client modifies the personal information. Exception: None Flow of Events: See Modify Account Use Case Realizations, section **Buy** No arguments. No return value.

	Pre-condition: the client has an account. Post-condition: the client buys items from the Web Store Front. Exception: None Flow of Events: See Buy Use Case Realizations, section **Check Orders** No arguments. No return value. Pre-condition: the client has an account. Post-condition: the client checks the purchasing orders those the client made from the Web Store Front. Exception: None Flow of Events: See Check Orders Use Case Realizations, section

Name	**Account Info Module**
Type	Code Module
Description	The client uses the log in service to log into the system that it allows the client to navigate the Web site as a customer with full privileges.
Operations	**Initialize form** No arguments. No return value. Pre-condition: the Servlets are initialized Post-condition: the log in form is initialized No Exception: Flow of Event: 1. The client access to the URL 2. The system initialize the servetls 3. The client requests the log in method 4. The Servlets invokes the log in form. 5. The log in initialization takes place **Dopost** Arguments: HttpServletRequest req, HttpServletResponse resp) No return value Pre-condition: The form is initialized, and the client has an account. Post-condition: The client logs into the Web store Front No Exception: Flow of Events: 1. The client accesses the URL

2. The client requests to log in
3. The system displays to prompt the log in
4. The client provides the user name and the password
5. The system invokes the check method to check the user name and the password in the customer table in the database
6. The system displays that the log in is successful.

Check

Arguments. Client username, client password
No return value.
Pre-condition: the client requests log in
Post-condition: the client information is checked
No Exception: if the client has no account, the system raises an exception that the client has to create account.
Flow of Event:

1. The Dopost method invokes the check method
2. The system invokes the customer table in the database through JDBC.
3. The system invokes getdata method to get the cline information if the client has an account.
4. The system verifies the client username, password.

Getdata

Arguments: the client information.
No Return value
Pre-condition: none
Post-condition: the client information returns to the check method from the customer table in the database
Exception: if the client has no account, the system raises an exception that the client is not existing.
Flow of Events:

1. The system invokes the getdata method.
2. The getdata invokes the customer table in the database.
3. The getdata method returns the client information to the method that has invoked it
4. The system destroys the database connection.

WriteData

Arguments: the client information

No Return value
Pre-condition: None.
Post-condition: the client information creates in customer table in the database

No Exception
Flow of Events:

1. The system invokes the Writedata method
2. The writedata invokes the customer table in the database.
3. The writedata method writes the client information to the customer table.
4. The system destroys the database connection

Create Account
No Arguments
No Return value
Pre-condition: None.
Post-condition: the client creates an account.
No Exception
Flow of Events:

1. The client requests to create account
2. The system displays method to display the create account form.
3. The client provides the required information.
4. The system invokes writedata method to write data to the customer table in the database through JDBC.
5. The system displays the client account has created

Modify Account

Arguments: the client information
No Return value
Pre-condition: the client has an account
Post-condition: the client information is modified in the customer table in the database
No Exception
Flow of Events:

1. The client requests to modify account.
2. The system invokes getdata method to displays the client information
3. The client provides the modifications.
4. the system invokes the writedata method to write the modifications to the customer table in the database
5. The system displays that the modifications take place in the client account.

Reset
No Arguments

	No Return value Pre-condition: None. Post-condition: the username and password fields are cleared **No Exception** Flow of Events: 1. The client provides username and password 2. The client request to reproved username or password or both 3. The system clears the username and password fields

THE UNIVERSITY OF
SCRANTON
A JESUIT UNIVERSITY

GRADUATE SCHOOL

UNIVERSITY OF SCRANTON
The Graduate School

The thesis of Saad Al-Ahmari

entitled "Web Store Front"

submitted to the Department of Computing Sciences

in partial fulfillment of the requirements for the degree of MS Software Engineering

in the Graduate School of the University of Scranton has been read and approved by the committee.

Dennis J. Martin
Dr. Dennis Martin

Charles Taylor
Professor Charles Taylor

[illegible]
Dr. Yaodong Bi

April 26, 2002
Date

SCRANTON, PENNSYLVANIA 18510-4632
(570) 941-7600 • FAX: (570) 941-5995

Web Store Front

Thesis

Saad Alahmari

Spring 2002

Submitted in partial fulfillment of the Requirements for the

degree of Master of Science in Software Engineering

University of Scranton