2004

Apparel Companies' Management System (APLAN)

Anjia Wang

APPAREL COMPANIES' MANAGEMENT SYSTEM (APLAN)

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Anjia Wang

March 2004

APPAREL COMPANIES' MANAGEMENT SYSTEM (APLAN)

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Anjia Wang

March 2004

Approved by:

[redacted]

Dr. Richard Botting, Committee Chair

Feb/4/2004

Date

[redacted]

Dr. David Turner

[redacted]

Dr. Kerstin Voigt

ABSTRACT

A web-based database system is an efficient way of managing and handling data communication between the users who are not in the same locations.

This project is an apparel company's management system that uses the facility of Internet and database. Via the system, the different type of users in apparel companies can insert, update or delete the information to related database tables, and then the users can easily view the reports through the created queries based on the information stored in database. The project also allows the users to send E-mails each other to inform the data creation or changes. Thus, APLAN will improve the management efficiency by sharing the up-to-date information and reducing the documentation transfers by faxes or phone calls.

To fulfill above requirements, the project is designed to choose MYSQL as the database system. A web browser is used to access the database system. JSP (Java Server Pages) is an interface between MySQL and web browser. The database access scheme is JDBC. The PostCast server is used as the SMTP mail server.

For the apparel companies who perform the manufacture activities, the project can serve as a model to handle Apparel

companies' plan and production control since such companies execute similar processes.

At last, the project may imply a wide marketing possibility in the garment area especially in China since it can help the similar companies to free from the heavy and detailed management works.

ACKNOWLEDGMENTS

I would like to thank Dr. Richard Botting to be my project advisor who strongly supports me in the project works. Under his guidance, I can successfully finish my project smoothly. Also I'd like to thank my committee members: Dr. David Turner and Dr. Kerstin Voigt for their valuable suggestions and helps.

I also want to thank for Dr. Josephine Mendoza for her patient teaching in CSCI572 and CSCI680. With the knowledge of database system from these courses, I can design the system easily. I also need to thank for all faculties in computer science department in CSUSB for their hard teaching in helping their students including me.

Finally, I'd like to thank for my family, especially my husband and my daughter. Without the support from my husband, I would never have a chance to return to the university. Also I must thank for my seven-years-old daughter. In order to support me, she must have been in the university's children center or stayed with me in the labs for many nights. I feel very proud of her support.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGMENTS v

LIST OF TABLES ix

LIST OF FIGURES x

CHAPTER ONE: SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction 1

1.2 Purpose of the Project 1

1.3 Context of the Problem 5

1.4 Significance of the Project 5

1.5 Assumptions 5

1.6 Limitations 6

1.7 Definition of Terms 6

1.8 Organization of the Project Report 7

CHAPTER TWO: SOFTWARE DESIGN

2.1 Introduction 9

2.2 Preliminary Design 9

2.3 Detailed Design 9

2.4 Software Design for Database 11

2.5 Software Design for Web 19

2.6 Software Design for Java Logic 22

2.7 Summary 25

CHAPTER THREE: SOFTWARE QUALITY ASSURANCE

3.1 Introduction 26

3.2 Unit Test 26

3.3 Integration Test Plan 31

3.4 System Test Plan 31

3.5 Summary 33

CHAPTER FOUR: MAINTENANCE

4.1 Introduction 34

4.2 Maintenance for Database 34

4.3 Maintenance for Web 36

4.4 Maintenance for Java Source 37

4.5 Summary 37

CHAPTER FIVE: USERS MANUAL

5.1 Introduction 39

5.2 Functions for Production Manager 39

5.3 Functions for Sales Staff 41

5.4 Functions for Branch Managers 43

5.5 Functions for Production Staff 44

5.6 View Pages 46

5.7 Summary 49

CHAPTER SIX: CONCLUSIONS

6.1 Introduction 51

6.2 Conclusions 51

6.3 Summary 53

APPENDIX A: DATABASE E-R DIAGRAM AND ENTITIES 55

APPENDIX B: SCREENSHOTS FOR USERS MANUAL 61

APPENDIX C: JAVE DESIGN 113

REFERENCES 145

LIST OF TABLES

Table 1. Definition of Terms 7

Table 2. Date Dictionary for Database of APLAN 15

Table 3. Unit Test for Classes 27

Table 4. Test Plan for System 32

Table 5. The Foreign Key Constraints of Database 36

LIST OF FIGURES

Figure 1. UML of System's Architecture 10

Figure 2. Web Site Use Case Diagram 20

Figure 3. Class Diagram for Production Process 24

Figure 4. Directory Structure of APLAN 35

CHAPTER ONE

SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction

Garment manufacturing companies have to do a great amount of production management. The main activities of such management are: confirming new orders, creating order reports, production reports, making schedules for the headquarters' and branch's productions. The main categories of user involving such works are the production manager, branch managers, sales staff and production staff.

Apparel Companies' Production Plan (APLAN) is such a management system that handles above garment manufacturing companies' main production activities in one system.

1.2 Purpose of the Project

The purpose of the project was to develop a web-connected system that assists the production management for an apparel company that has branches in different locations. The system assists to share company's common information and arrange the production well. APLAN will provide following functions:

1. Once the company has signed an order contract with a buyer, a production manager in the headquarters should confirm the new order. In the order confirmation, the

information regarding order's name, buyer's name, buyer's ID, buyer's order number, order's destination, order quantity and the expected shipment date should be input. At this time, a unique order number, which will be referred through the whole handling process, would be generated automatically. This information will be saved in the database system of APLAN.

2. The production manager can update, delete or view order confirmation at any time. All changes will be saved in the database system of APLAN.
3. The order confirmation also can be viewed by branch managers, sales staff and production staff.
4. After confirming a new order, a production manager will plan the order to a branch based on the branch's productivity, order quantity and the due date. At the same time, the production manager also needs assign a sales staffer to handle the order. All above information will be saved in the database system and automatically E-mail to the production manager in a branch and the handling sales staff in the headquarters.
5. A production manager can change or delete the order plan at any time. All changes also be updated in the database system and informed to the related sales staff

and branch managers. Sales staff and branch managers can view the headquarters' plan.

6. A production manager can view all orders' status by choosing the query type. Thus, if there is a problem with the order, she/he can try to negotiate the order with the customer as soon as he/she can. Orders' status also can be viewed by branch managers and sales staff.

7. In order to contact with the buyers or company staff conveniently, all uses in APLAN can check the contact list through the system. The contact list contains the name, address, phone number, fax number and web address (if any).

8. After assigning a new order to the sales staff, a sales staffer should input the order report. An order report consists of color/pattern name, color/pattern number, size, quantity of each pattern/color for each size, each expected shipment date and shipment quantity. The information should be saved in the database system and E-mailed to the production manager and branch manager.

9. The sales staff can view, change or delete the order report at any time. All changed information should be updated in the database of APLAN and sent to the production manager by E-mail.

10. Order report would be viewed by the production manager, branch managers and production staff besides the sales staff.

11. Once getting a new order from the headquarters, the branch manager should plan it to one or more production lines based on the productivity for each line, the due date and the order quantity. The system can assist the manager to evaluate the productivity of each line if the order's unit production time (min./pcs) is given.

12. The branch manager can view and change the production plan at any time to make sure that an order can be finished on time. If having some problems to arrange an order in the factory, the branch manager should E-mail to the production manager, then the production manager may adjust the order to another factory. The plan should be viewed by the production manager and production staff.

13. A production staff in the branch should enter a production report that records products' input and output day by day. If the actual output is less than the expected output, the production staff should warn the branch manager through E-mail. The information in production report should be saved in database of APLAN.

14. The production staff can delete, change or view the production report. The report also should be viewed by the production manager, branch managers and sales staff.

1.3 Context of the Problem

The context of the problem was a typical apparel company that has the branches in different locations.

1.4 Significance of the Project

The significance of the project is that it helps greatly improve the efficiency of production management in garment manufacturing companies by combining the companies' main production activities in one system. By sharing the common information from different departments or different branches, it sharply decreases the information transfer through phone or fax; also it will avoid many duplication works since the sharing information. The most importance is: since the web-based database management system always keeps the newly information, the related managers or staff can easily get the order status by the up-to-date information.

1.5 Assumptions

The following assumptions were made regarding the project:

1. There is a matched personnel system in such companies in order to fit APLAN's function design.

2. All data in database table of buyers has existed in database system. We assume that an administrator has entered those data through database command line.

1.6 Limitations

During the development of the project, a limitation was noted. The limitation is presented here.

Since there is no a standard management model for garment manufacturing companies, the management model used here may not be satisfy with all requirements for all such companies. But the system still will satisfy the main activities in the production management.

1.7 Definition of Terms

The following terms are defined as they apply to the project. It is described in table 1.

Table 1. Definition of Terms

Ant	Ant is a Java-based build tool which is extended using Java classes. Instead of writing shell commands, the configuration files are XML-based, calling out a target tree where various tasks get executed. Each task is run by an object that implements a particular task interface.
Java	A programming language introduced by Sun Microsystems. Java is a multiplatform, platform-independent, object oriented programming language. Java programs are not compiled, but rather interpreted as run. [1]
JDBC	Java Data Base Connectivity. A standard interface provided in Java for talking to different database formats such as SQL, Oracle and Microsoft Access. [2]
JSP	Java Server Pages. This is a technology defined by Sum Microsystems to create dynamic content on the Web. JSPs are a server-side application; they accept a request and generate a response. [3]
Java Servelets	These are Java classes that accept a request and generate a response. [4]
MySQL	An Open Source multi-user relational database. [5]

1.8 Organization of the Project Report

The project is divided into six chapters. Chapter one provides software requirements specification, an introduction to the context of the problem, purpose of the project, significance of the project, limitations, and definitions of terms. Chapter two consists of the software design. Chapter three documents the steps used in testing the project. Chapter four presents the maintenance required from the project. Chapter five presents the users manual from the

project. Chapter six presents conclusions drawn from the development of the project. The Appendices follows Chapter six. Finally, the references for the project are presented.

CHAPTER TWO

SOFTWARE DESIGN

2.1 Introduction

Chapter Two consists of a discussion of the software design. Specifically, database design, web presentation design and logic part between database and web presentation are three main topics. In this chapter, the three designs will be introduced in detail.

2.2 Preliminary Design

APLAN is a web-based database management system. The database system is accessed via Internet by using browser. The external interface contains a set of Java Server Pages (JSPs) that display data queried from a database. The interface between the MySQL database and JSP is the java logic parts connected by JDBC. The system's architecture is shown in figure 1.

2.3 Detailed Design

To support the functions in APLAN, Java logic code connects the JSP presentation interface to the database. In order to explain the detailed design, the product perspective would be introduced here. The web design, database design and logic design would be explained in

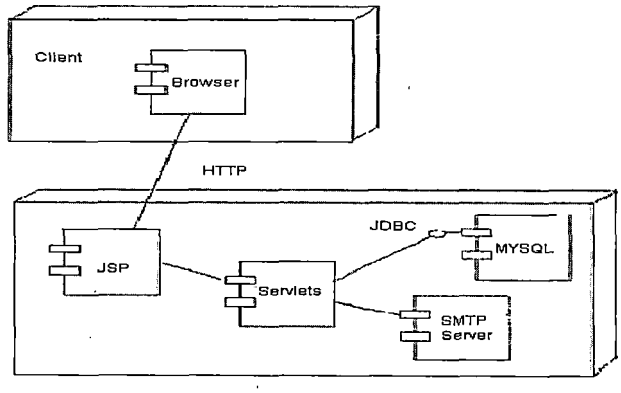


Figure 1. UML of System's Architecture

subsections of "Software Design for Web", "Software for Database" and "Software Design for Java Logic."

System Interfaces

The system requires an internet connected browser that can run Java, a web server capable of interfacing with a Servlet/JSP engine, and a database that provides a JDBC driver. A SMTP server used to send E-mail.

Software Interfaces

The operating system to develop this project is Windows 2000, the web server engine is Jakarta-tomcat-4.1.24, the java version is j2sdk1.4.1_02 and MySQL version is 3.23, the JDBC driver is mm.mysql-2.0.14, the SMTP mail server is PostCast server 2.4.95.

Following subsections of software design should explain the project in detail.

2.4 Software Design for Database

Database design is the most important part in the project since a perfect database system must satisfy the project's concerns with a clear and easily maintained structure. The database system should fit the project requirements and performance objectives such as order number, user id, and more.

The database system in APLAN will be introduced in following two subsections: description and data dictionary.

Description Database of APLAN

The database system can be expressed graphically by using the Entity Relationship (E-R) model. E-R diagram can refer to Figure 5. In this documentation, figures for database design including E-R diagram are arranged in APPENDIX A.

The entity properties for each table are indicated. The overview of APLAN will focus on the functionality of APLAN itself, so it doesn't include table of USERS, which used for authentication and authorization purpose. The detailed attributes in each entity are shown in APPENDIX A from Figure 6 to Figure 13.

Table of ORDERS. Each order contains order number, order name, buyer id, buyer's order number, unit time for production, order description, order destination, total

quantity and quantity unit, shipment quantity and shipment date. Shipment quantity and shipment date are composite attributes which include four times shipment quantity and date separately. Both order number and buyer's order number are unique keys. The project uses order number as the primary key in table orders. Total quantity is a derived entity derived from the sum of the four times shipment quantities. See Figure 6.

Table of COMPANIES of GROUP. For a group which has several branches, the table of companies_of_group contains the branch's name, the category (belongs to the headquarters or a branch), address, phone number, fax number, email address, web address (if any), productivity and the unit. Company name is unique and is used as primary key in this table. See figure 7.

Table of PRODUCTIVITY for LINE. Entity of PRODUCTIVITY_for_LINE is a weak entity since this entity is based on the entity of COMPANIES_OF_GROUP. The entity contains attributes of line number, line's productivity and unit. Line number is a weak key since it has unique number in one branch. The combination of the line number and branch's company name, which is the primary key in COMPANIES_OF_GROUP, is the primary key of this table. See Figure 8.

Table of HD PLAN. In a plan in the headquarters, one order will be assigned to any one branch in the group and any one sales staffer to handle it. So the entity of PLAN_IN_HEADQUARTERS is a weak entity based on entity of ORDERS, COMPANIES_OF_GROUP and SALES_STAFF. SALES_STAFF is one category from table of USERS. The entity itself has no more attributes. Here this table uses the combination of primary key from the above three tables as its primary key. So this table's primary key is order number, branch's company name and the sales staff's user Id. See Figure 9.

Table of BRANCH PLAN. In a branch, an order will be scheduled in the BRANCH_PLAN. The BRANCH_PLAN will be processed by LINES_IN_BRANCH. The entity also contains the shipment quantity, production begin date and production end date. Shipment quantity, production begins and ends date all are composite attributes which contain four times shipment quantity, four times production begins and ends date separately. The combination key from table ORDERS and LINES_IN_BRANCH is the primary key of table BRANCH_PLAN. They are Order number, branch's company name and line number. See Figure 10.

Table of ORDER REPORT. An order should be described in ORDER_REPORT. The entity contains pattern name, pattern number, size, first shipment quantity, second shipment

quantity, third shipment quantity and fourth shipment quantity. The chosen partial key for the entity is the combination of pattern number and size. Since ORDER_REPORT is the weak entity based on ORDERS, the primary key of the entity is combination of order number from entity ORDERS and the entity's its own weak key. They are order number, pattern number and size. See Figure 11.

Table of PRODUCTION REPORT. The production result for BRANCH_PLAN will be shown in the table of PRODUCTION_REPORT. The attributes of the entity are production date, production input and output and remark. The partial key is production date. The combination of the partial key and the key from BRANCH_PLAN is the entity's primary key. The primary key is the combination of order number, branch's company name, line number and production date. See Figure 12.

Table of BUYER. BUYER entity's attributes are buyer name, buyer id, buyer's address (street, city/town, state/province, zip and country), phone number, fax number, email address and web address. The primary key for this entity is buyer Id. See Figure 13.

Date Dictionary for Database

The data dictionary for database of the project is described in table 2:

Table 2. Date Dictionary for Database of APLAN

Table Name	Attributes	Description	Sample Data
ORDERS	orderno	Order number of an order	AR-1
	oname	Order's name	Men's long sleeve shirt
	buyerid	Buyer's Id for an order.	
	buyerono	Buyer's order number	
	utime	Unit production time for the order	
	odescribe	Description of the order.	
	destination	Destination for the order	
	totalqty	Order's total quantity	
	unit	Quantity's unit	Sets, Dozens
	efdate	Expected first order ship date	2003-02-01
	esdate	Expected second order ship quantity	
	esqty	Expected second order ship quantity	
	ethddate	Expected third ship date	
	ethdqty	Expected third ship quantity	
	efthdate	Expected fourth order ship date	
	efthqty	Expected fourth order shipment's quantity	

HDPLAN	ono	Order number	
	bname	Branch's company name	Acorn Woven LTD. (H.K.)
	salesid	User id for the handling sales staff	AM07
COMPANIESOF-GROUP	companyname	Company's name in group	
	category	Is branches or the headquarters	Branch
	street	Company's street	
	cityortown	City or town	
	Stateorprovince	State or province of the company	
	zip	Zip code	
	country	The country of the company	
	phone	Phone number	
	fax	Fax number	
	email	E-mail of the company	
	web	Company's web address	
	productability	Productivity per day	576000
	abilityunit	Unit of Productivity	Sec/dy
LINEABILITY	cname	Company's name	
	lineno	Number of line	
	lineability	Line's Productivity	
	aunit	Unit of Productivity	
BRANCHPLAN	bpono	Order number	
	bpcname	The branch's name that will runs the order	
	bplineno	The number of a line who will produces the order	

	bpfqty	First time finished quantity in a line	
	bpfbegindate	The fist time begin date to produce the order in a line	
	bpsenddate	The second time end date to produce the order in a line	
	bpsqty	Second time's finished quantity in a line	
	bpsbegindate	The second time begin date	
	bpsenddate	The second time end date to produce the order in a line	
	bpthdqty	The third time finished quantity in a line	
	bpthdbegindate	The third time begin date to produce the order in line	
	bpthdenddate	The third time end date to produce an order in line	
	bpfhtqty	The fourth time finished quantity in a line	
	bpfthbegindate	The fourth time begin date to produce the	

		order in a line	
	bpfthenddate	The fourth time end date to produce the order in a line	
ORDERREPORT	reportono	Order's number	
	patternname	Name of pattern name	Batik purple
	patternno	Number of the pattern	LSSCT0
	size	Size	XL
	reportfqty	Expected first ship quantity for a specific size and pattern in an order	
	reportsqty	Expected second ship quantity for a specific size and pattern in an order	
	reportthdqty	Expected third ship quantity for a specific Size and pattern in an order	
	reportfthqty	Expected fourth ship quantity for a specific Size and pattern in an order	
PRODUCTION-REPORT	prono	Order report's order number	
	prcname	The company's name that processes the order	

	prlineno	Number of a line who produces the order	
	printput	Production's input	
	proutput	Production's output	
	prdate	The date	
	prremark	Remark	
BUYER	buyername	Buyer's name	
	buyerid	Buyer's ID	
	buyerstreet	Street of buyer's address	
	buyercity	City/town for the address	
	buyerstate	State/province for the address	
	buyerphone	Buyer's phone number	
	buyerfax	Fax number for the buyer	
	buyeremail	Email address	

2.5 Software Design for Web

In APLAN, the users would access the database system via Internet from different locations. A structural web design can guide the users to maintain or process management works in an efficient way. Figure 2 is the Web Site Use Case Diagram. Based on the functionality for different categories of users, the Web design would be expressed in the user interfaces. Related pages introduced in the user interfaces can be referred to the screenshots in the users' manual (See Appendix B).

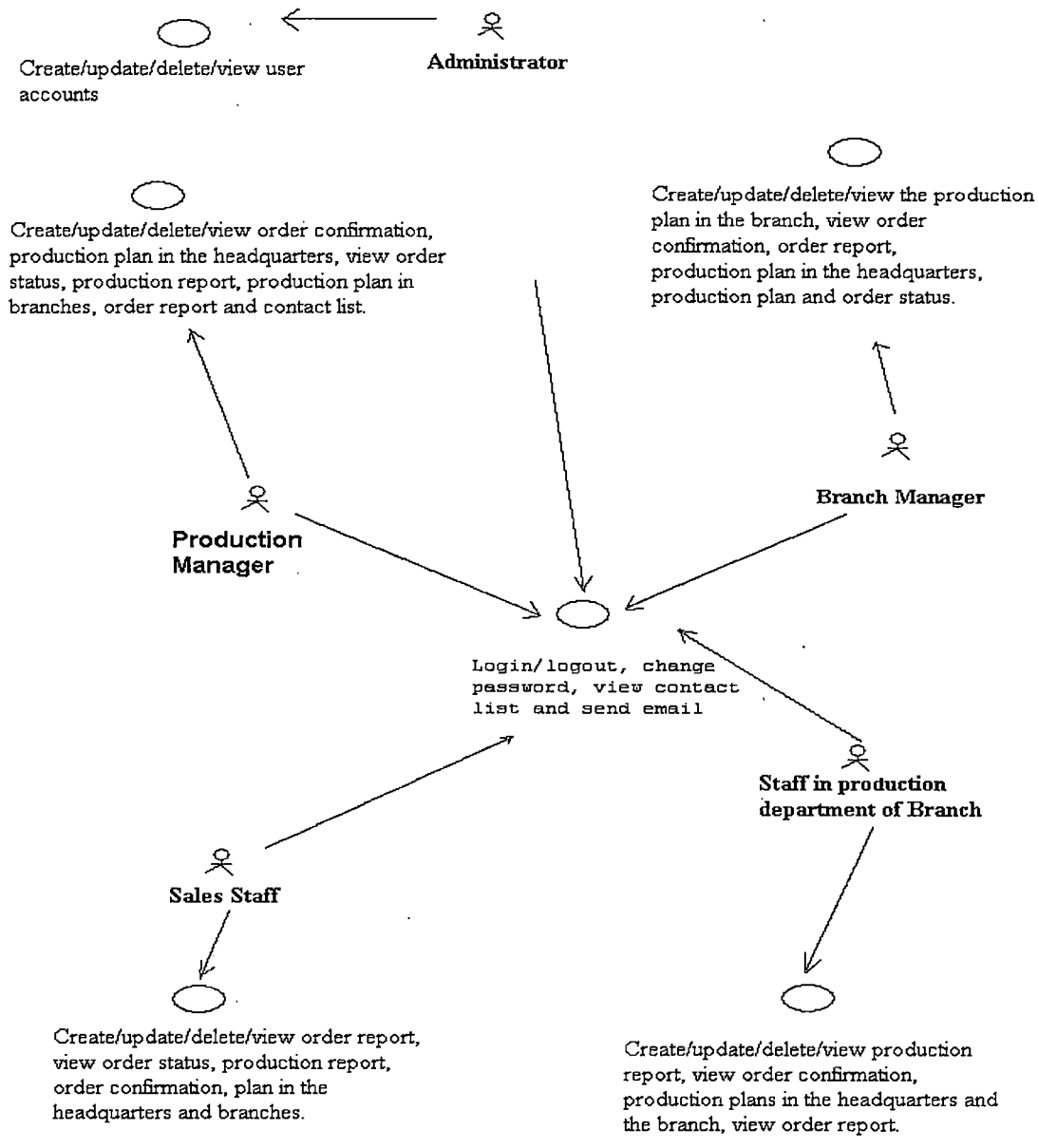


Figure 2. Web Site Use Case Diagram

Main Menu Page

The page guides the users to find their own menu based on the users' categories.

Menu Pages

These pages guide users to find out the function pages they want to access.

Pages to Enter Information

These pages ask the related users to enter the needed information. They include: 1) the page for an administrator to create users' accounts; 2) the pages for a production manager to enter order confirmation reports and the production plans in the headquarters; 3) the pages for a sales staffer to enter order reports; 4) the page for a branch manager to enter the schedules for the production in branch and 5) the page for a production staff to create production reports.

Pages to Update Information

The pages allow the users to update related information. These page are: 1) a page to ask an administrator to update users' information; 2) pages for the production manager to update order confirmation record and the headquarters' plan; 3) the pages for the sales staff to update the order reports; 4) the pages for the branch manager to update the branch production plans and 5) the

page for the production staff to update the production reports.

Pages to Delete Information

The users can delete the related information through the deletion functions. These functions are: 1) a page for the administrator to delete one user account; 2) pages for the production manager to delete one order confirmation record and one production plan in the headquarters; 3) pages for the sales staff to delete one order report record; 4) pages for the branch manager to delete one branch production plan record and 5) a page for the production staff to delete one production report record.

View Pages

After entering information into APLAN, it will generate several functional tables to allow the users to view the related information. These viewed tables are: 1) contact list tables; 2) order confirmation record; 3) order report; 4) production report; 5) the headquarters' production plan; 6) the branches production plan and 7) the order status.

2.6 Software Design for Java Logic

The connection between JSP and database system is fulfilled by JDBC. JDBC is the SQL-based database access interfaces from Java. Since the purpose of Java logic, the

design of Java classes here is based on the functionality of production managements. Java design would be express in the overview of class diagram, class description. Codes can refer to Append C.

Class Diagram

The class diagram of APLAN is introduced in Figure 3. In order to simplify the presentation of the class diagram, the attributes and operations of this class are hidden from the diagram. All relationships between any two classes are associations.

Class Description

APLAN's classes' descriptions are described as following:

Orders. The objects of maintaining table of Order Confirmations are in the class of Orders. The class contains the objects of creating, updating or deletion of order confirmation, finding an order confirmation with a specific order number. The objects to check the creating and updating order confirmation form are also included in this class.

Orderreport. Orderreport is a class to maintain the functions for order report. Except for the objects to maintain the table of order report, the object from class

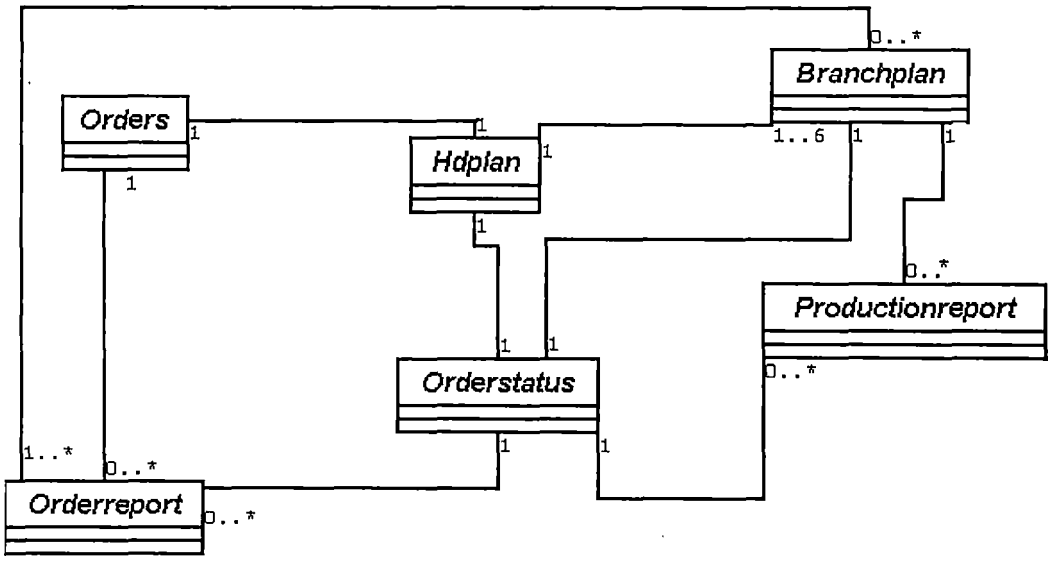


Figure 3. Class Diagram for Production Process

of Orders to find an order's confirmation with a specific order number is used.

Hdplan. Hdplan is used to maintain the headquarters' plans. Except for all objects to maintain the headquarters' plans, the class also contains the object to calculate the productivity for branches. In this class, the object from class of Orders to find if an order exists in order confirmation is used.

Branchplan. Class of Branchplan contains the objects to maintain the tables of branch plans and the object to calculate the lines' productivities. In this class, the object from class of Orderreport to find order report's total quantity in each shipment is needed. The class of Branchplan also needs the object from class of Hdplan to

find if a HD plan item with a specific order number and branch name exists.

Productionreport. Class of Productionreport contains the objects to maintain the tables of production report. The class needs the object from class of Branchplan to check if a specific item is listed in branch plans.

Orderstatus. The class of Orderstatus contains the objects to maintain the table of order status. This class needs to use the objects from classes of Orderreport, Hdplan, Branchplan, and Productionreport to find if an order exists in related statuses. The total quantitites in these statuses are needed.

2.7 Summary

The software design of the project was presented in Chapter Two. As you have seen, it contains three parts: web design, detabase design and logical design. The three parts are combined together to fulfill the functions of APLAN.

CHAPTER THREE

SOFTWARE QUALITY ASSURANCE

3.1 Introduction

Chapter Three documents the software quality assurance. Specifically, there are three main test stages: unit test, integration test and system test.

3.2 Unit Test

APLAN is a web-based application, so the units to be tested are mainly web pages plus the classes and methods hidden behind them.

Unit Test Plan for Web

In general, the process of unit test for web page is to test information and hyperlinks in each page. In this test, following criteria should be tested:

Graphics Check. In graphics check, we should make sure that the design of graphics satisfies the users' requirements and avoid any unexpected errors to view the page.

Text Check. In text check, the spelling error, text fonts, size and color should be checked.

Link Check. In unit test stage, hyperlinks should be checked to see if they are in the appropriate pages. Whether the hyperlinks go to the proper pages is not checked in this

stage since it will be checked in the stage of integration test.

Unit Test Plan for Classes

The unit test for main functions in each class is described in table 3:

Table 3. Unit Test for Classes

Class Name	Functions Tested	Expected Result
ORDERS	public static Orders findOrder(String)	If an order confirmation with a specific order number was found in database, the functions should return the object of ORDERS, otherwise, it should return null.
	public void insertOrders()	When an order confirmation is provided, an order item should be added in table of ORDERS in database
	DeleteOrders (String)	If an order confirmation with an order number were found, after running the function, the record would be deleted from table of ORDERS in database.
	public void updateOrder()	If order number and all update information are given, this order record should be updated to the table of ORDERS in database.
	public static String findUnit(String)	If an order with an order number is found, the quantity unit should be return, otherwise, it should return null.
	public boolean orderFormCorrect( )	If there is no error in the form to create or update ORDERS, the function should return true, otherwise, it should return false.

Order-report	public static Orderreport findOrderReport-Item (String, String, String)	Once the key for an order report item is given and this item is found, it should return the orderreport, otherwise, it should return null.
	public static boolean findOrder(String)	If find an order with a specific number, it will return true, otherwise, it should return false.
	public void insert-Orderreport()	When an order report is given, an order report item should be inserted in the table of ORDERREPOT.
	public void deleteOrderreport (String)	When an order report's order number is given, all order report items should be deleted from the table of ORDERREPORT in database.
	public void deleteOps(String, String, String)	If the key information for one order report item is given, this order report item should be deleted from table of ORDERREPORT.
	public void updateOrderreport (String, String, String)	Order report item should be updated in the table of ORDERREPOT in database.
	Public Boolean ifOrderreport-FormError()	If no error is in the form to create an item of ORDERREPORT, the function should return true, otherwise, it should return false.
	Public boolean ifUpdateOrder-reportFormError()	If no error is in the form to update an item of ORDERREPORT, the function should return true, otherwise, it should return false.
Hdplan	public static Hdplan findHdplanItem (String)	Once an order's plan with an order number is found, it should return this Hdplan's object.
	public void insertPlan()	A hplan item should be added into the table of HDPLAN in database.

	public void deleteHdplan (String)	Delete hdplan items with an order number from the table of HDPLAN in database.
	public void updatePlan()	An order's plan item of headquarters should be updated.
	public static double calculateAbility (String coeff, String unittime, String cname)	When the coefficient of productivity, branch name and unit time for a product are given, this function should return the branch's productivity.
Brnchplan	public static boolean findOrder(String)	When the branch plan items with an order are found, the function should return true, otherwise, it should return false.
	public static Branchplan getBpItem(String, String, String)	When one branch plan item with an order number, branch name and line number is found, the function should return an object of branchplan, otherwise, it should return null.
	public static double calLineAbility (String, String, String, String)	By giving an order, when the productivity coefficient, unit time for this order, branch name and line number are given, the function should return the order's productivity for each given line.
	public void insertBranchPlan( )	Insert a branch plan item into the BRANCHPLAN table.
	public void deleteBp(String)	All the branch plan items with an order should be deleted from the table of BRANCHPLAN in database.
	public void deleteBpItem (String,String)	A branch plan item should be deleted successfully from the table of BRANCHPLAN in database.
	public void updateBranchPlan( )	A branch plan item should be updated in the table of BRANCHPLAN in database.

	public Vector getDeleteBranch-PlanItems(String)	When an order number is given, the function should return a vector that has this order's plan items for all branches.
	public boolean checkBranch - PlanForm()	If the form to create the branch plan is correct, the function should return true, otherwise, it should return false.
	public boolean ifUpdateBpError()	If the form to update the branch plan is correct, the function should return true, otherwise, it should return false.
	public static boolean find_order(String)	If production report items with an order number exist, the function should return true, otherwise, it should return false.
Production -report	public static Productionreport findProductionrep ortItem(String, String, String, String)	Once a production report item with order number, branch name, line number and date are given, it will return this production report object. Otherwise, it will return null.
	public void insertProduction-Report()	If the information for a production report item is given, a production report item should be inserted into the PRODUCTIONREPORT table in database.
	public void deleteProduction-Report(String)	If an order number is given, its all production report items should be deleted from the table of PRODUCTIONREPORT.
	public void deleteProduction-ReportItem(String , String, String, String)	If order number, branch name, line number and date are provided, this item should be deleted from the table of PRODUCTIONREPORT.
	public void updateProductionR eport()	An item of production report should be updated in the table of PRODUCTIONREPORT.

	public static int TtlOutputInLine (String, String, String)	It should return the production report's total output with an order in a specific line.
	public static int TtlInputInLine (String, String, String)	It should return the total production report input for an order in a specific line.
	public boolean ifPrFormError()	If the form to create the production report is valid, it will return true, otherwise, it will return false.
	public boolean ifUpdateProductionReportError()	If the form to update the production report is valid, it will return true. Otherwise, it will return false.

3.3 Integration Test Plan

The process of integration testing is aimed at exposing problems that arise when two components are combined. The typical problem in this project focuses on whether hyperlinks go to a proper page.

These hyperlinks include the hyperlinks listed in home page and all users' main menus and all buttons in each page.

3.4 System Test Plan

System test should focuses on verifying the handling of valid input data, checking error handling, checking the output result.

For all functions in this project, all essential items list in pages should be entered. Once an error message is

prompted, the user can fix the problem based on the information in the message. In table 4, the main testing topics would be described.

Table 4. Test Plan for System

User's Category	Functions	Test Items
Production Manager	Create Order Confirmation/ Plan in the headquarters; Update Order Confirmation/ Plan in the headquarters	1.Enter correct order number to go to the table to create or update function. If the order number is wrong, a fail message should be seen. 2.Check all essential items should be entered. 3.After clicking the submit button, a success message should be got. Verify a record is entered or updated
Branch Manager	Create Branch Plan; Update Branch Plan	Similar checks as for thoses in Production Managers' create/update functions
Sales	Create/Update Order Report	Similar checks as for those in Production Managers' create/update functions. Check for: the sum of each shipment quantity in order report with the same order number can't be more than the corresponding each shipment quantity in the order's order confirmation.
Production Staff	Create/Update Production Report	Similar checks as for those in Production Managers' create/update functions. Make sure the total output can't be more than the total input for an order.
All Users	Deletion	After selecting the deleted items and clicking the delete button, a success page would be presented. Verify that the select items are deleted.

	View tables	Make sure after selecting the view type, and entering the correct information, the responding pages can be viewed.

3.5 Summary

To make sure the quality of project, testing is a very important work. From unit test to system test, we aimed at developing a trusted product for the users. Before integration test, each unit test items should be pass. System tests rely on the integration test. Once there are any changes, first we should do the unit test for the changed function, then do integration test, last do system test. To finish developing a project, these stages should be repeated constantly.

CHAPTER FOUR

MAINTENANCE

4.1 Introduction

This chapter is designed to help with the maintenance of the APLAN. It describes the organization of APLAN's code, data, and documentation. It explains how the system administrator can modify and recompile APLAN.

4.2 Maintenance for Database

Change Attributes of Tables

Once there are any changes for the database tables, the administrator should know where to find the files to maintain the database. In the structure of project (See Figure 4), the directory named C:/APLAN/databaseTextFile/sql, has all the .sql files to create the tables of APLAN. If there are any attributes to be added or deleted for a table, the administrator should first find the corresponding files in this directory.

In order to update the tables successfully, the administrator must pay attention on the integrity constraints in tables. When a table's attribute that works as a foreign key in other tables was deleted or changed, these affected tables also need do the corresponding

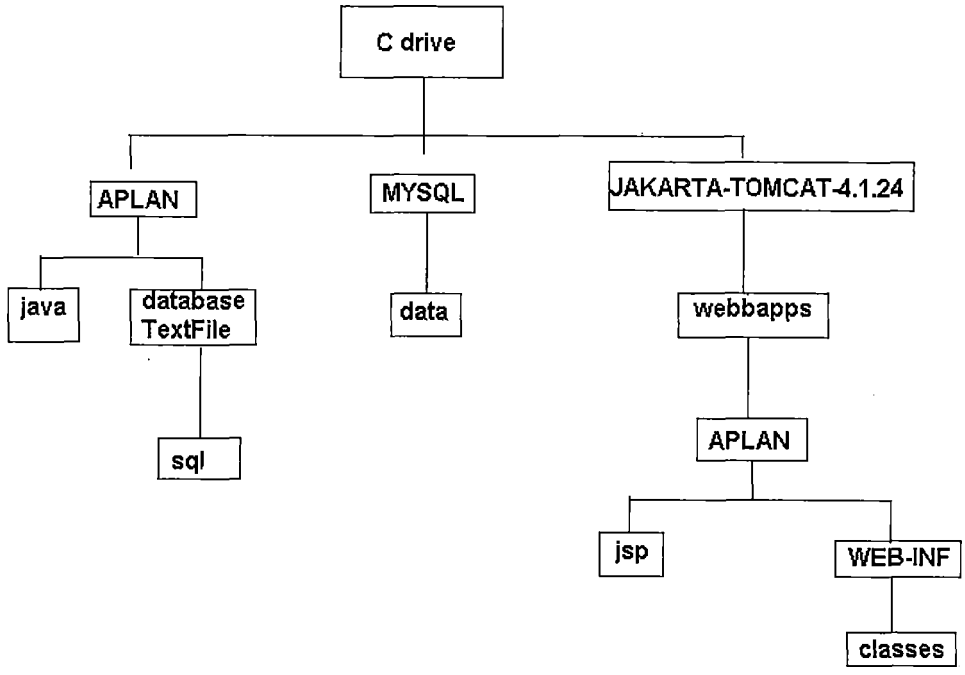


Figure 4. Directory Structure of APLAN

changes. This documentation lists all database foreign constraints for each table in table 5.

Maintain the Database Data

It is necessary for an administrator to keep saving the database's data files into the secondary disk once or twice a day. In directory "C:/mysql/data/aplan," all data files of APLAN are located there. The administrator can save these files into his/her secondary disks.

Table 5. The Foreign Key Constraints of Database

Table Name	Attribute Name	Tables which Use this Attribute as Foreign Key	Attribute Name in the Related Table
COMPANYS_OF_GROUP	companyname	USERS	cname
		LINEABILITY	cname
		PRODUCTIONREPORT	prcname
		HDPLAN	bname
		BRANCHPLAN	Bpcname
LINEABILITY	lineno	PRODUCTIONREPORT	prlineno
		BRANCHPLAN	bplineno
ORDERS	orderno	ORDERREPORT	reportono
		PRODUCTIONREPORT	prono
		BRNAHPLAN	bpono

4.3 Maintenance for Web

It is a requisite to change the presentation or functionality of Web site, so to keep records of all html/jsp files of APLAN is necessary for a Webmaster.

The jsp files of APLAN are located in the directory of "C:/jakarta-tomcat-4.1.24/wabapps/APLAN/jsp." Since the most of jsp files are arranged upon APLAN users' categories, if there were any changes for a function, the corresponding files will be found easily.

When updating the jsp files, the related java logical files may be changed too.

Moreover, if there were any functions added for a user, this users' main menu jsp file also need be updated; If there is a user's category is added, the homepage's jsp file needs be changed too. These files are:

/jsp/com/aplan/homepage.jsp

/jsp/com/aplan/BranchManager/branchManagerMainMenu.jsp

/jsp/com/aplan/ProductionManager/productionManagerMenu.jsp

/jsp/com/aplan/ProductionStaff/productionStaffMenu.jsp

/jsp/com/aplan/Sales/salesMenu.jsp

4.4 Maintenance for Java Source

From Figure 4, we can find that java source code is located in the directory of "C:/APLAN/java," and the class files is located in "c:/Jakarta-tomcat-4.1.24/webapps/APLAN/WEB-INF/classes." Similar as the structure of JSP, the files of java source code are also arranged based on the users' categories.

4.5 Summary

Based on the project's file structure, an administrator can easily find the way to maintain the project. Frequently saving the data in database file into the secondary disks is necessary for APLAN. If there is any attribute changes in tables, the administrator should concern about the integrity

constraints. The update for a web page may result in the revision of java logical files behind it.

CHAPTER FIVE

USERS MANUAL

5.1 Introduction

This users manual also focuses on the functionality of APLN. It is arranged in following subsections. They are:

1) Creating, updating and deleting functions maintained by a production manager; 2) Creating, updating and deleting functions maintained by sales staff; 3) Creating, updating and deleting functions maintained by branch managers; 4) Creating, updating and deleting functions maintained by production staff; 5) View pages. Although the screen shots doesn't presented in this chapter, the users can refer to them in APPENDIX B.

5.2 Functions for Production Manager

A production manager should maintain two types of tables: order confirmation and plan in the headquarters. In following subsections, they will be introduced in detailed.

Maintain Order Confirmation

Create Order Confirmation. When a production manager clicks the item to confirm a new order in production manager's main menu, the production manager can see a page has automatically generated a new order number (See Figure 18). When clicking the continue button, a page asks the user

to enter the order confirmation information. The information includes order number, order name, buyer's ID, buyer's order number, unit production time, description of order, order's destination, total quantity, quantity unit, four times shipment quantities and date. The item marked with asterisks cannot be left as empty (See Figure 19).

Delete Order Confirmation. Once clicking the item to delete an order's confirmation in the production manager's main menu, the system will allow the user to access a page to select the deleted items (See Figure 20). After checking the items and clicking the delete button, the deletion will be successful.

Update Order Confirmation. When the production manager clicks the item to change an order's confirmation in his/her main menu, the user will see a page that allows the user to enter this order's number (See Figure 21). If the order number is correct, the system will go to a page to let the production manager to update the order's confirmation items (See Figure 22). Except for the order number, all items in the page can be changed. The items marked with asterisks cannot be blank.

Maintain the Headquarters' Plan

Evaluate Branch's Productivity. Before assigning an order to one branch, the production manager can evaluate the

productivity for each branch in the group (See Figure 23). The evaluation will be shown in Figure 24. When clicking the "make plan" button, the page that allows the production manager to assign one branch and one sales staff will be accessed (See Figure 25).

Delete the HD Plan. When clicking the item to delete the headquarters' plan in the production manager's main menu, the system will go to a page to ask the user to select the deleted items. See Figure 26. Once clicking the delete button, the system will show the success deletion message.

Update the HD Plan. Similar to previous updating functions, once the production manager clicks the item to update plan for the headquarters, a page that allows the user enter the order number will be provided. If the number is correct, the system will provide a page that allows the production manager to update plans of the headquarters. See Figure 27.

5.3 Functions for Sales Staff

For a sales staffer, he/she needs to maintain table of order report. In this manual, these functions to maintain the table of order report will be introduced as following:

Create Order Report

In the main sales menu, when a sales staffer clicks the item to entry the clothing order report, a page that allows the user to enter the information for the order report will be given. The information includes report's order number, pattern name, pattern number, size, four expected shipment quantities. See Figure 28.

Delete Order Report

When the production manager clicks the item to delete clothing report in the sales staff menu, the system will let the user to select the delete item's order, if the order number is correct, all order report items for the order will be presented (See Figure 29). After checking the selected items and clicking the delete button, a success page will be shown.

Update Order Report

When choosing the item to change order's report in the production manager's main menu, the system will show a page to ask the sales staff to enter the order number, pattern number and size for an item that he/she wants to update (See Figure 30). If the entered information is correct, a page that presents the items' original order report's information will be accessed (See Figure 31). Except for the order number, pattern number and size, all items can be updated.

5.4 Functions for Branch Managers

A branch manager should maintain the table of branch production plan. Before making the plan, the branch can evaluate the lines' productivities for a specific order. The functions maintained by the branch manager will be introduced in follwing subsections:

Evaluate Lines' Productivity

When a branch manager clicks the item to evaluate the lines' productivity, the branch manager will see a page to ask the user to enter the number of order that will be evaluated (See Figure 32). If the order number is valid, a page that asks the branch manager to enter the coefficient for each line will be provided (See Figure 33). After entering the coefficient and clicking the "evaluate productivity" button, the evaluation page will be presented (See Figure 34). If the manager wants continue to make a plan for the order, he/she can click the button of "making a branch plan."

Maintain Branch's Plan

Create Branch's Plan. The branch manager can access the page to create branch's plan by two ways: either by clicking the item to make a plan in branch managers' main menu or from the button of "making a plan" in the page to present the productivity evaluation result. A page to make a

branch production plan is shown in figure 35. The items with asterisks are essential.

Delete Branch's Plan. After clicking the item to delete a plan in branch managers' main menu, the branch manager will see a page to ask the user to enter the order number to delete its production plan. If the order number is correct, a page allows the user to select the deleted items will be presented (See Figure 36). After the manager chooses the deleted items and click delete button, a page to show the deletion success is accessed.

Update Branch's Plan. When a branch manager clicks the item to change production plans in branch managers' main menu, a page that asks the manager to enter the update item's information will be shown. The information includes item's order number, branch name and line number (See Figure 37). After entering these information and clicking the submit button, a page that presents the original branch plan item will be presented (See Figure 38). In this page, the information regarding to quantity and date can be updated.

5.5 Functions for Production Staff

A table maintained by production staff is the production report. The table will be introduced in following subsections:

Create Production Report

When a production staff clicks the item to create a production report, he/she will see a page to ask him/her to enter the information for a production report. All items with red asterisks are essential (See Figure 39).

Delete Production Report

When a production staff clicks the item to delete a production report, a page that allows the staff to enter the deletion items' order number will be presented (See Figure 40). If the user selects the deleted items and clicks the delete button, the deletion will be successful.

Update Production Report

If a production staff clicks the item to update production report in production staff's main menu, a page that permits the user to enter the updated item's information will be presented. The information includes order number, branch name, line number and date (See Figure 41). If all information is correct, the page that shows the original production report's information will be seen (See Figure 42). In the page, the production staff can update the remark, input and output quantity.

5.6 View Pages

There are twelve tables can be viewed by the authorized users in APLAN. The twelve tables are contact list, order confirmation, order report, the production plan in the headquarters, branch's production plan and order status. These tables will be presented as:

View Contact List

There are three kinds of contact list tables in APLAN. They are contact list for buyers and staff or managers in the group. To see the contact list, the user just clicks the item to view contact list in the main menu, a page for selecting contact list type would be presented (See Figure 43). If the user chooses the type for the staff and managers in the group, he/she should continually select to view all users, view all sales staff or more (See Figure 44). The sample of present result is provided in Figure 45.

View Orders' Confirmation

A production manager, sales staff, branch managers and production staff have the authorization to view orders' confirmation. When they click the item to view orders' confirmation in their own main menu, a page to let them to choose the view type will be presented (See Figure 46). They can select to view all orders, by buyers or by destinations. If they choose the next two types, they should continually

to select the view type (See Figure 47). Then the result will be presented (See Figure 48).

View Order Report

Order Report can be viewed by a production manager, sales staff, branch managers and production staff. They can see the page by clicking the item to view order report in their own main menu. Then a page will allow them to select to view all orders' report or only one order's report (See Figure 49). If the user chooses the view only one order's report, he/she will be asked to enter the order's number (See Figure 50). If the order number were valid, the viewed page would be presented (See Figure 51).

View Plans in the Headquarters

The headquarters' plans are maintained by the production manager. The production manager, sales staff and branch managers can view the plans. When a user clicks the item to view the headquarters' production plan in their own main menu, the user will be entered into a page to allow him/her to choose the view type: all orders, by destinations, by buyers and buy branches (See Figure 52). If the user chooses the type by destinations, he/she would be asked to choose the destination's location (See Figure 53). If the type by buyers was chosen, the user needs to choose the buyer's name (See Figure 54). If the user chooses the

type by branches, then he/she should repeatedly to choose the branch's name (See Figure 55). The headquarters' plans would be presented like the page in Figure 56. The presented information includes order number, order name, buyer name, buyer's order number, total quantity, quantity unit, handling branch name, handling sales name.

View the Branch's Plans

Branch plans are maintained by the branch managers. The plans can be viewed by the production manager, branch managers and sales staff. When a user clicks the item to view the branches' plans in their own main menu, a page that asks the user to select the sort type will be presented (See Figure 57). Based on the select sort type, the result will be presented (See Figure 58). The page includes order number, branch name, line number four times shipment quantity, four times production begin date and end date.

View the Production Report

Production reports are maintained by production staff. The reports can be viewed by the production manager, branch managers, sales staff and production staff. A user can click the item to view the production report in his/her main menu, and then a page that allows the user to select the view type will be shown (See Figure 59). If a user selects to view only one order's production report, he/she would be asked to

type the order' number (See Figure 60). If the order's number were correct, the result would be shown (See Figure 61).

View Order Status

All users' works is presented graphically in the page of order status. The order status page can be viewed by the production manager, branch managers and sales staff. When a user clicks to view orders' status in his/her main menu, a page that allows the user to select the view type will be shown (See Figure 62). If the user selects to view only one order, he/she will be asked to enter the order number (See Figure 63). The result looks like the Figure 64. Normally there are three colors to presents three status: Red color means that an item does not exist or can not be completed at all; Green color means that the item is completed or exists; Yellow color means that the item is partly completed or exists in part.

5.7 Summary

From the manual introduced in above subsections, we can find that APLAN is a system that can be maintained and used very easily. By studying the functions listed in users' main menu, the related users can quickly understand the system's

structure. The system's menu pages and all buttons in each page can guide the users to manage the process well.

CHAPTER SIX

CONCLUSIONS

6.1 Introduction

Included in Chapter six was a presentation of the conclusions gleaned as a result of completing the project. Lastly, the Chapter concludes with a summary that summarizes the product's contributions in the management for a garment manufacturing company.

6.2 Conclusions

The conclusions extracted from the project follows:

1. The web-aided database management system was developed to provide a model to manage the production for garment manufacturing companies in one system. With the assist of the system, this kind of companies will greatly improve their efficiency and accuracy of management.
2. The product allows the users to request or enter data by using browsers. The data is stored in a database system named MySQL. The data is retrieved from a server. The interface between the MySQL database and the JSP is the java logic parts connected by JDBC.
3. APLAN almost covers all the production management works for a garment manufacturing company. In order to use the

system well, this kind of companies should have a corresponding personnel system.

4. Sharing viewed tables helps the related managers or staff retrieve their needed up-to-date information easily and instantly.

5. Software design is the main part in this documentation. The functionality of users is presented in the use case diagram. E-R diagram introduces all database entities and the relationships in each other. The attributes of each entity are shown in those pages that describe the properties of entities. Class diagram outlines the java classes and their associations between two classes. In this document, in order to simplify the expression of class diagram, the operations and attributes for each class are hidden from the diagram.

6. A project test plan for APLAN is used to ensure the quality of product. This plan includes the unit test plan, integration test plan and system test plan. Units for APLAN to be tested are mainly web pages plus the classes and methods hidden behind them. The integration test for the project focuses on whether hyperlinks go to a proper page. System test plan focuses on verifying the handling of valid input data, checking error and checking the output result.

7. Based on the system's structure of directory and the project's unit test description, an administrator can easily find a way to maintain the system. Integrity constraints are the main concern when database is changed. For any functionality changes of web pages, the classes hidden behind them may be changed too. Once there is any update for the user's category, the home page of APLAN should be changed correspondingly; once there is any added or updated functionality item for a user's category, the main menu for this user's category also should be changed.

8. APLAN's user manual not only guides the users to use the product but also presents the functions in detail by showing the screenshots. The user manual is arranged upon the user's category and the functionalities. An administrator also can refer to the manual for the purpose of maintenance.

6.3 Summary

Chapter six reviewed the conclusions extracted from the project. From software design to user manual, the conclusions summarize the main topics for each chapter. By reading the conclusions, the users can easily grasp the structure of the product. Form these conclusion, we also can find how the product helps a garment manufacturing company

to handle its management works. We believe that APLAN is such a product that greatly improves the production management in garment companies, so it also frees the staff and managers in these companies from the heavy load in production managements.

APPENDIX A

DATABASE E-R DIAGRAM AND ENTITIES

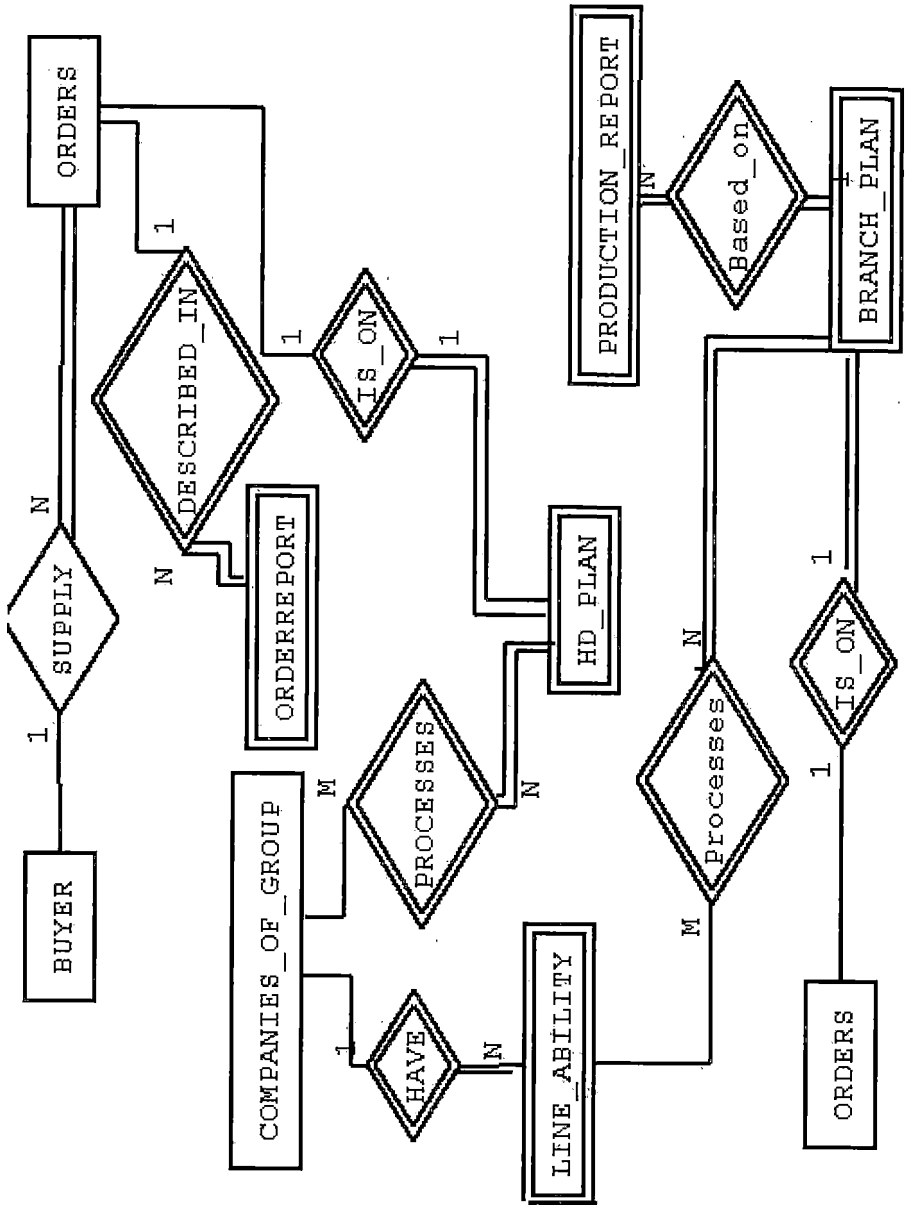


Figure 5. E-R Diagram

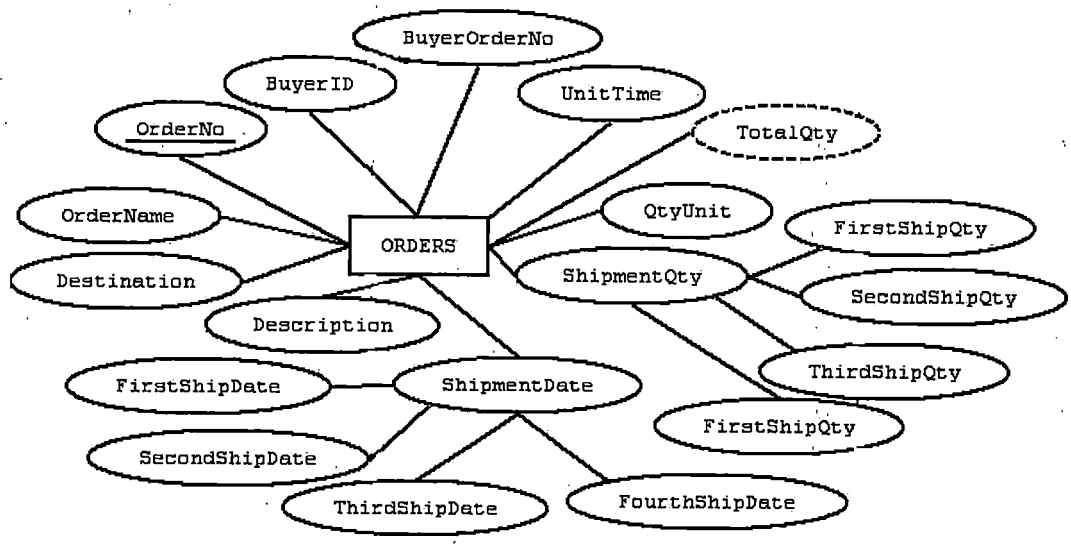


Figure 6. ORDER Entity

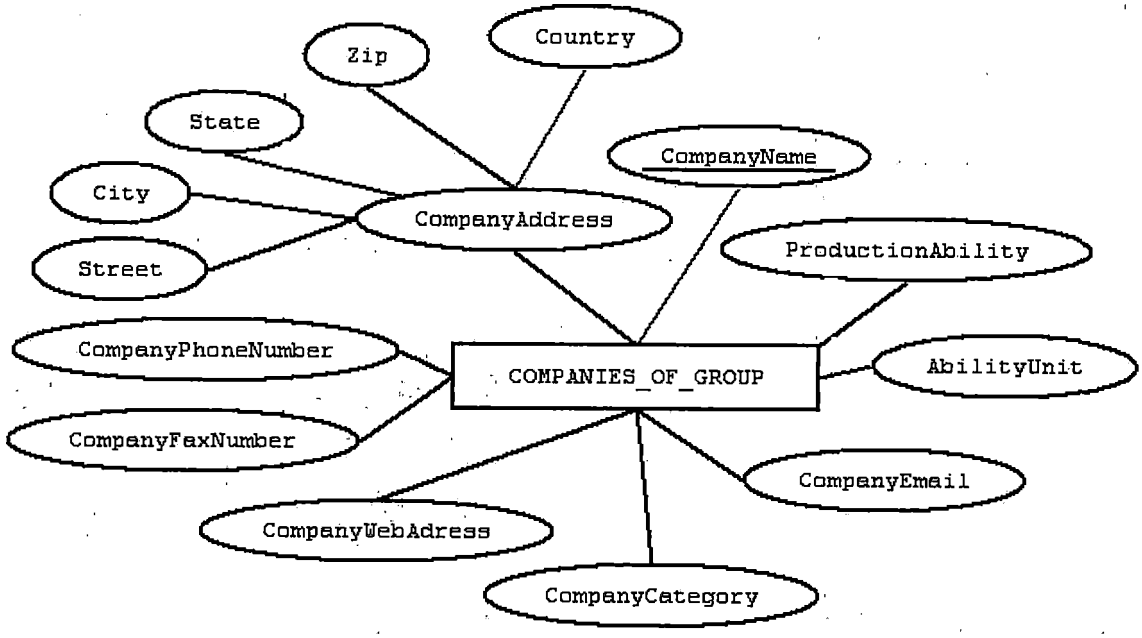


Figure 7. COMPANIES_Of_GROUP Entity

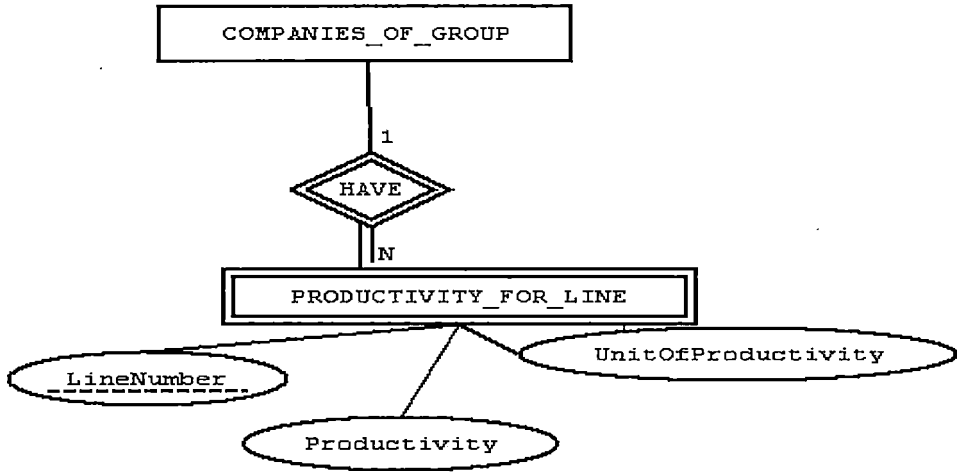


Figure 8. PRODUCTIVITY_for_LINE Entity

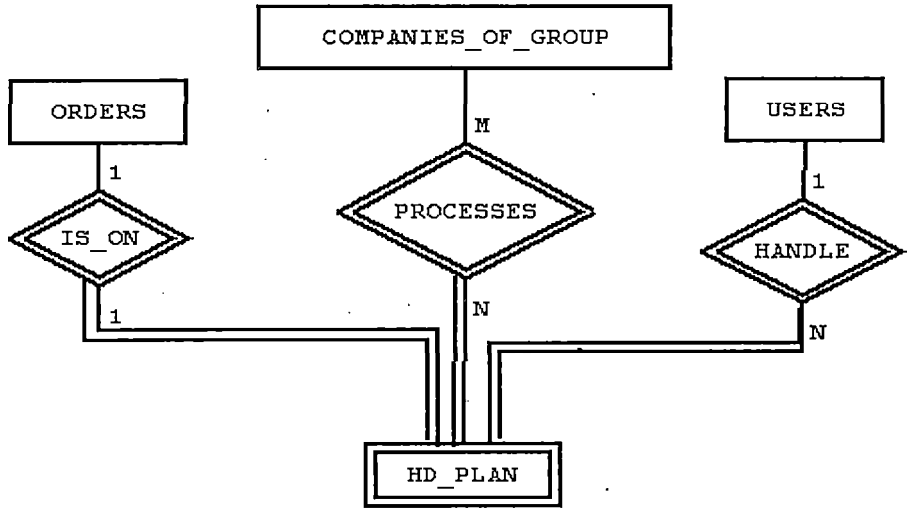


Figure 9. HD_PLAN Entity

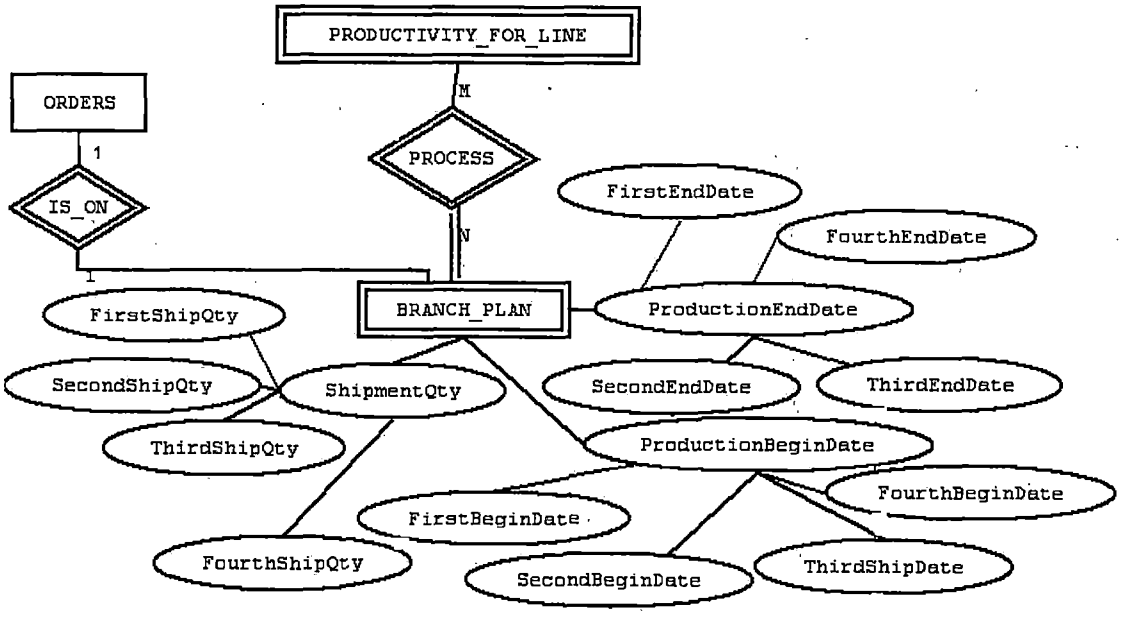


Figure 10. BRANCH_PLAN Entity

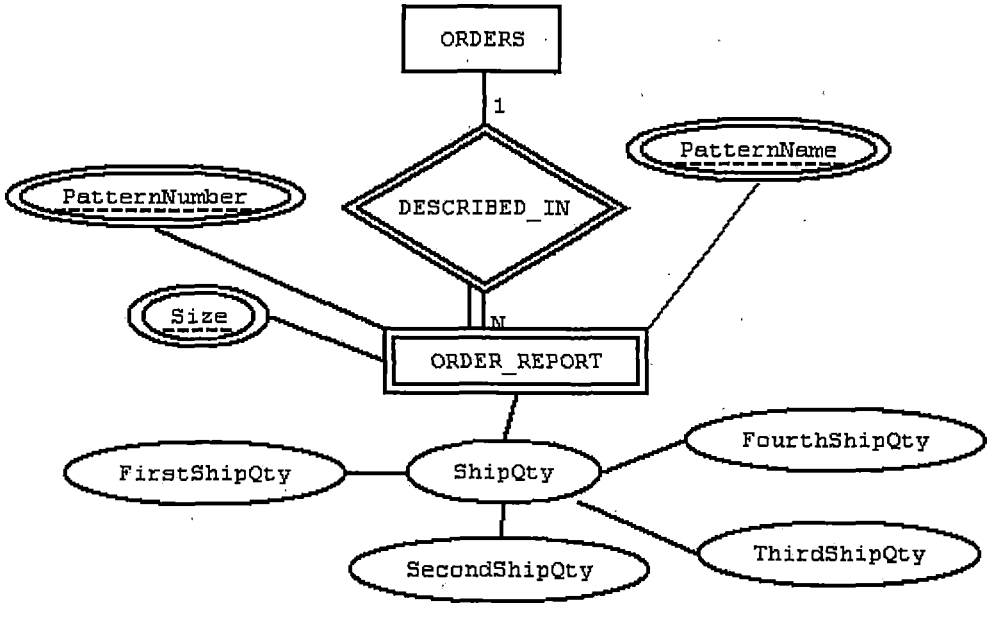


Figure 11. ORDER_REPORT Entity

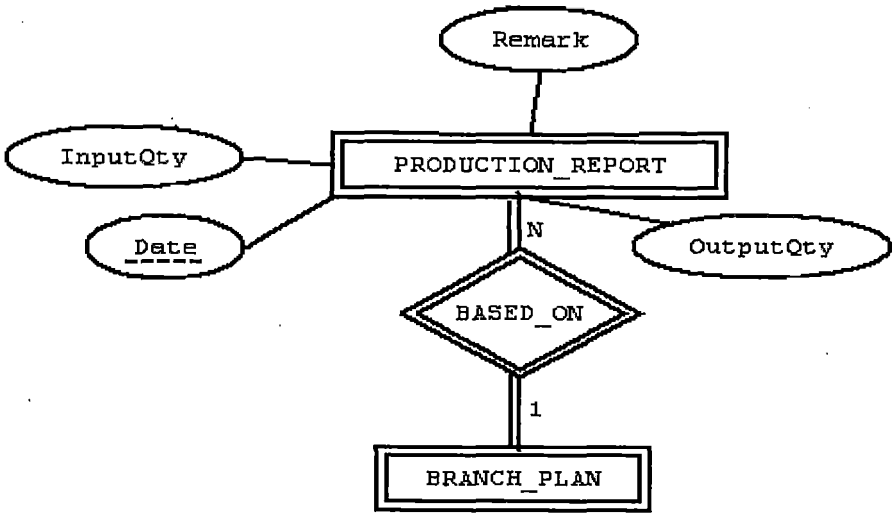


Figure 12. PRODUCTION_REPORT Entity

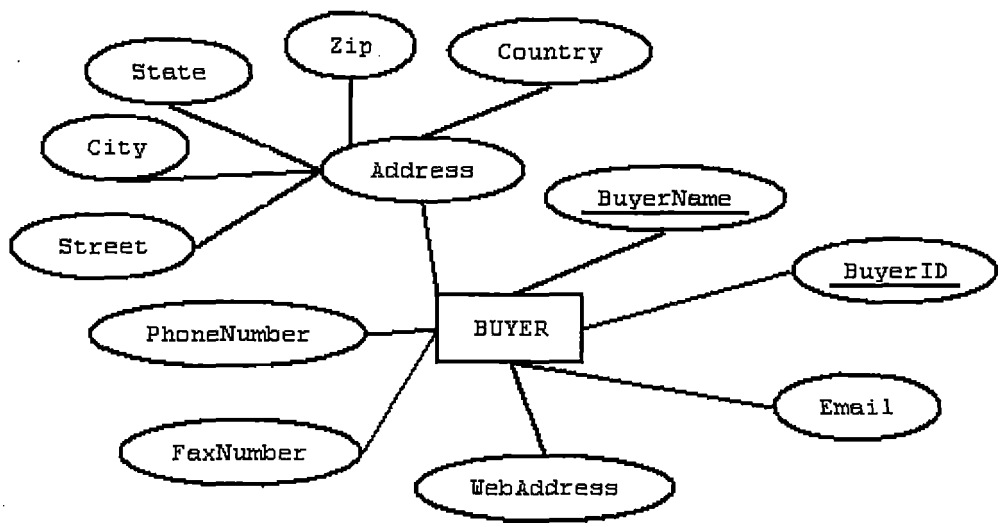


Figure 13. BUYER Entity

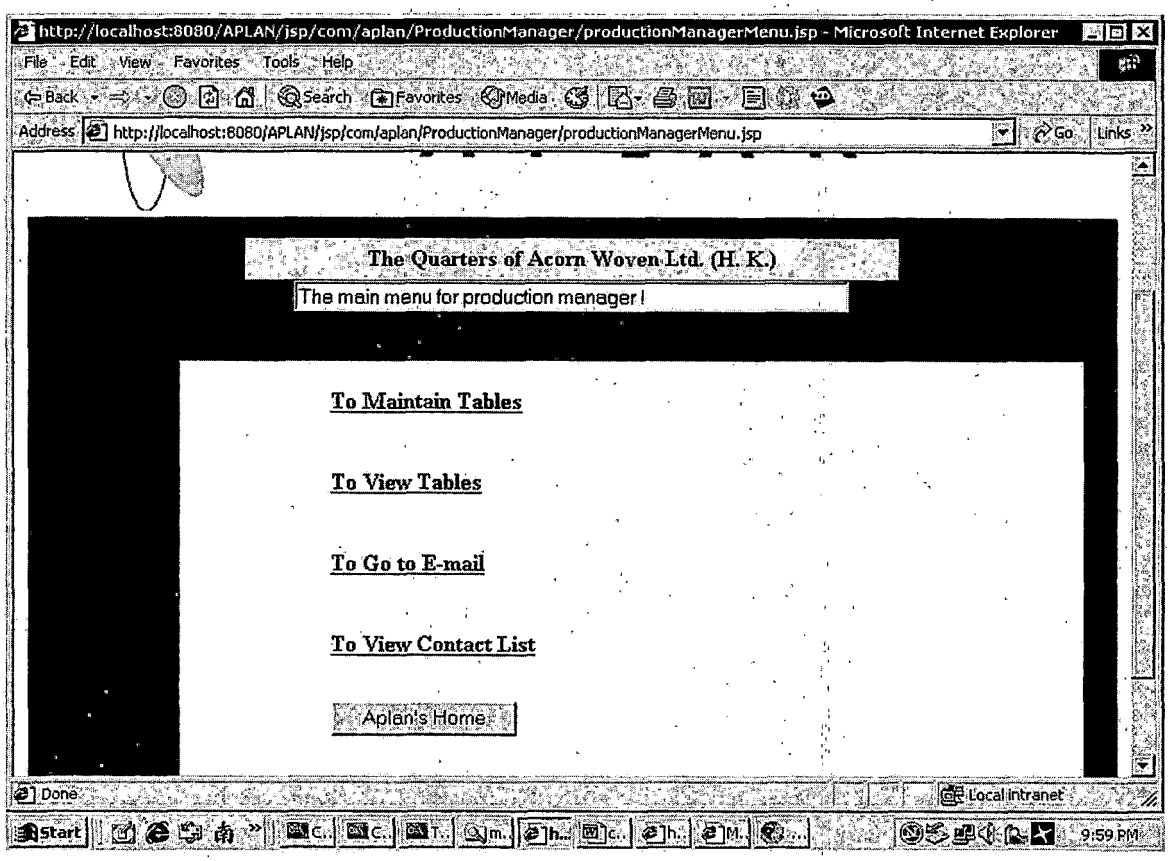


Figure 14. Production Manager's Main Menu

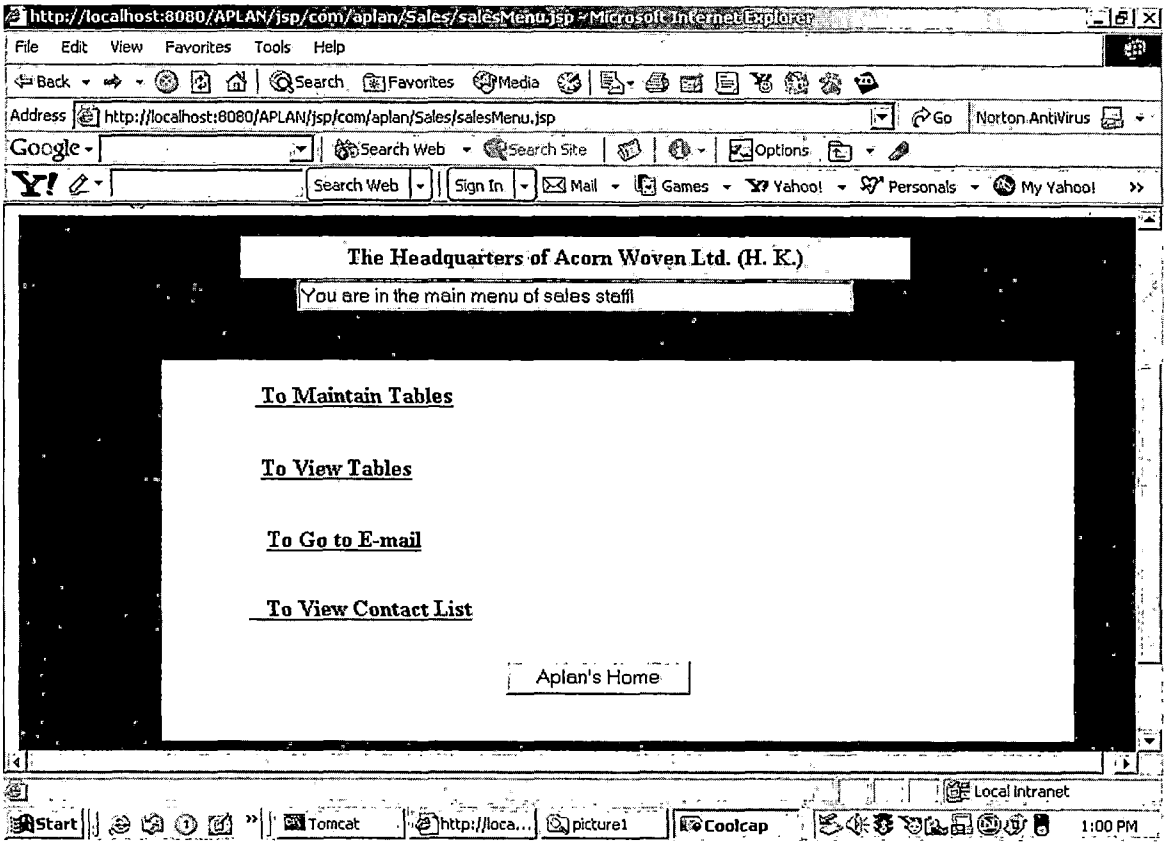


Figure 15. Sales Staff's Main Menu

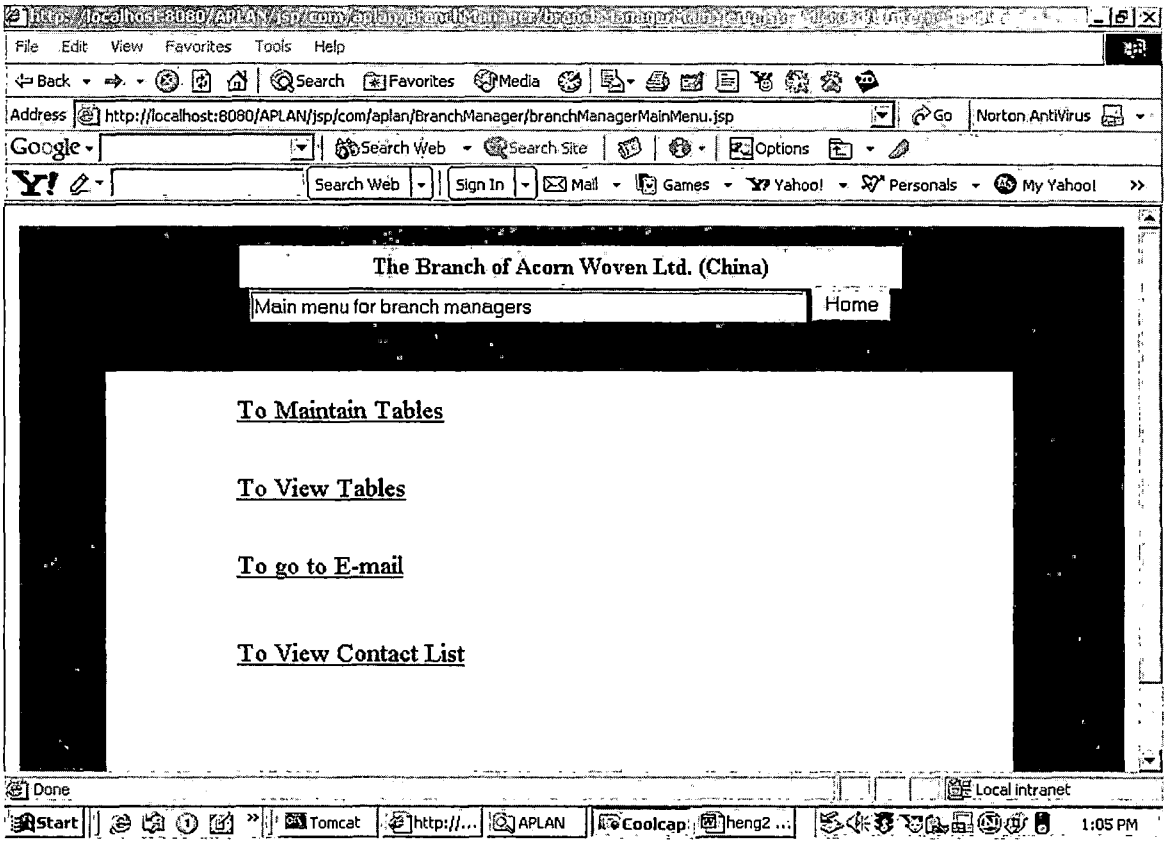


Figure 16. Branch Manager's Main Menu

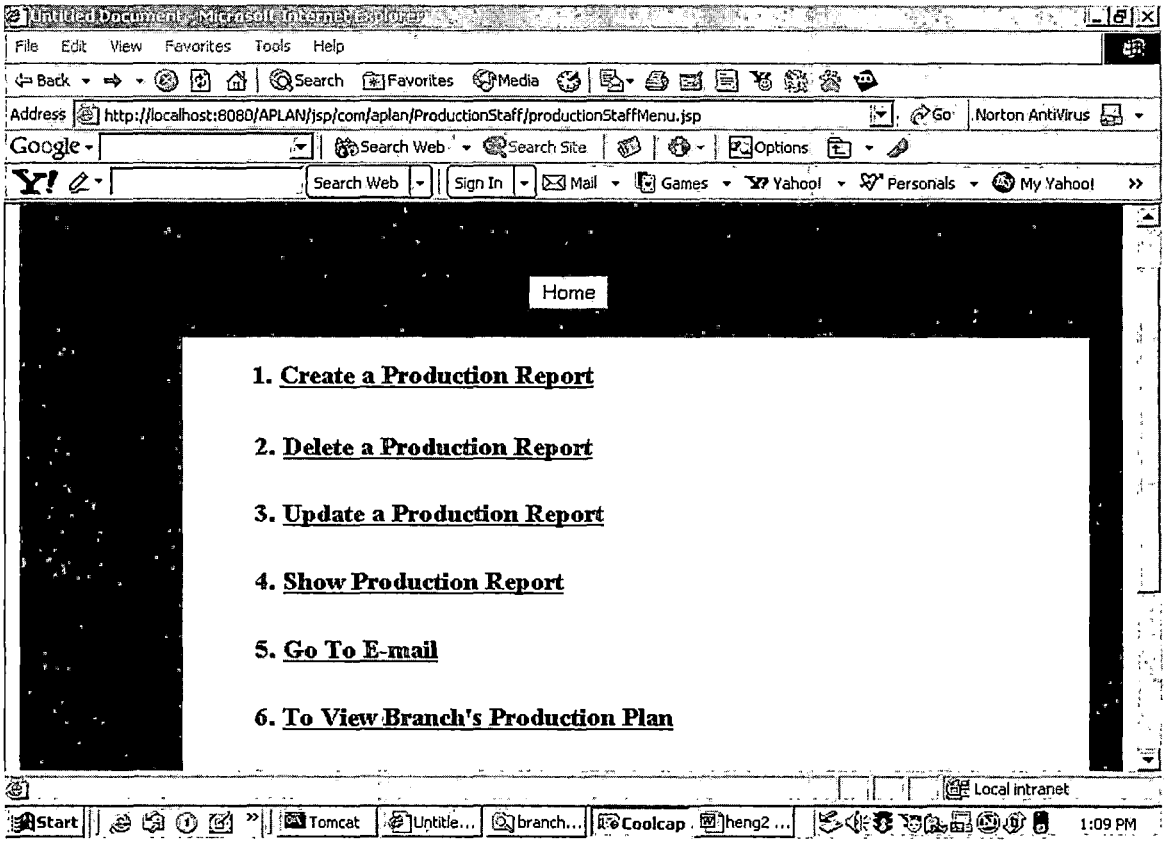


Figure 17. Production Staff's Main Menu

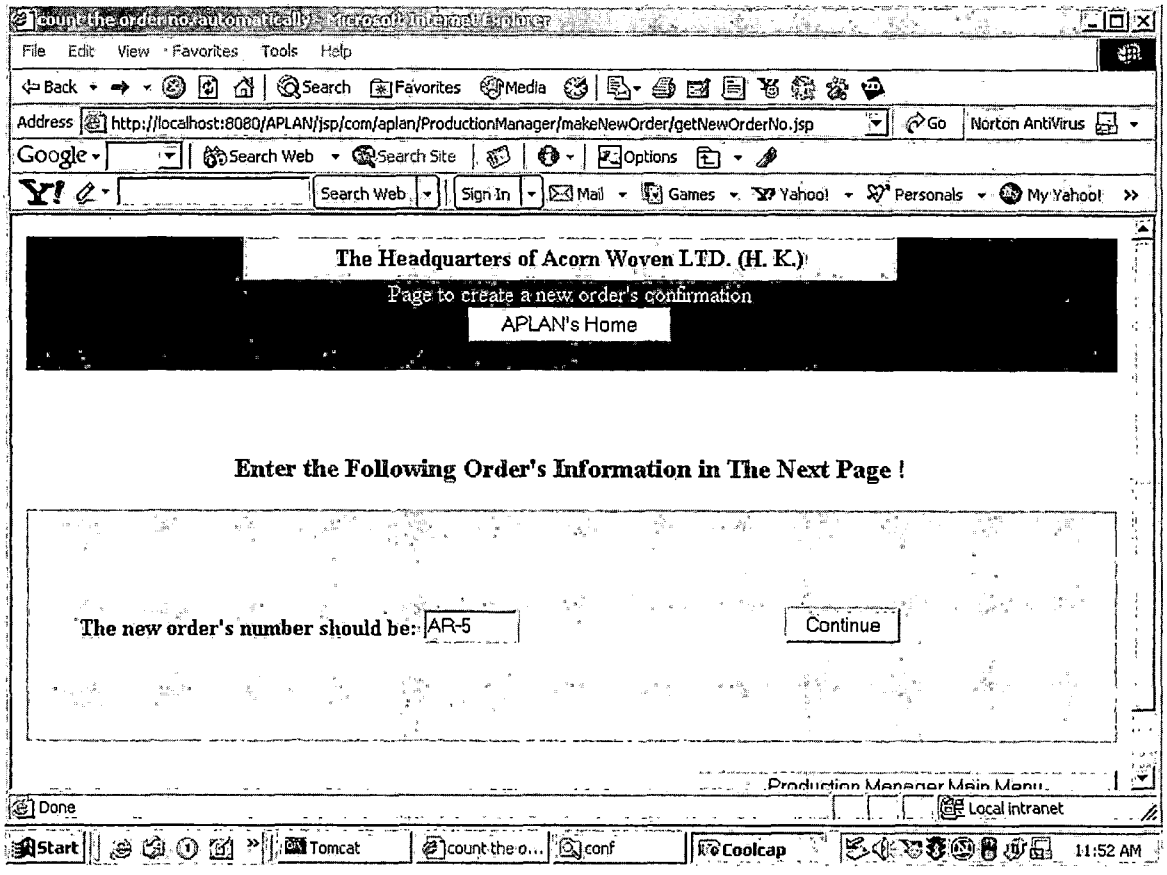


Figure 18. Page of Showing Order's Number

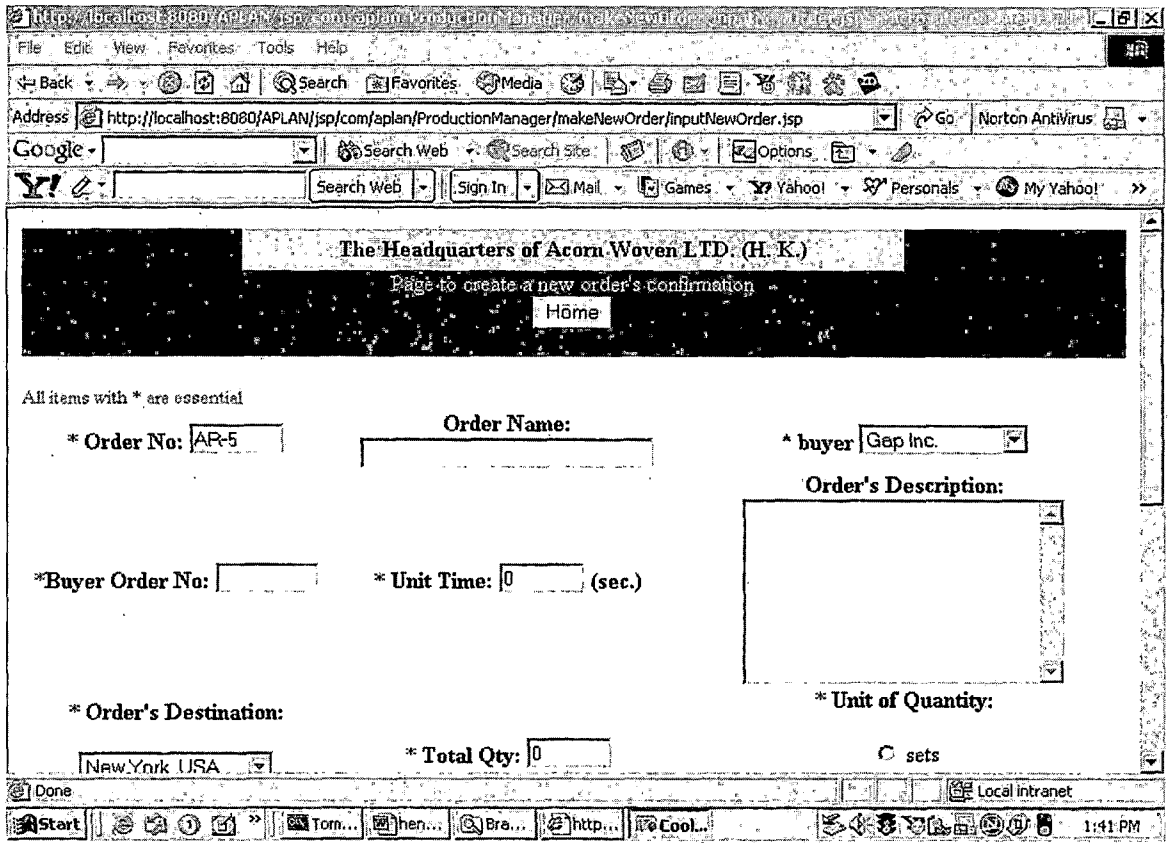


Figure 19. Enter Order Confirmation's Information

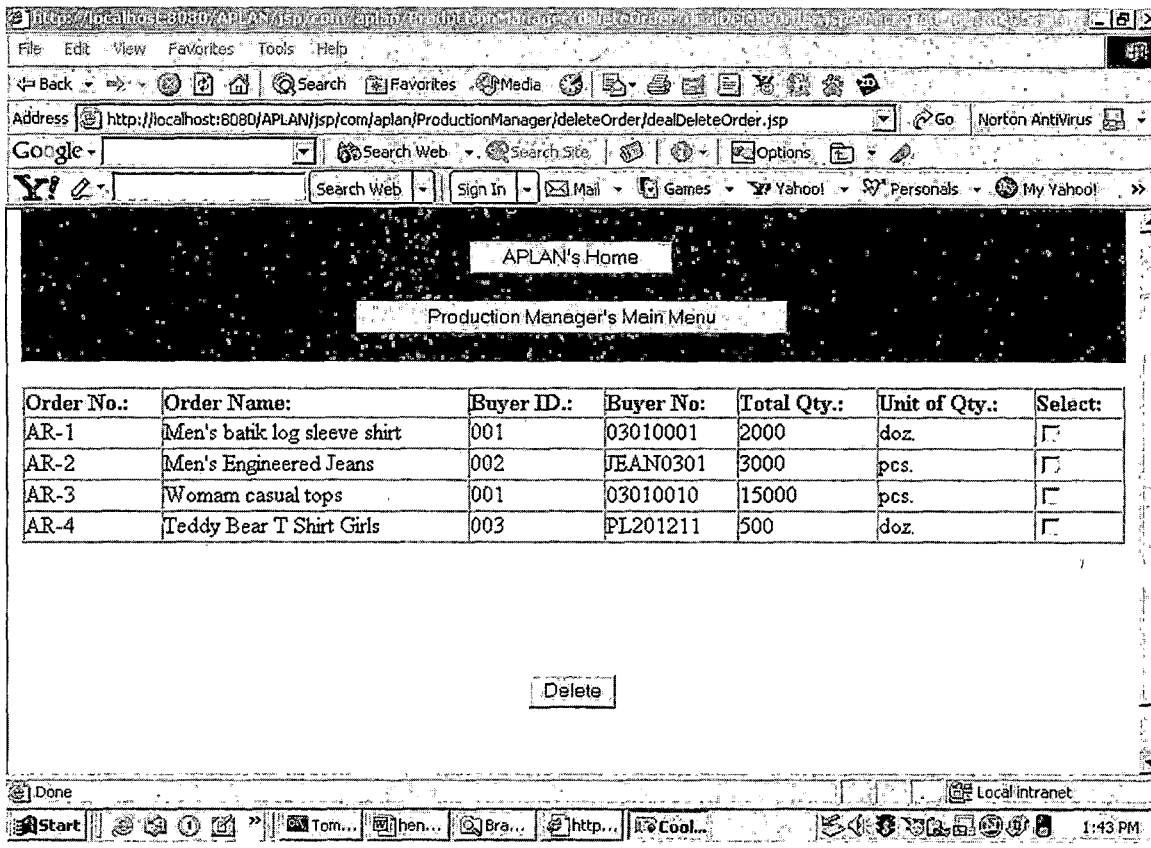


Figure 20. Page to Delete Order's Confirmation

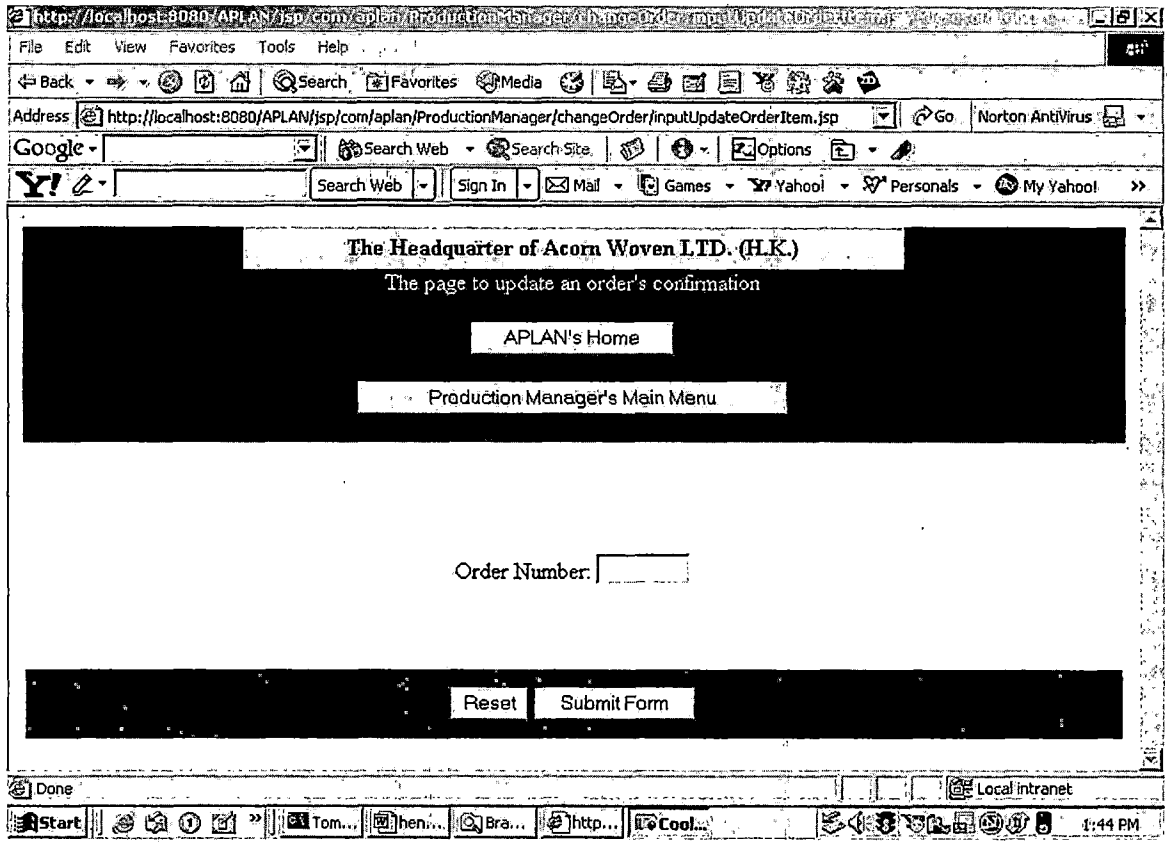


Figure 21. Enter Order to Update Order

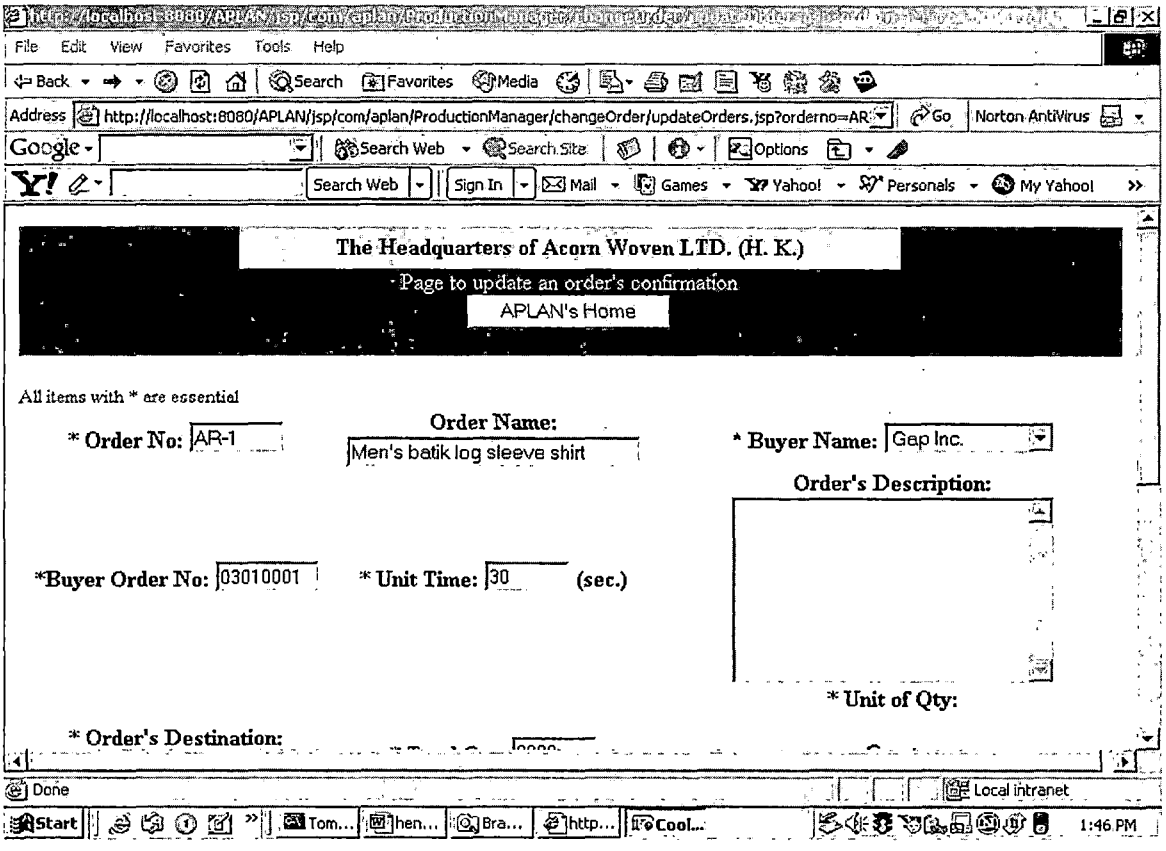


Figure 22. Page to Update Order Confirmation

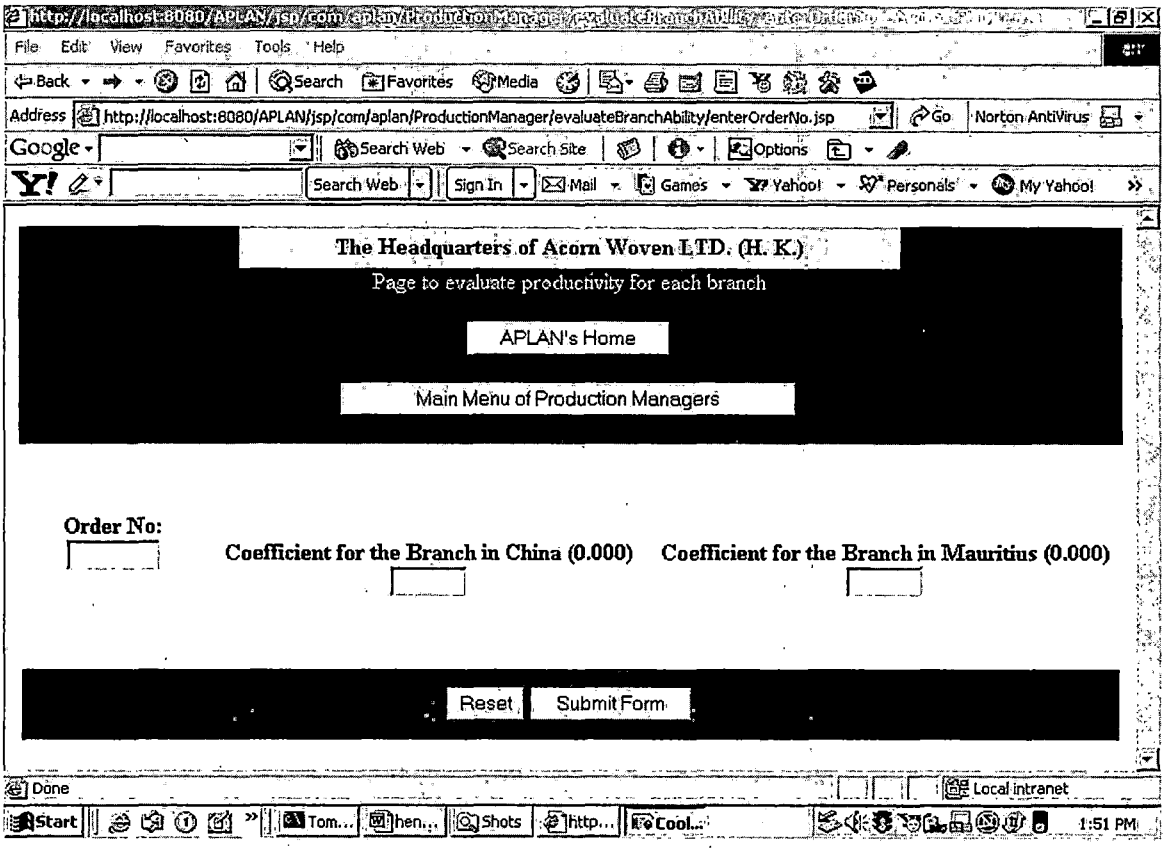


Figure 23. Evaluate Productivity of Branches

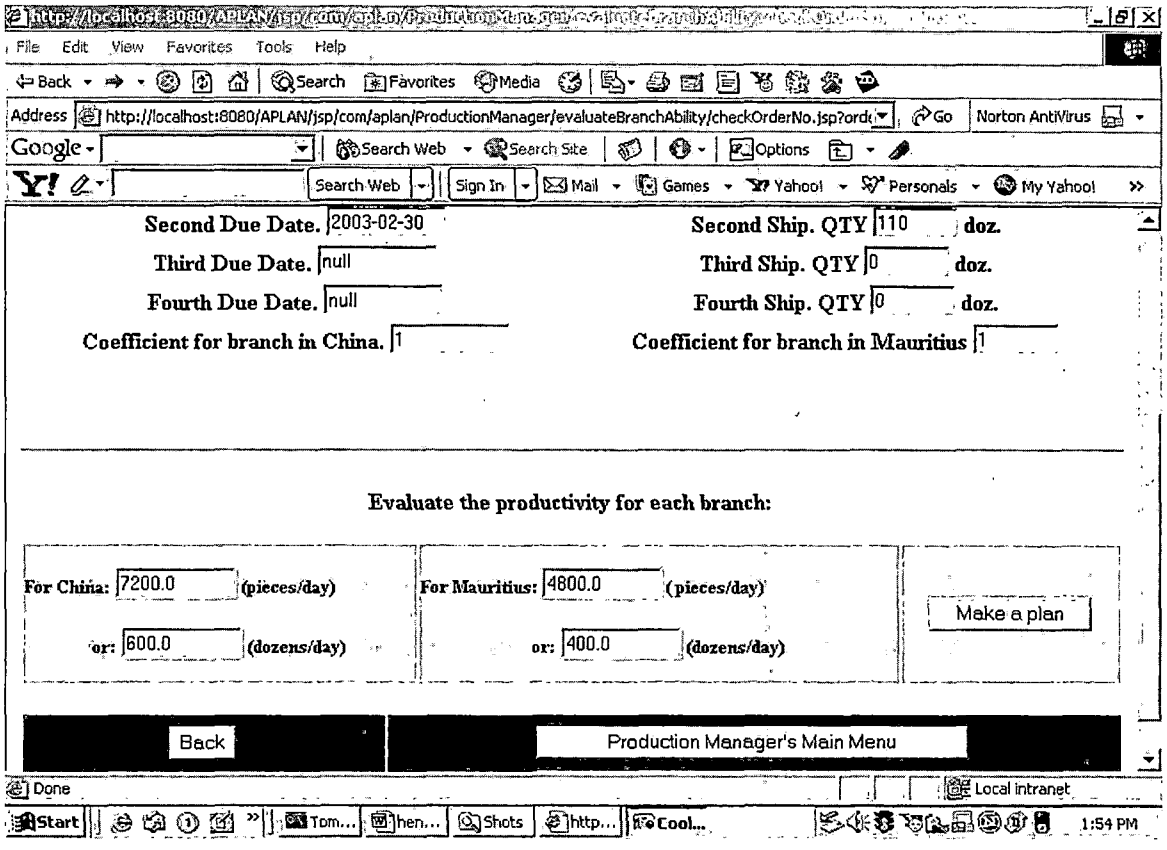


Figure 24. Result of Branches' Productivity

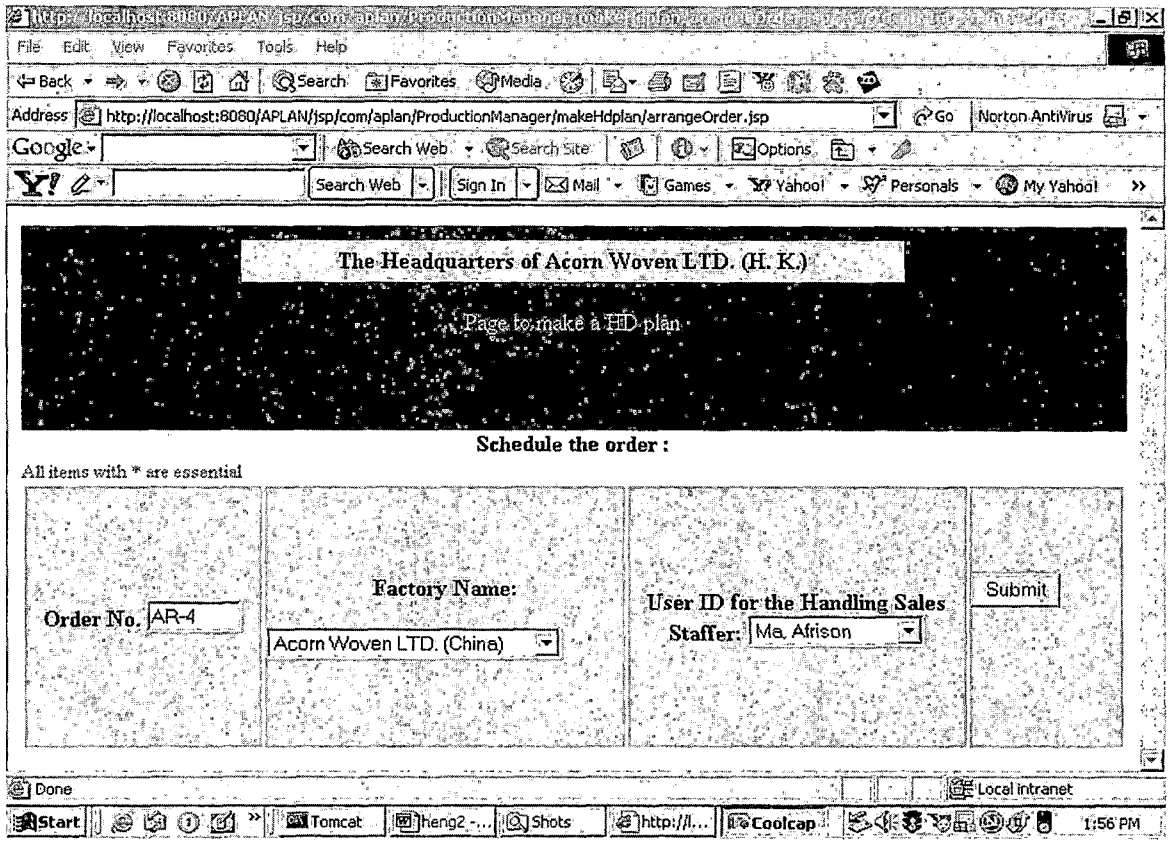


Figure 25. Make the Headquarters Plan

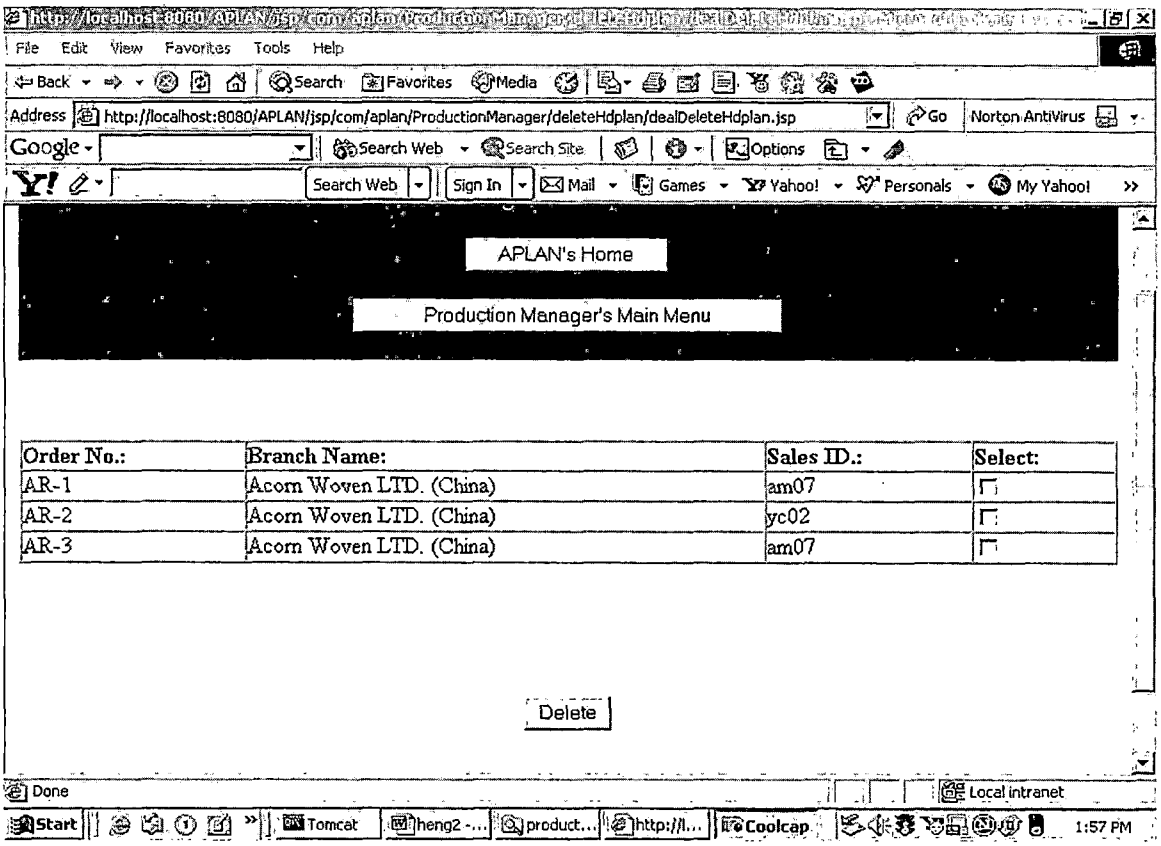


Figure 26. Delete the Headquarters Plan

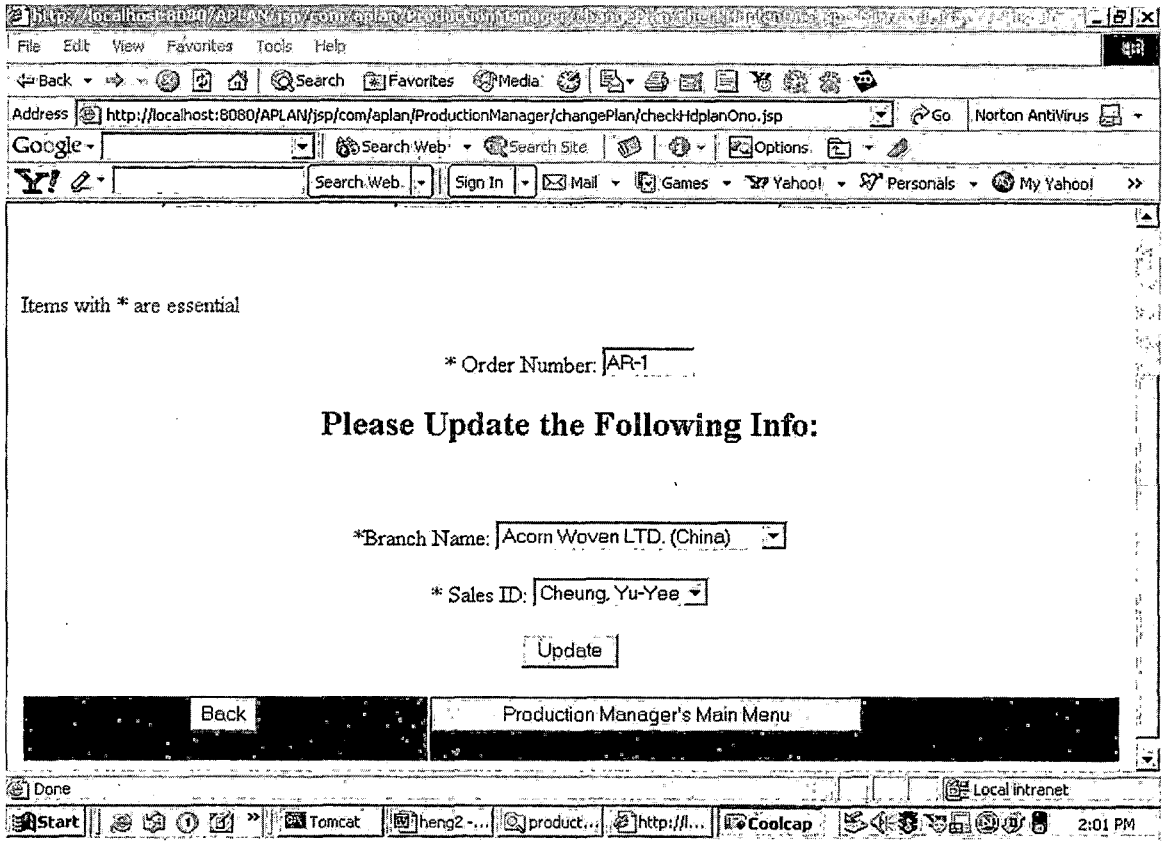


Figure 27. Update the Headquarters Plan

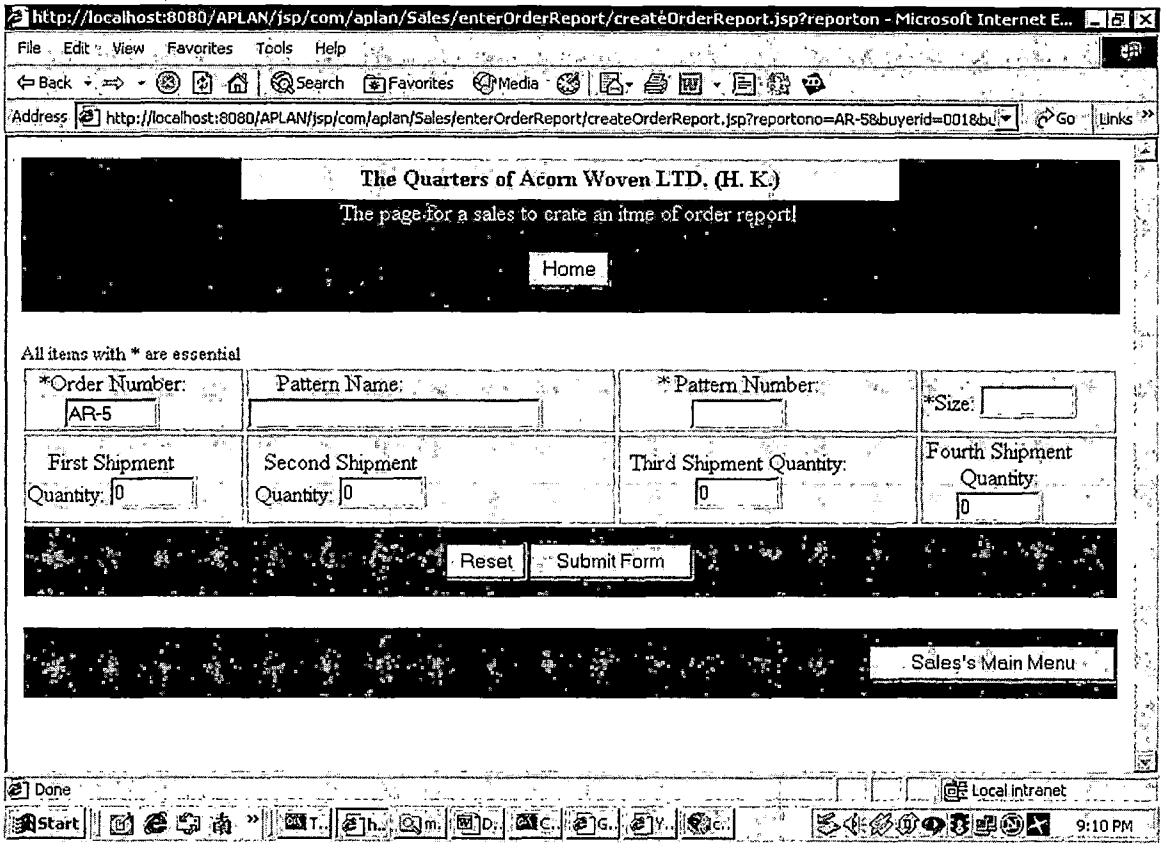


Figure 28. Page to Enter Order Report

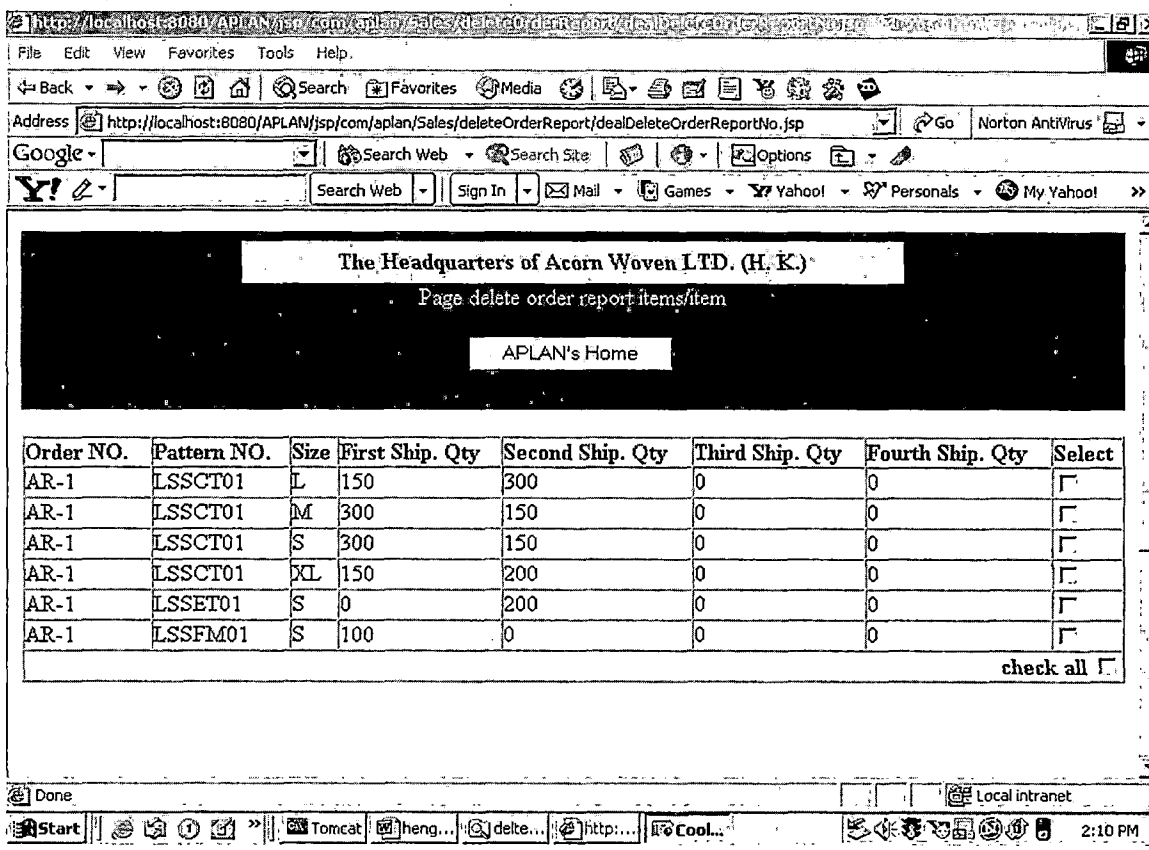


Figure 29. Page to Delete Order Report

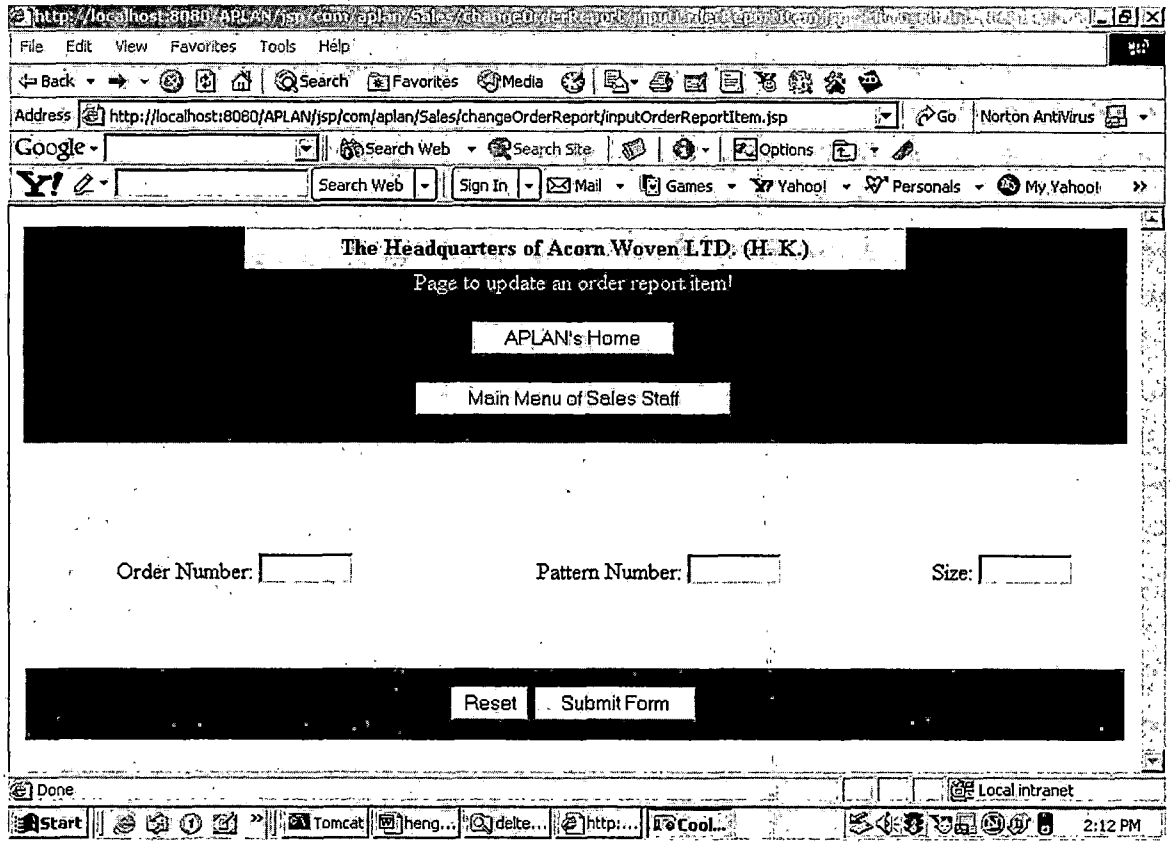


Figure 30. Enter Item to Update Order Report

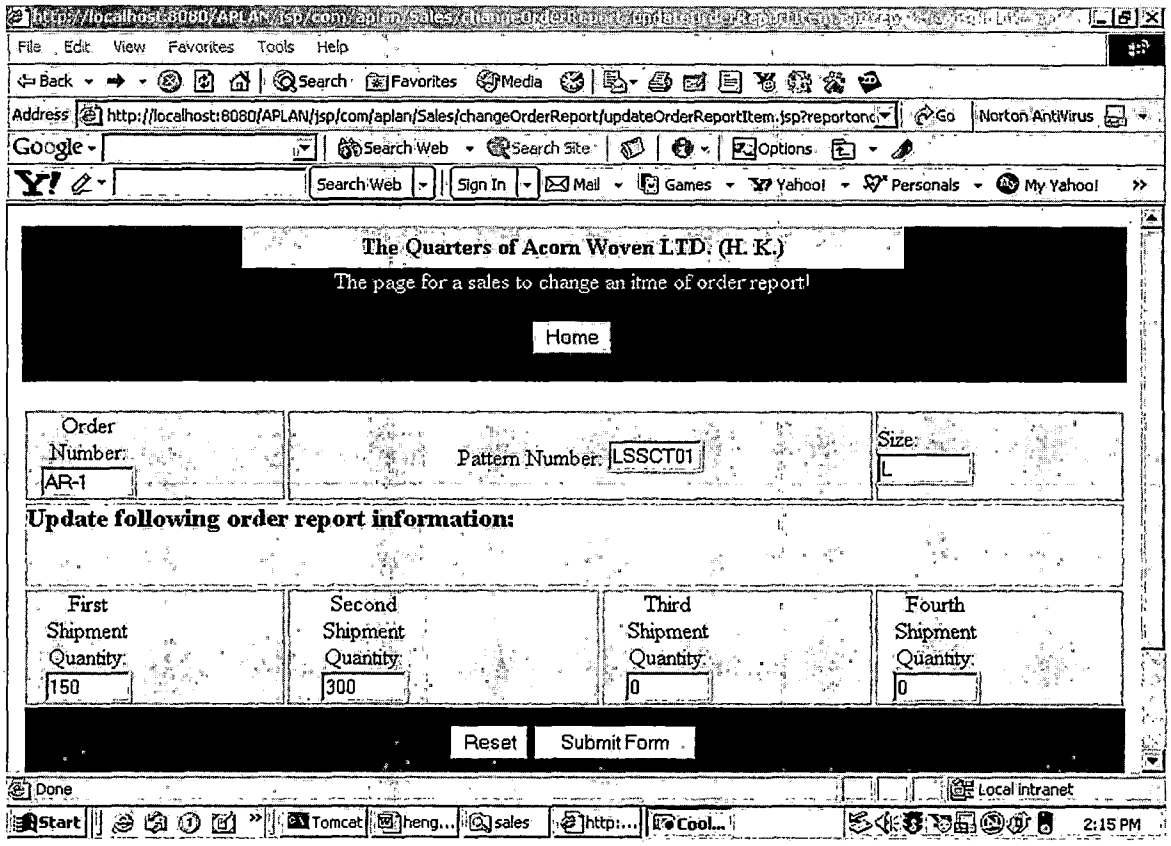


Figure 31. Page to Update Order Report

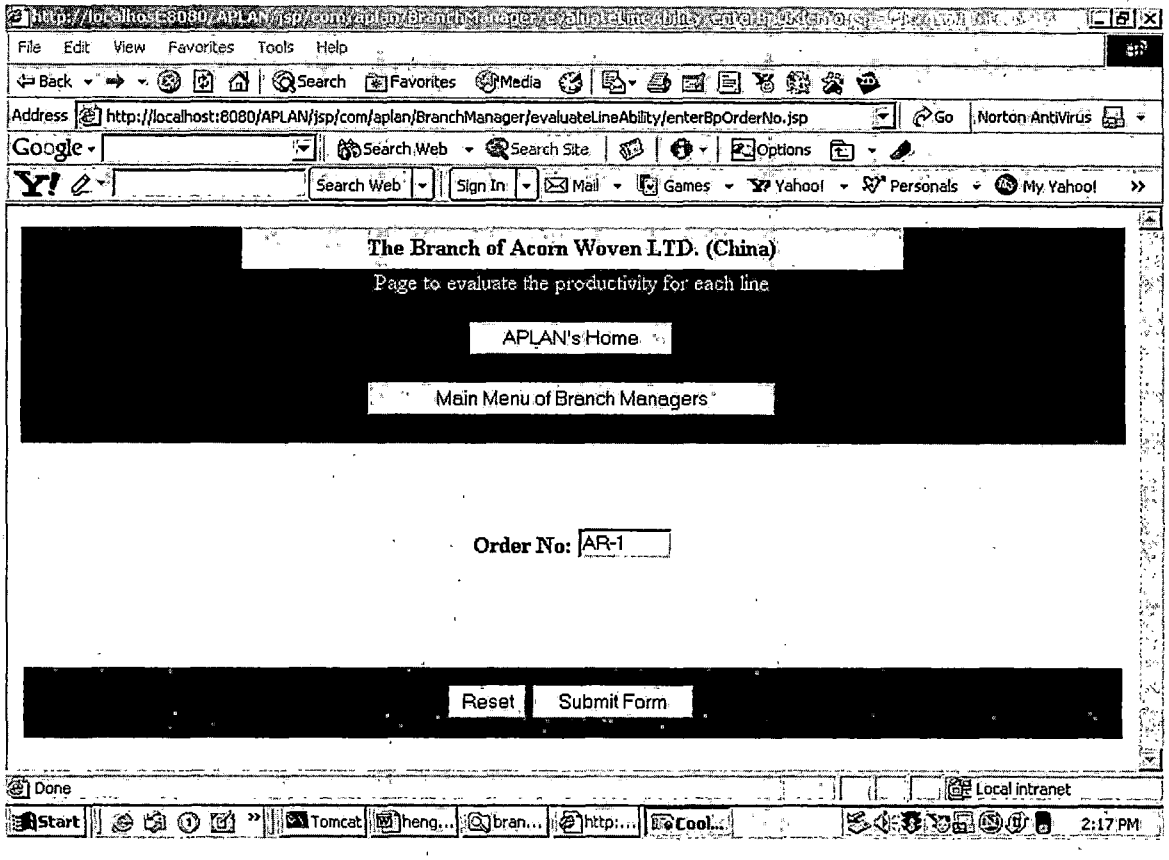


Figure 32. Evaluate Productivity of Each Line

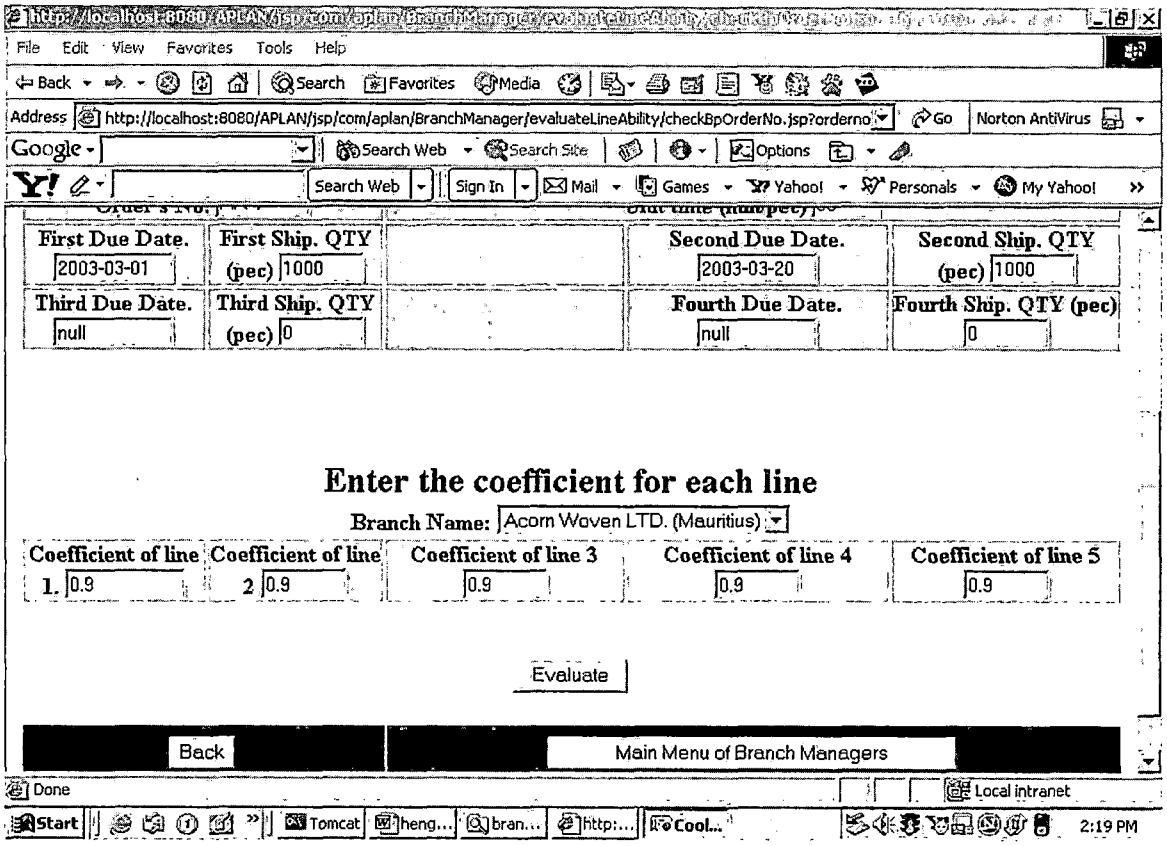


Figure 33. Enter Coefficient for Each Line

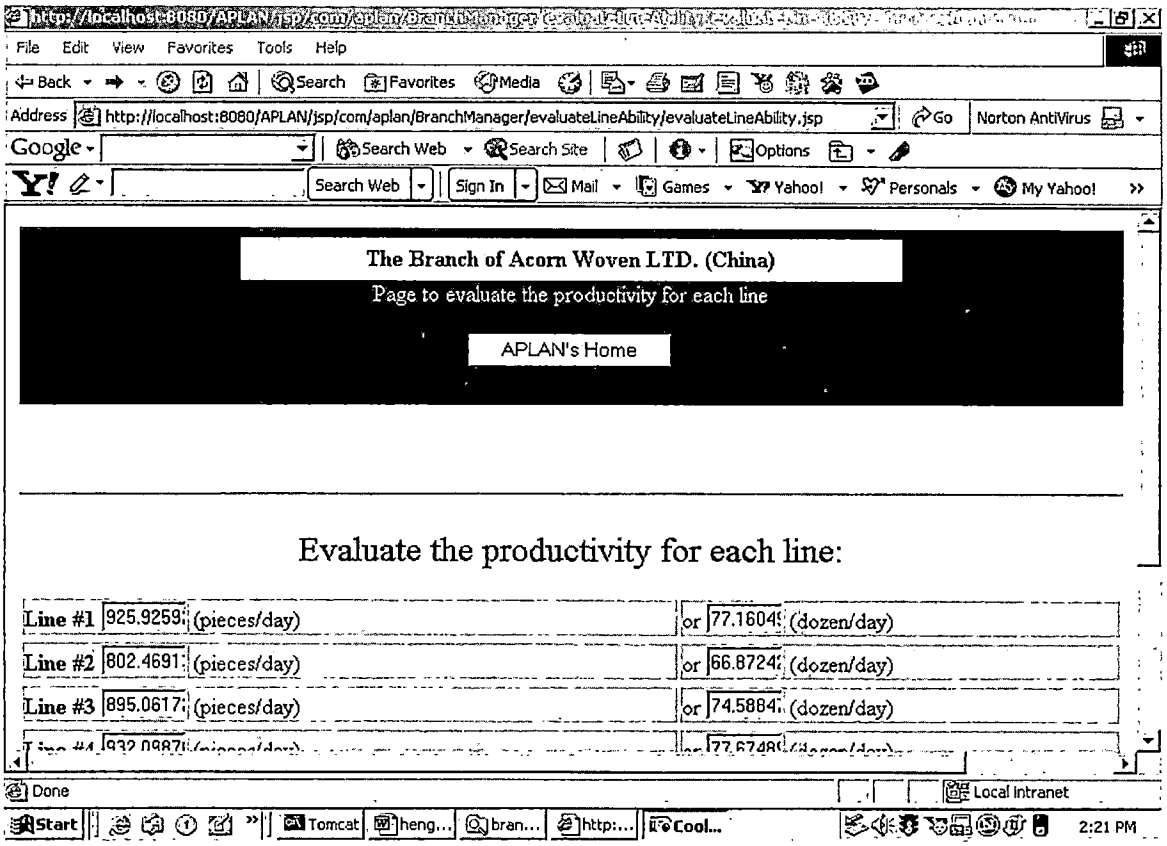


Figure 34. Result for Line's Productivity

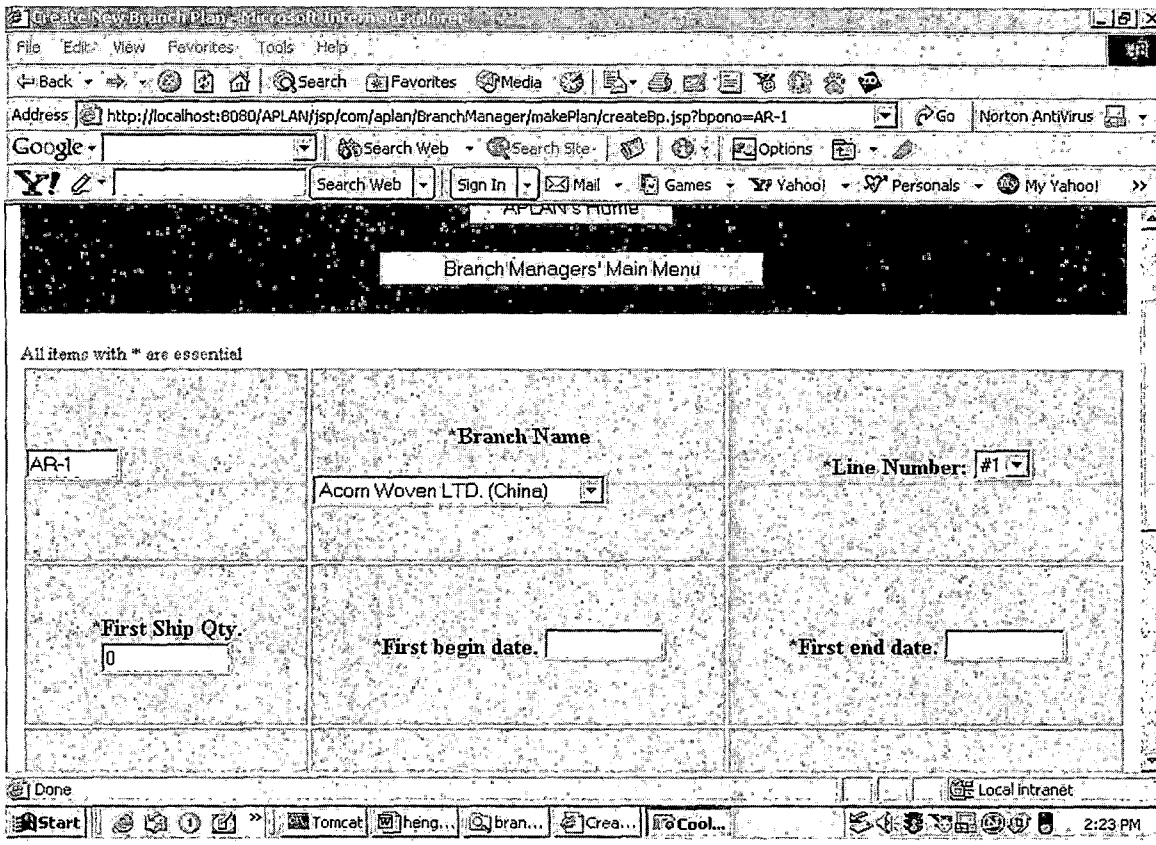


Figure 35. Make a Branch's Production Plan

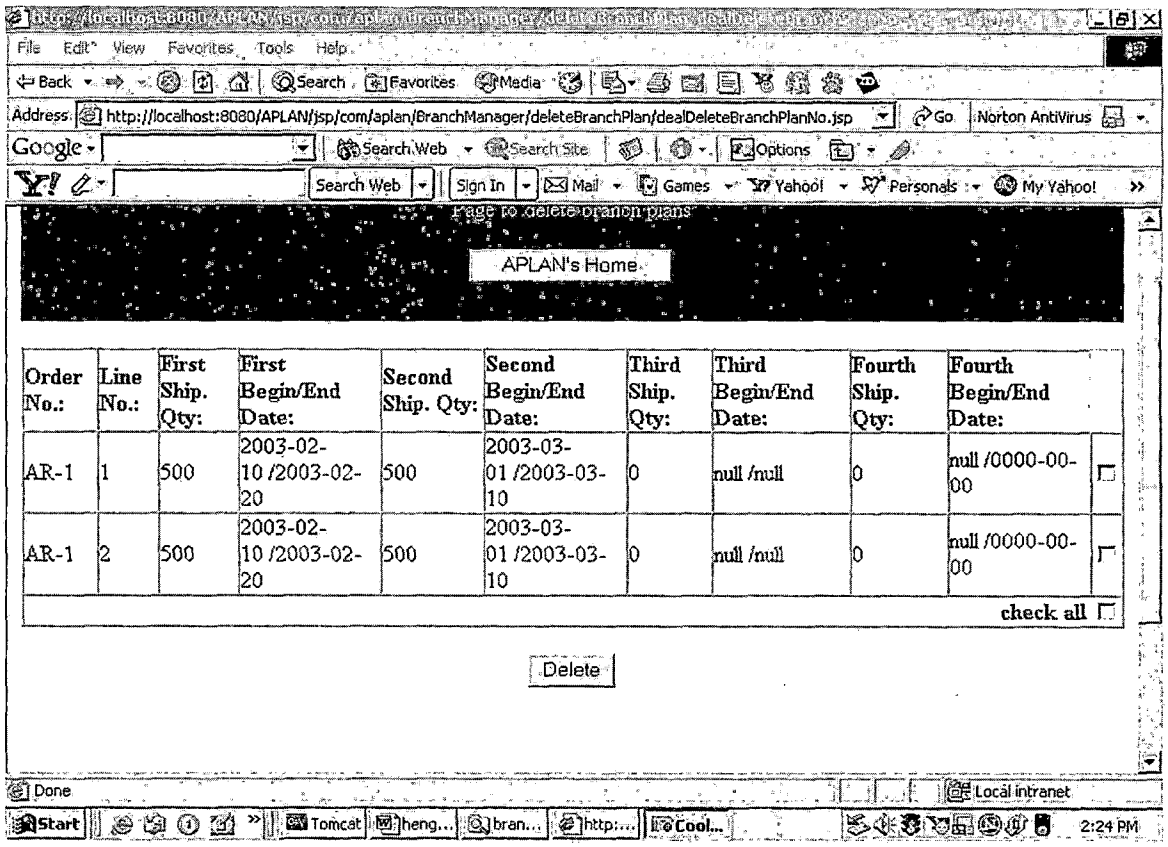


Figure 36. Page to Delete Branch's Plan

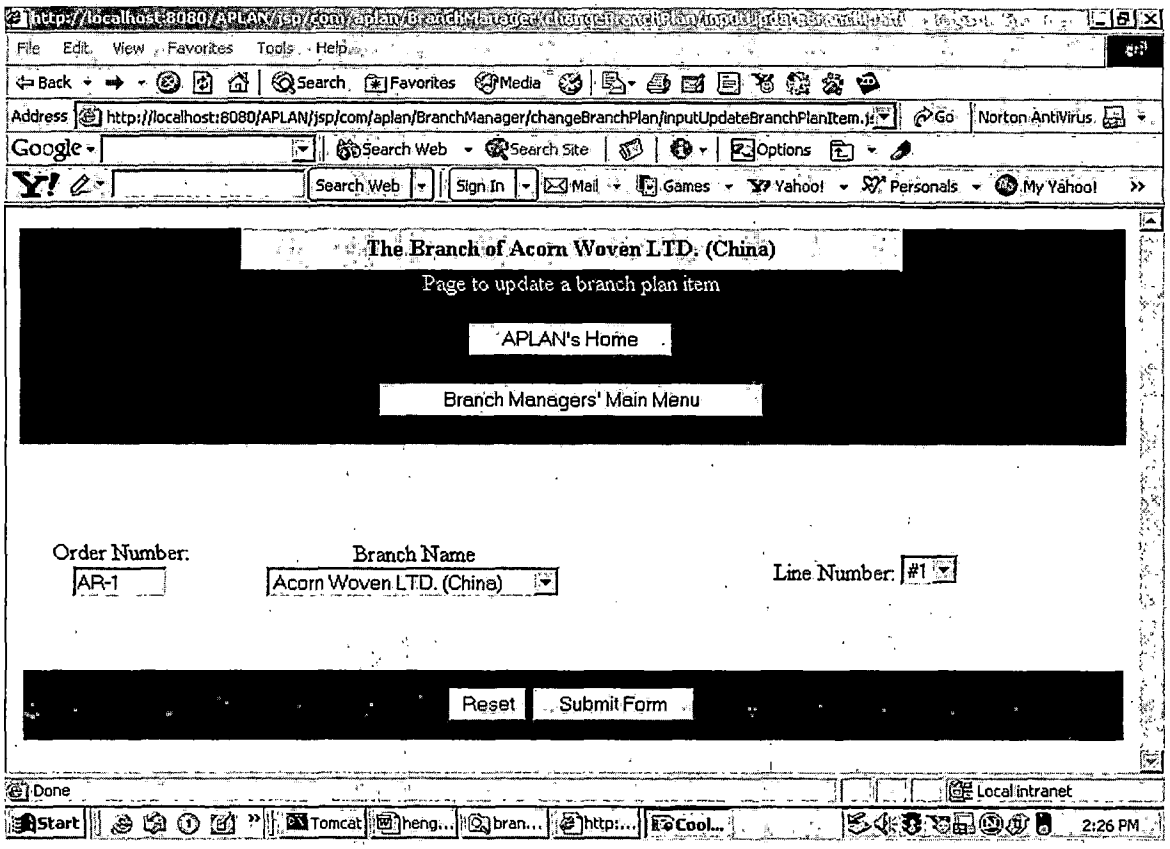


Figure 37. Enter Branch Plan for Updating

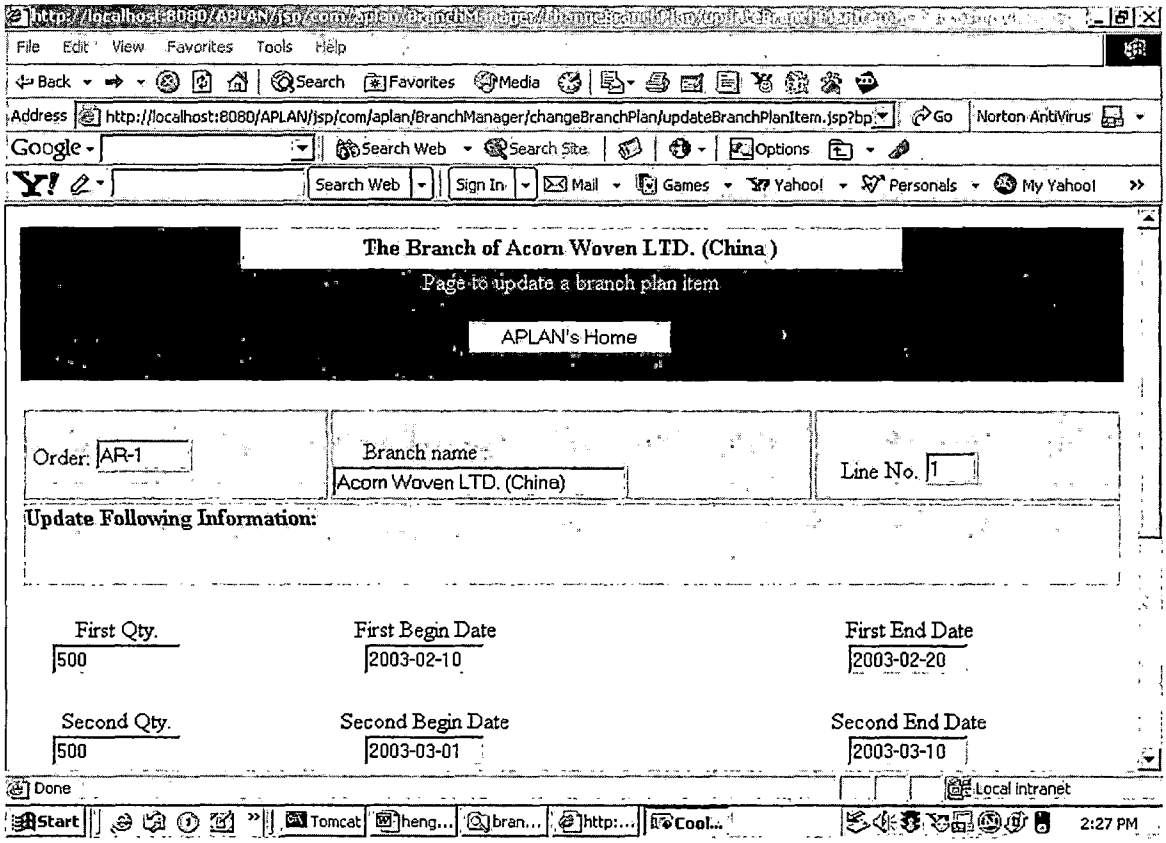


Figure 38. Page to Update Branch's Plan

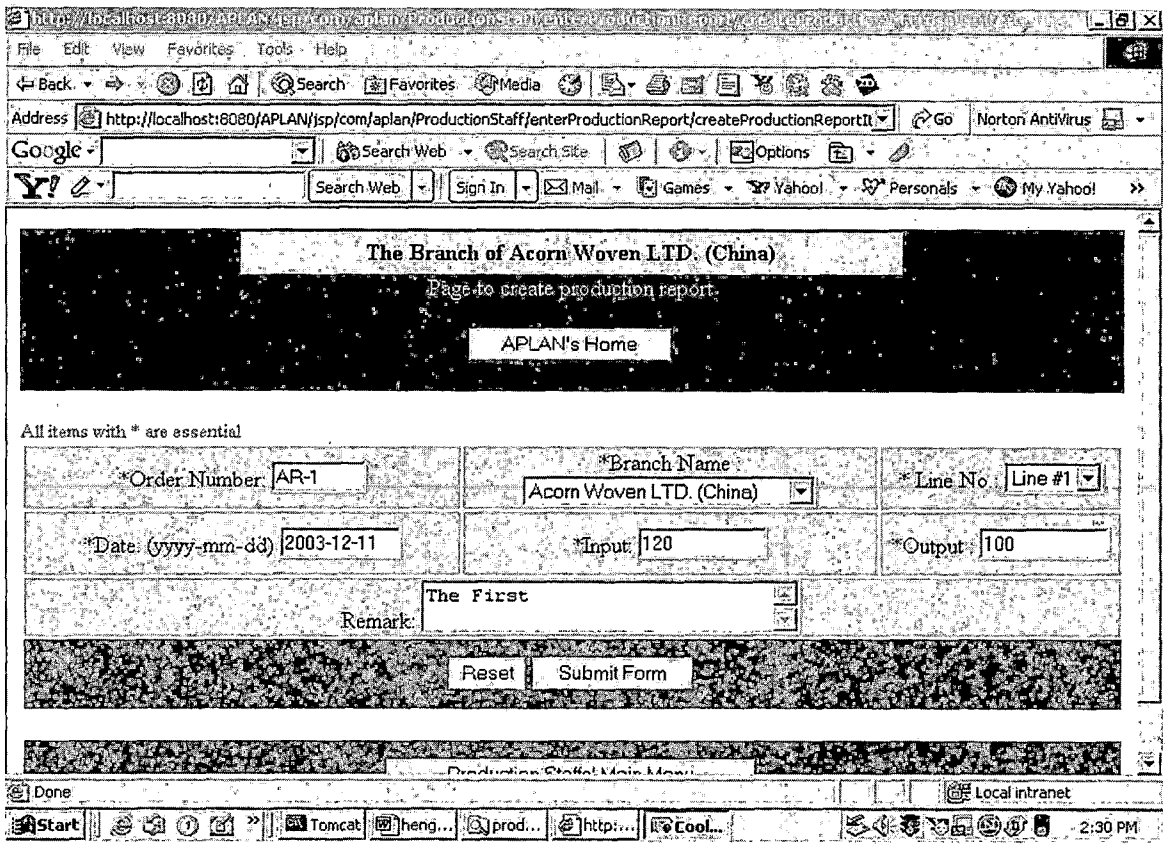


Figure 39. Page to Create Production Report

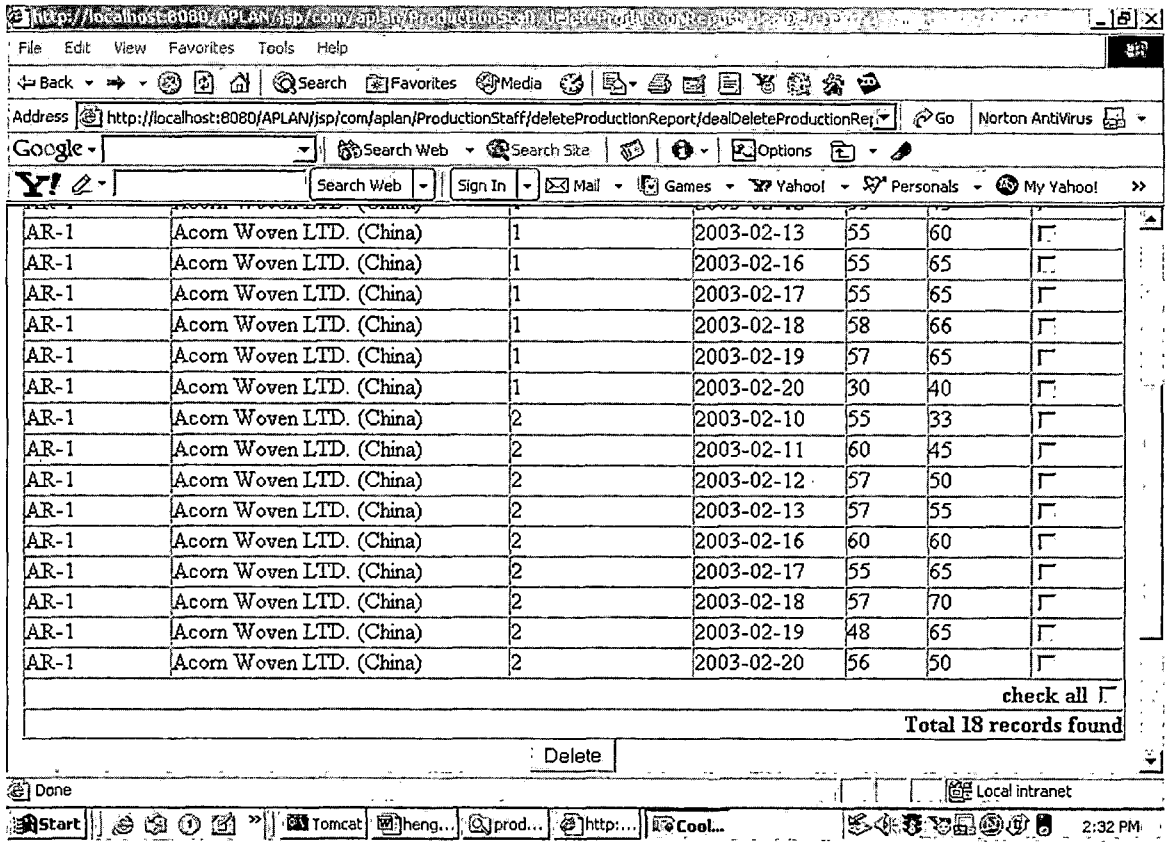


Figure 40. Page to Delete Production Report

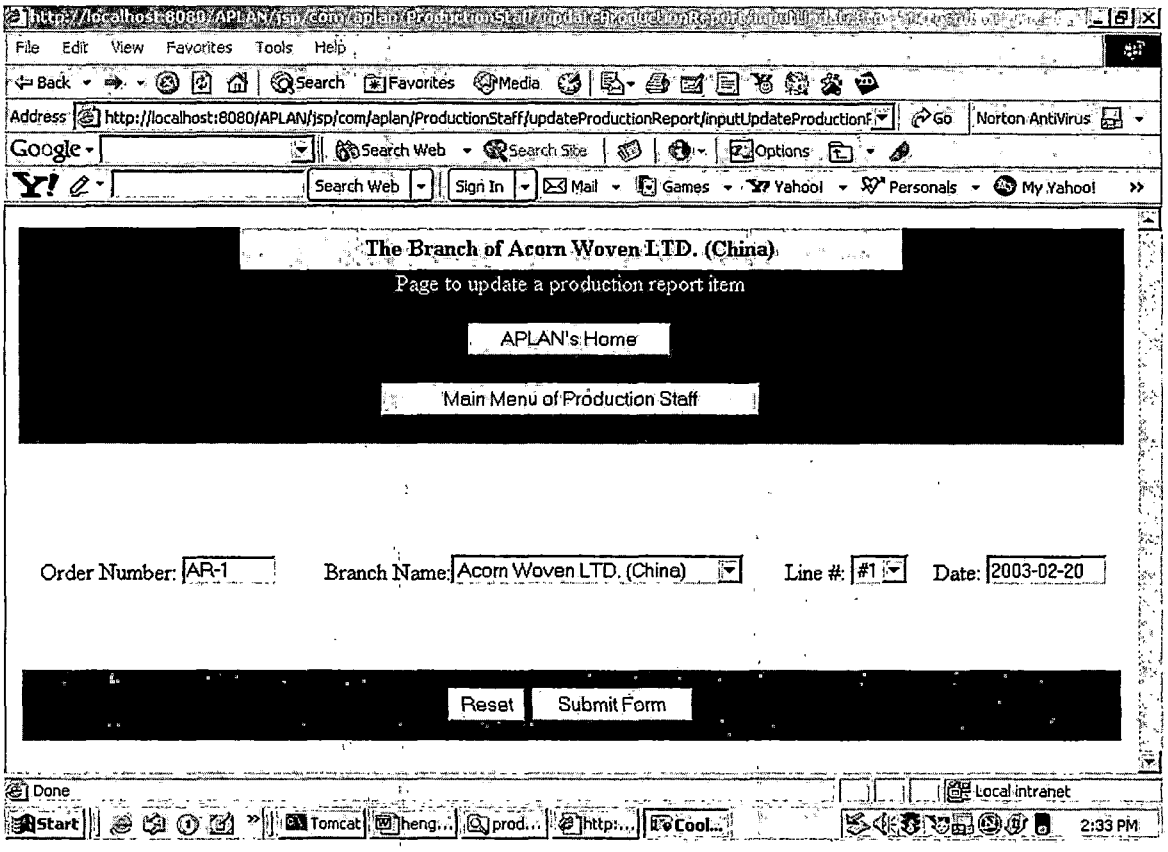


Figure 41. Enter Items to Update Production Report

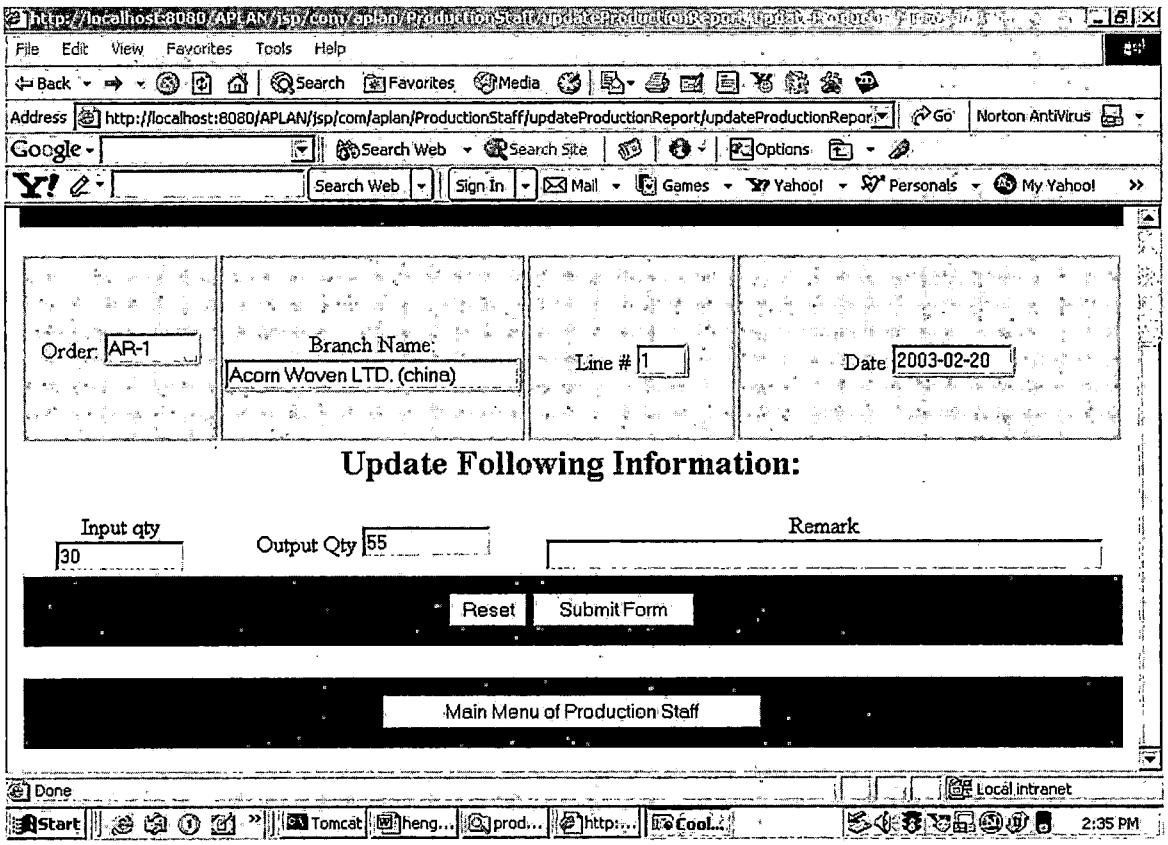


Figure 42. Page to Update Production Report

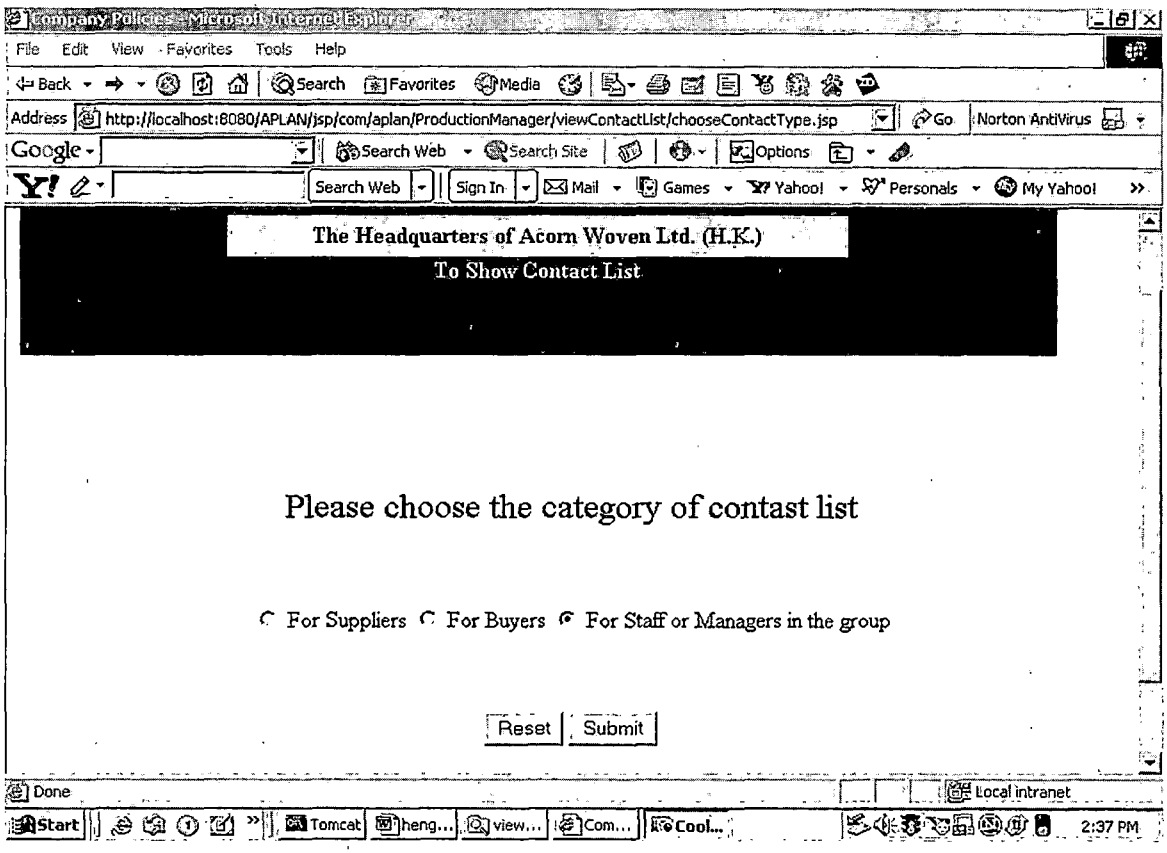


Figure 43. Page to Choose Contact List Type

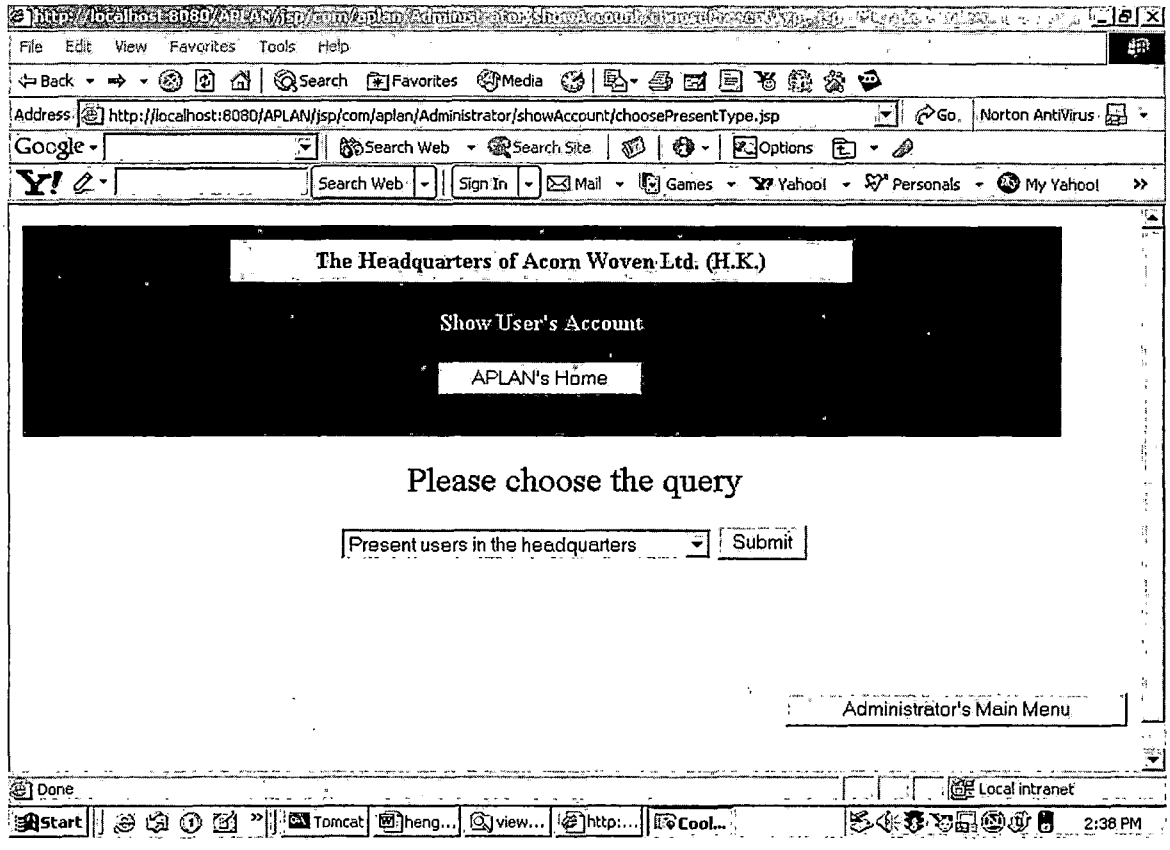


Figure 44. Page to Select User Account Type

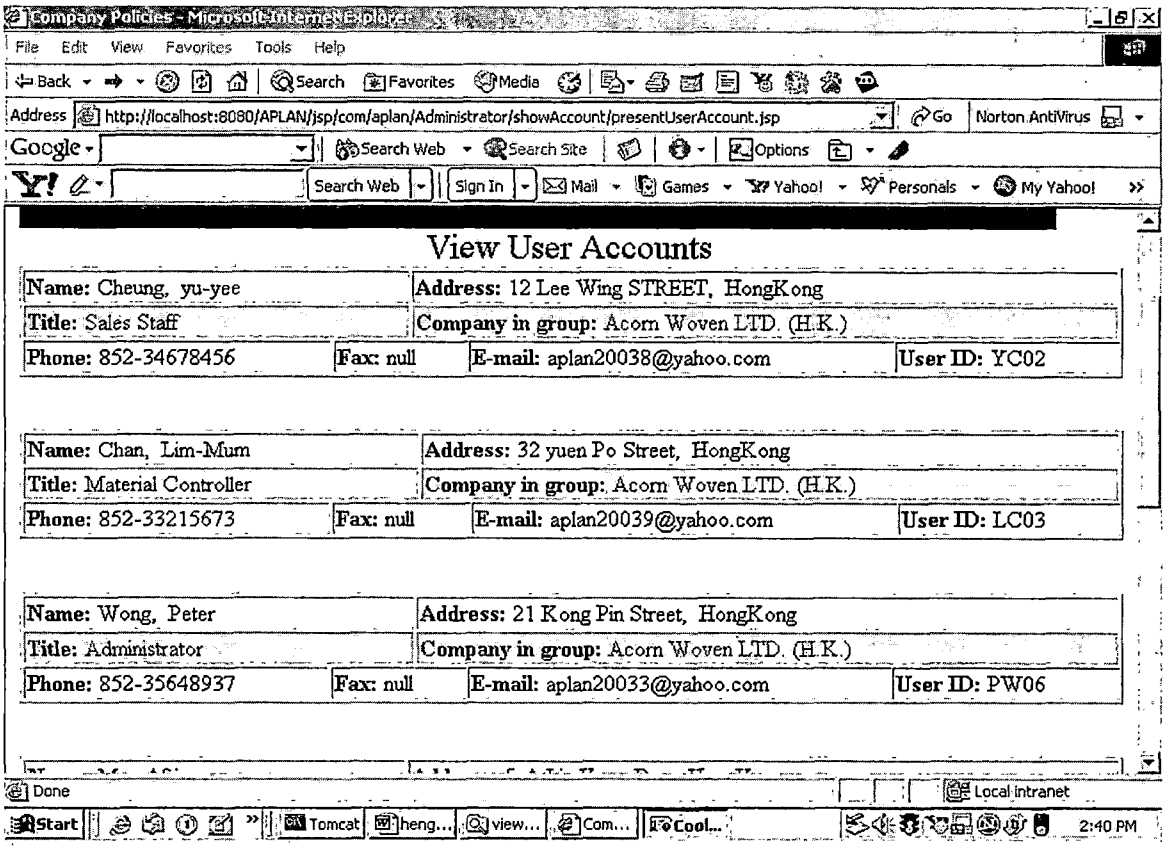


Figure 45. Page to Show Contact List

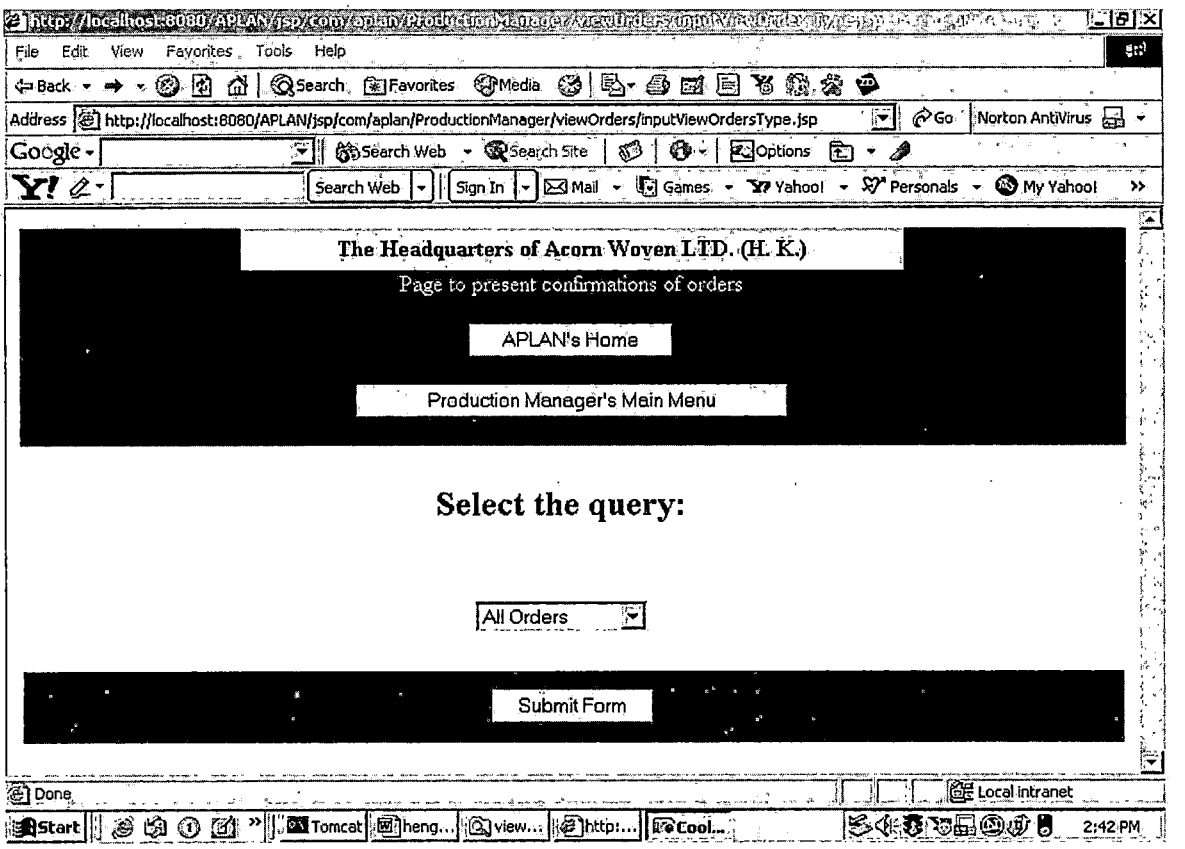


Figure 46. Choose View Type for Order

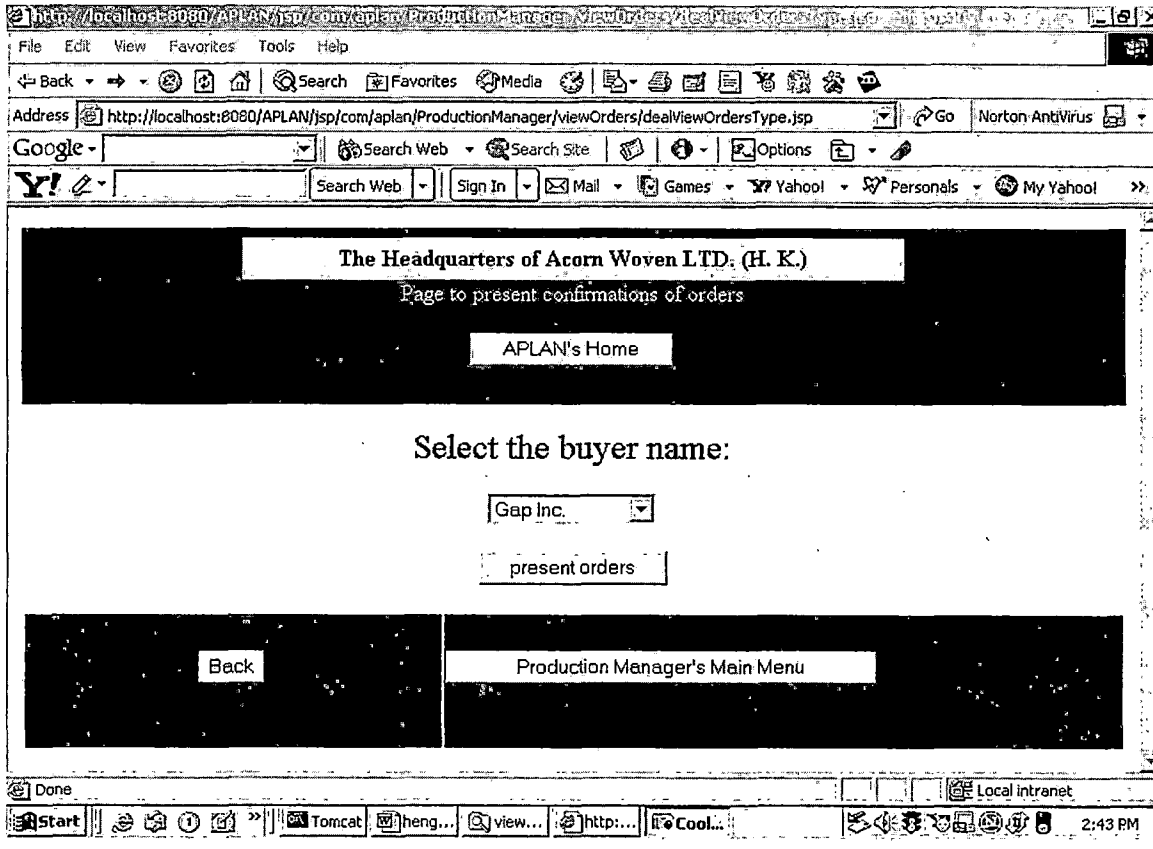


Figure 47. Select Buyer to View Order

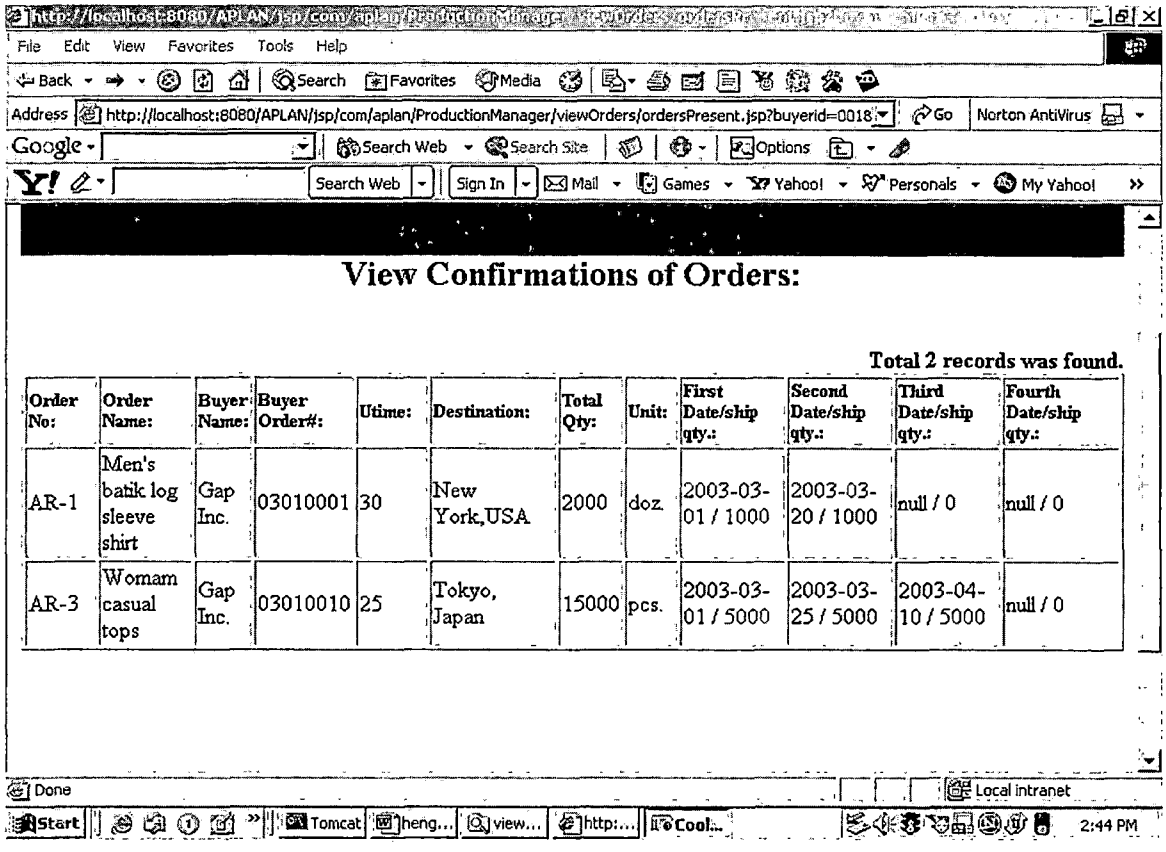


Figure 48. Page to Present Order Confirmation

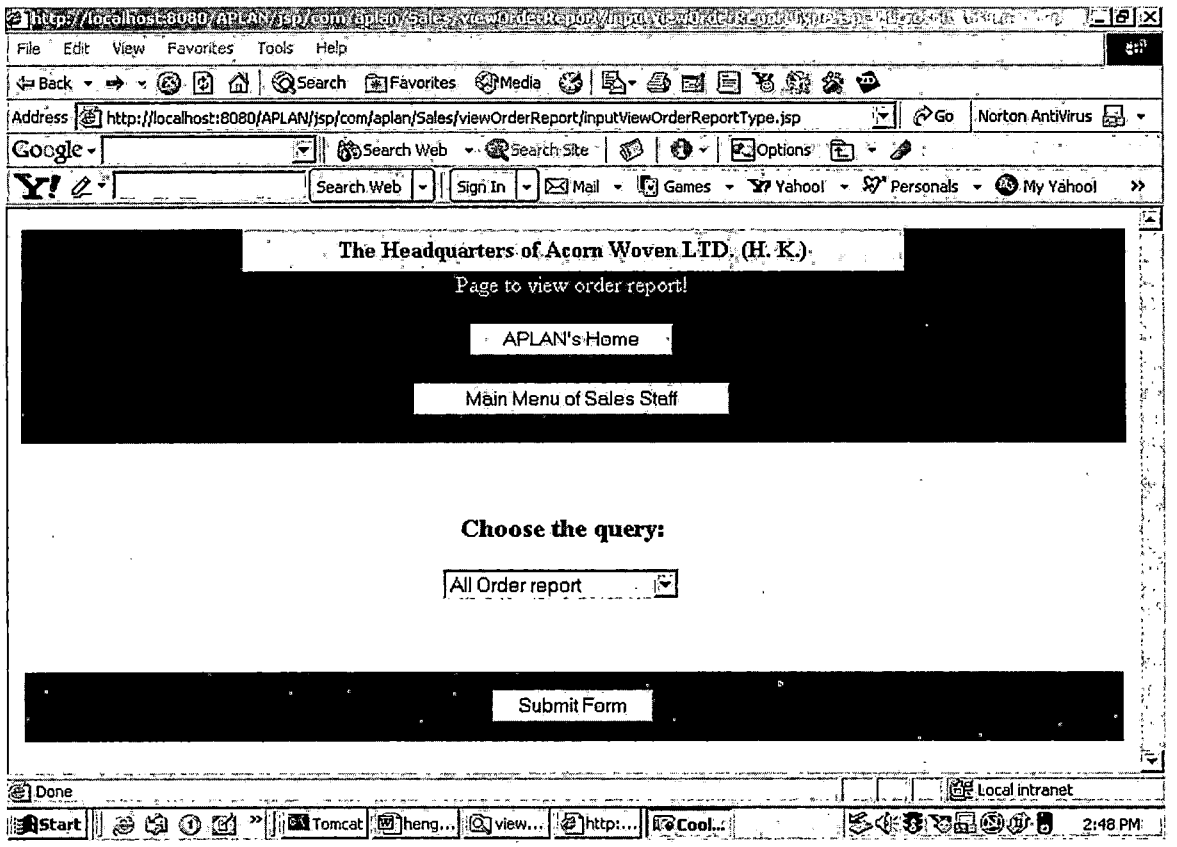


Figure 49. Choose View Type for Order Report

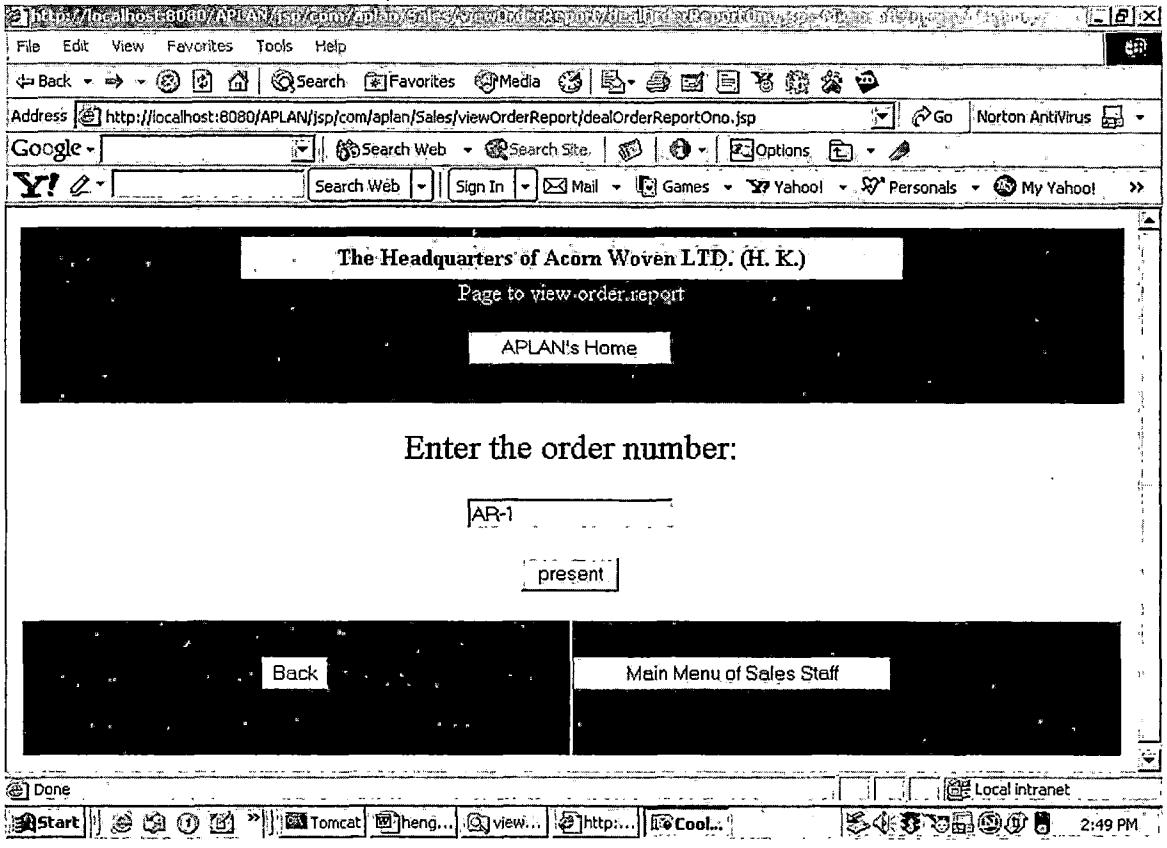


Figure 50. Enter Order Number of Order Report

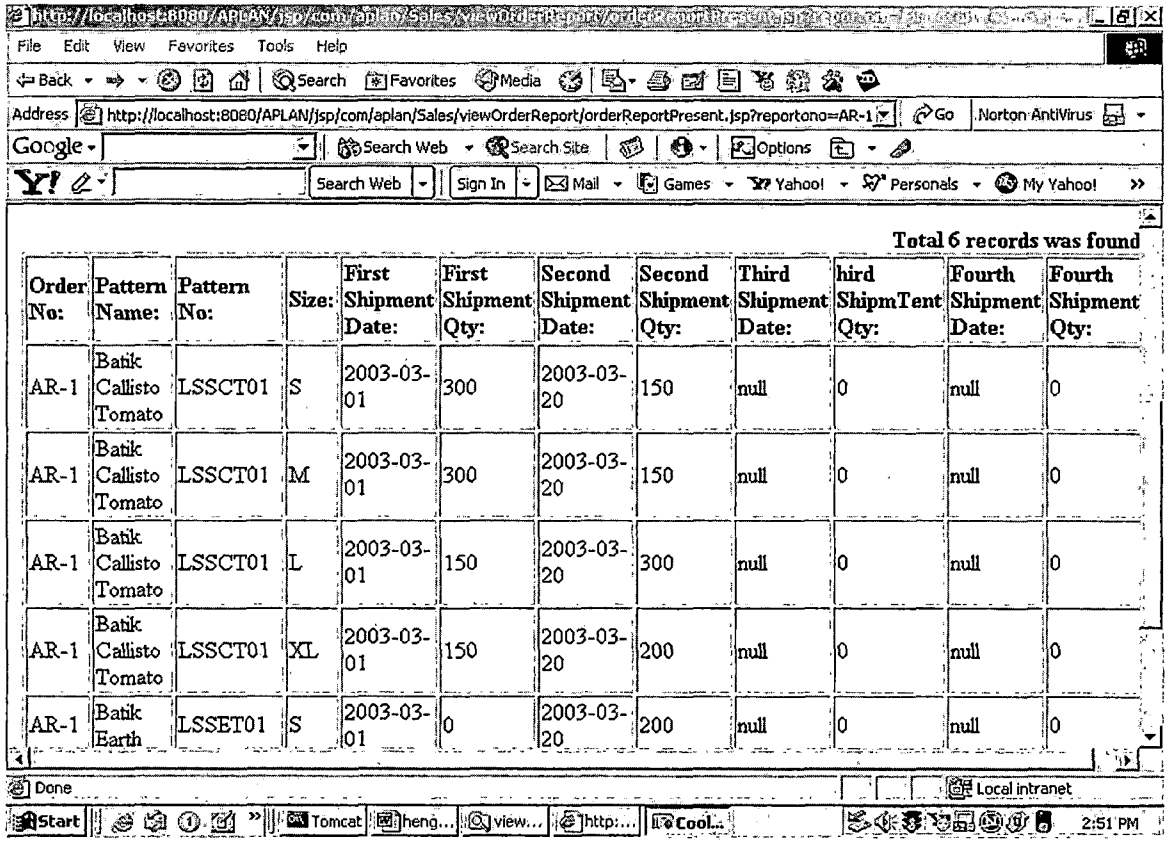


Figure 51. Page to Present Order Report

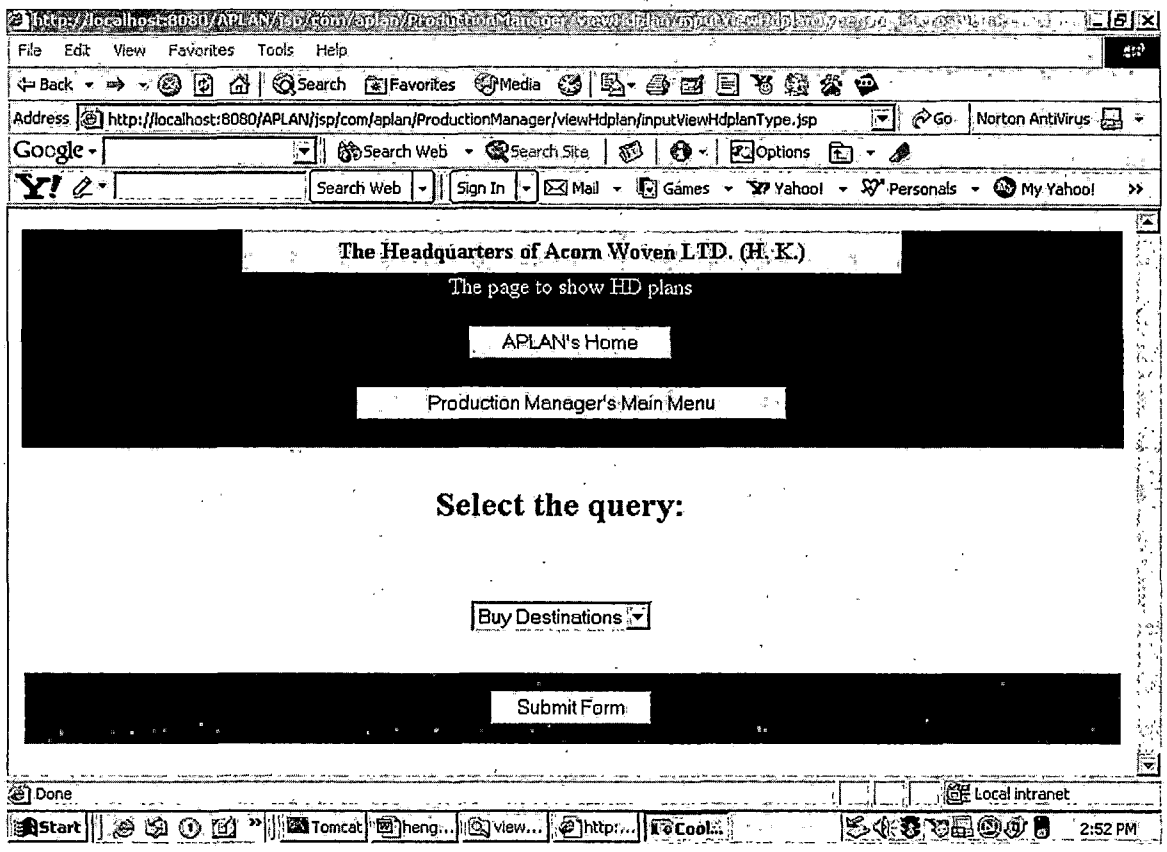


Figure 52. Page to Select View Type for HD Plan

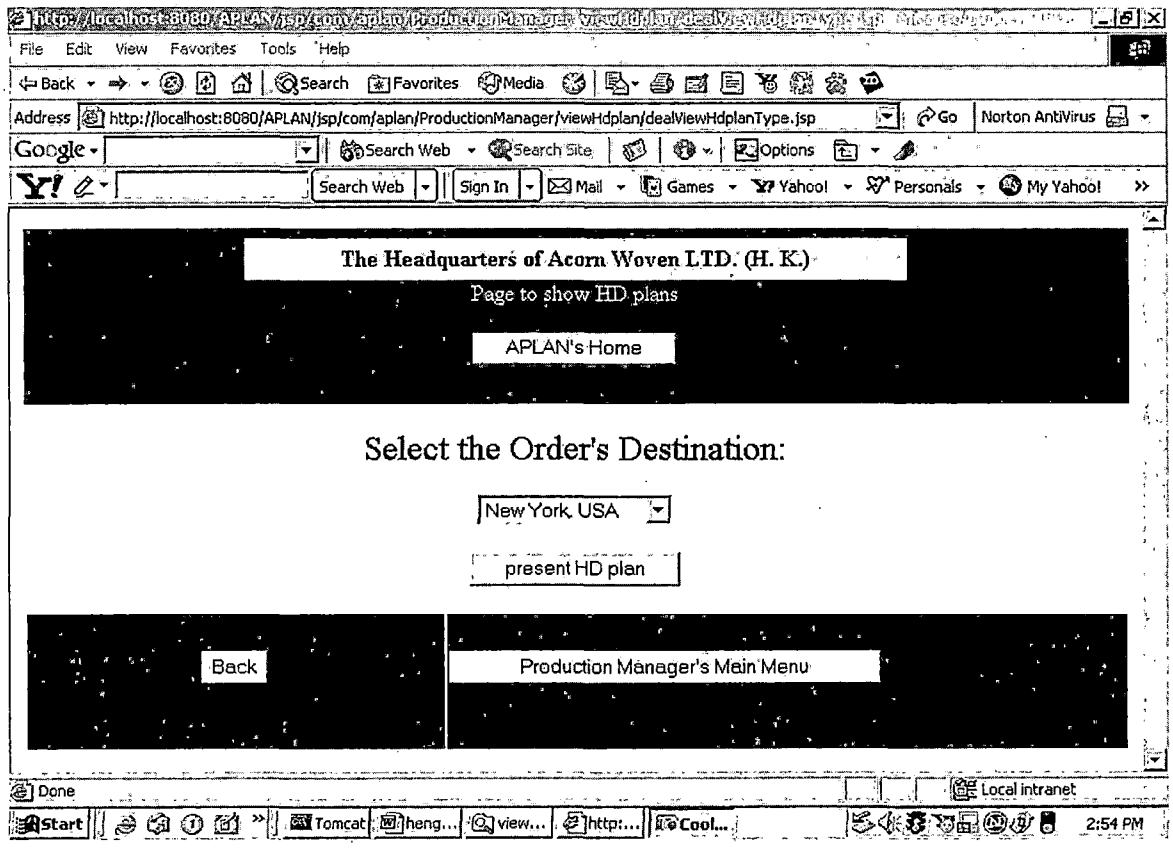


Figure 53. Select Destination to View HD Plan

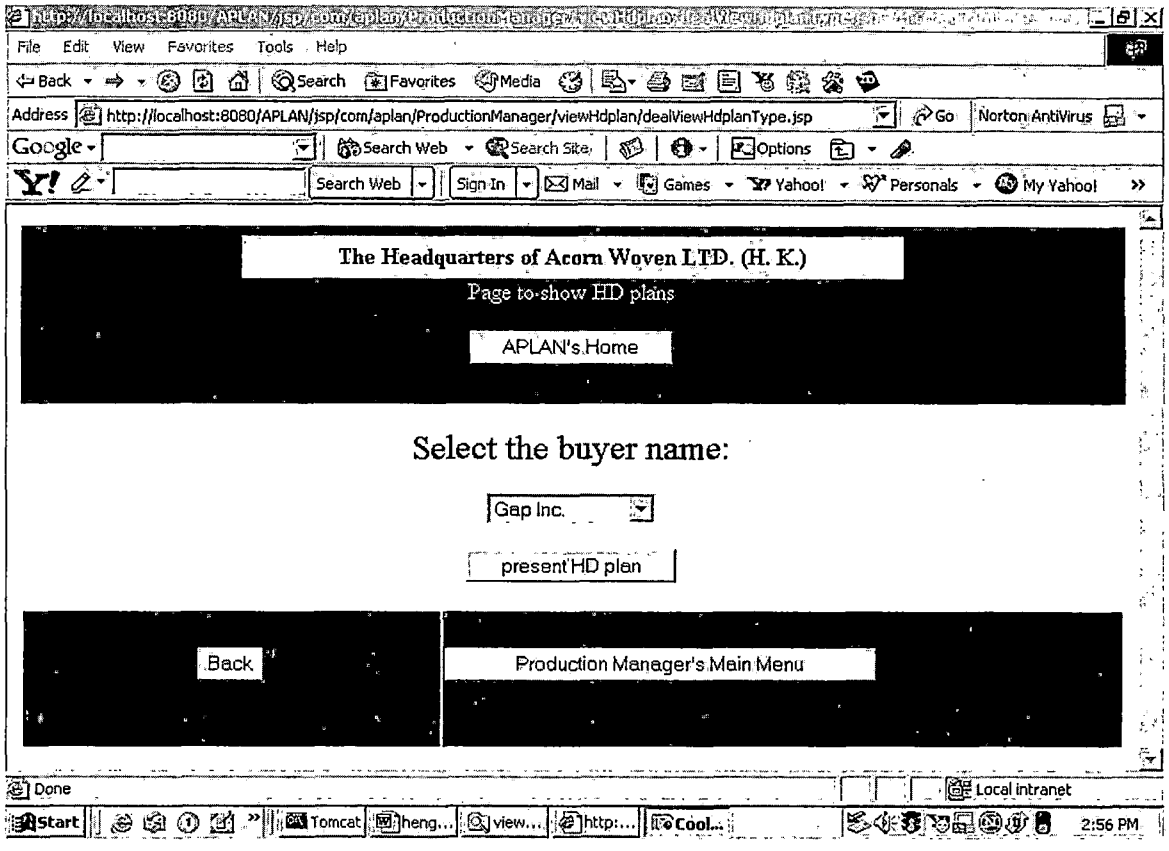


Figure 54. Select Buyer to View HD Plan

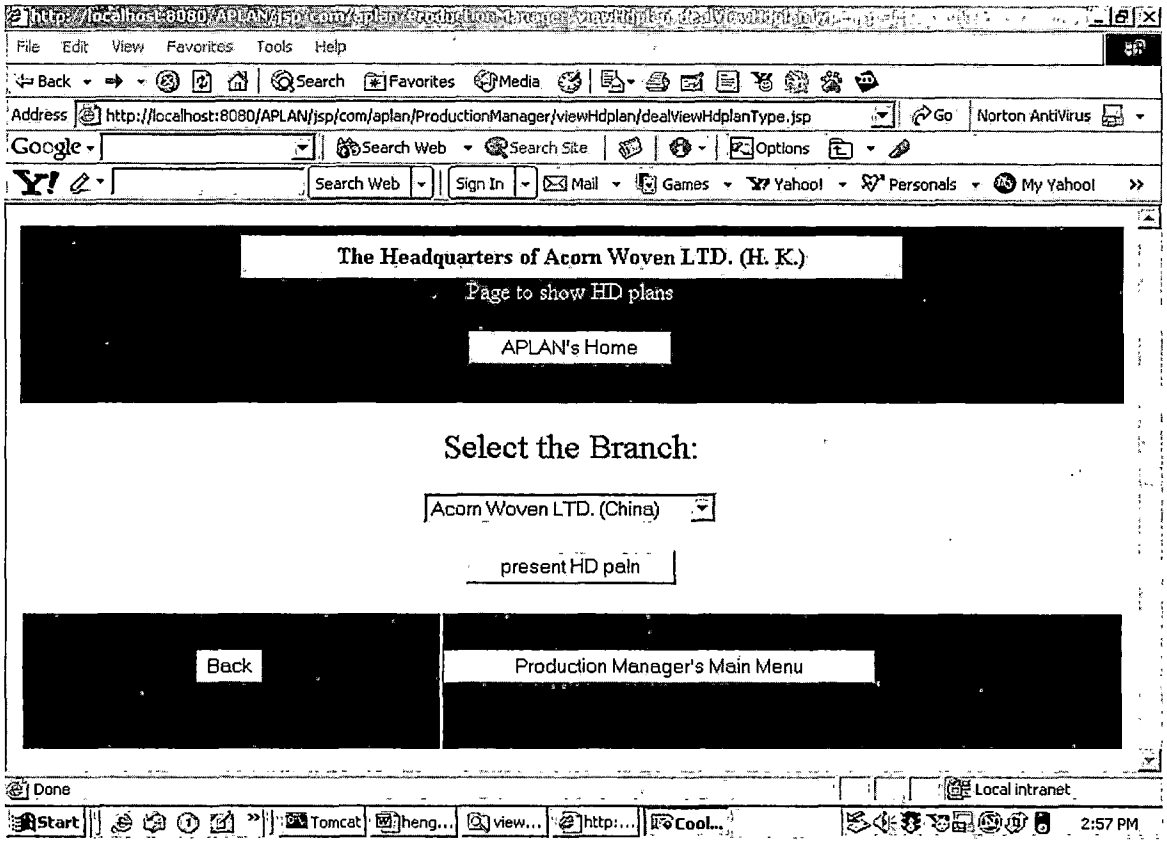


Figure 55. Select Branch to View HD Plan

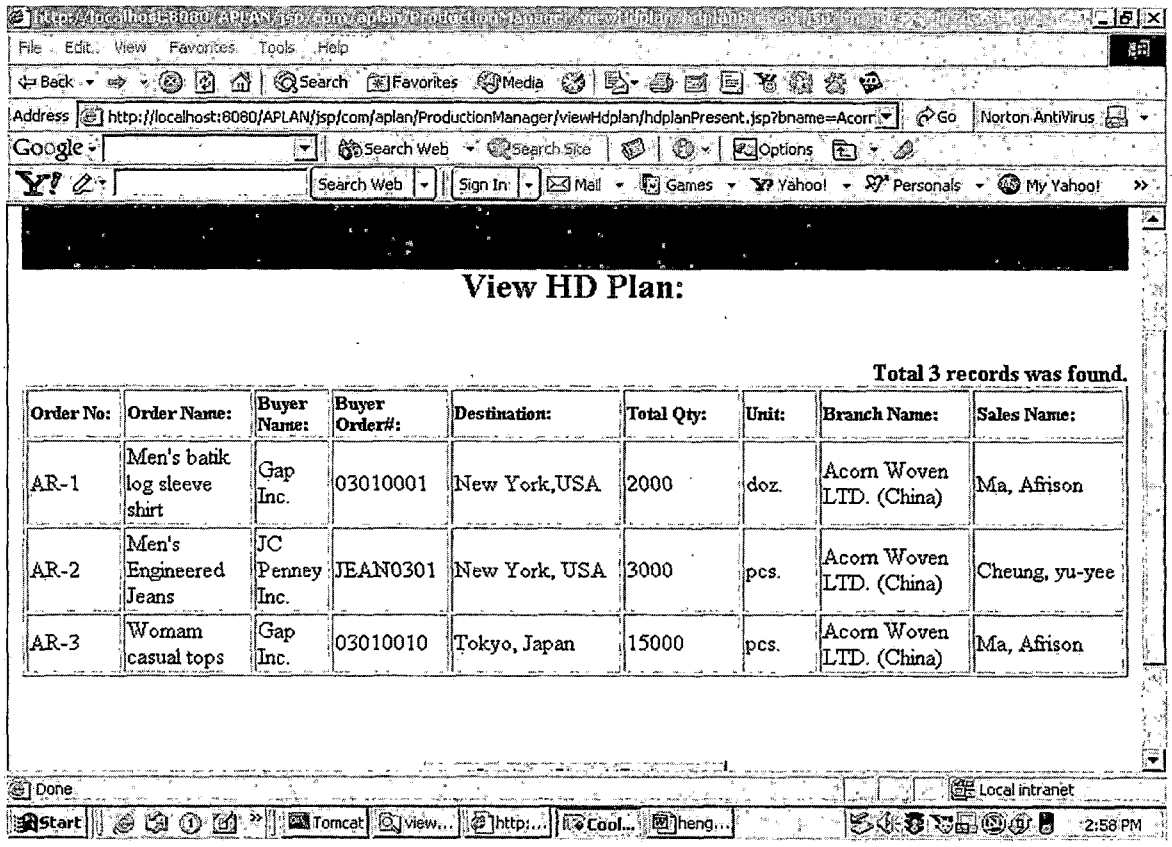


Figure 56. Page to Present HD Plan

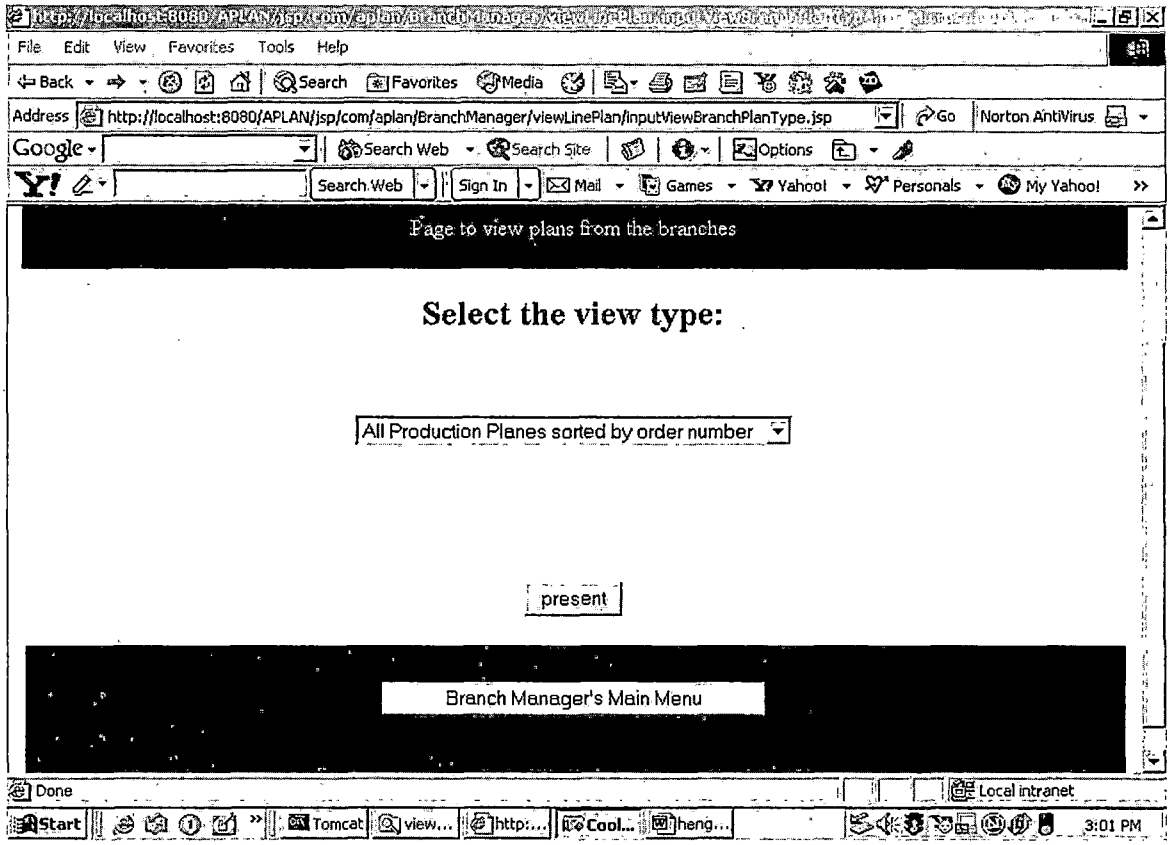


Figure 57. Choose Sort Type to View Branch Plan

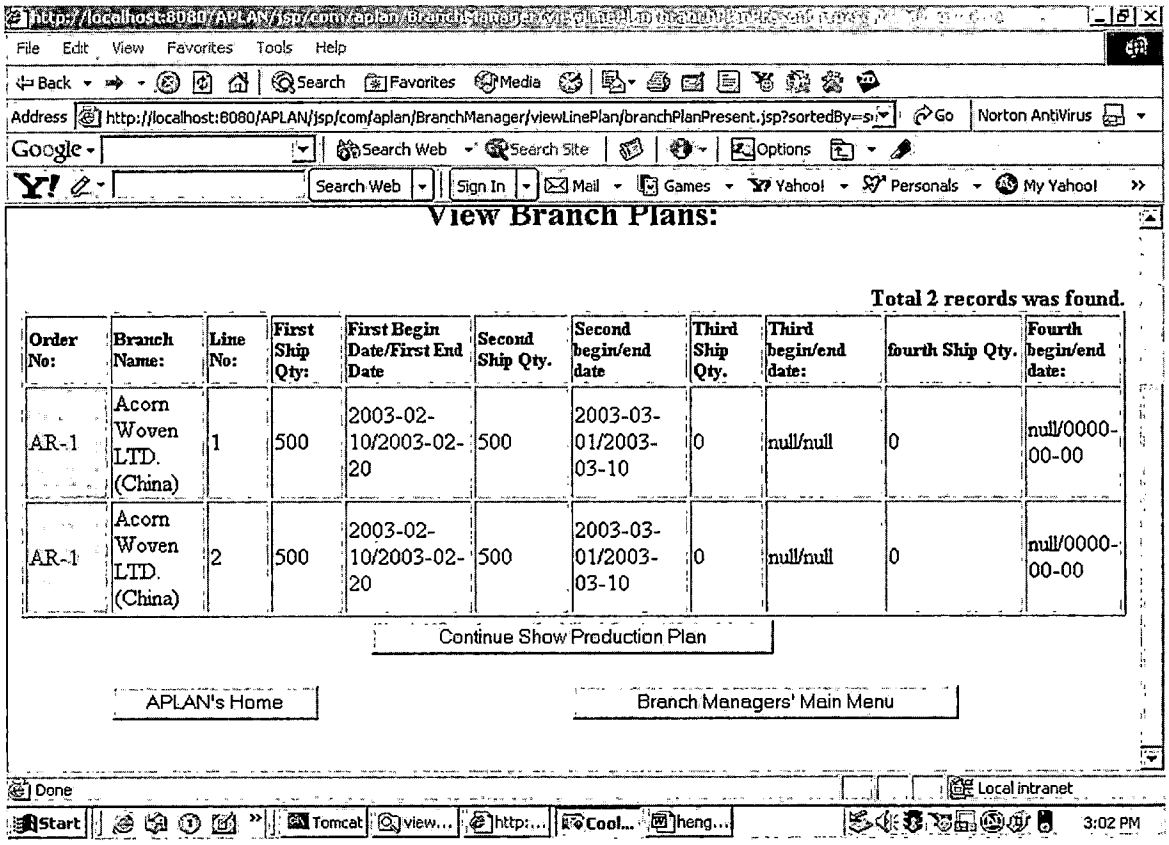


Figure 58. Page to Present Branch Plan

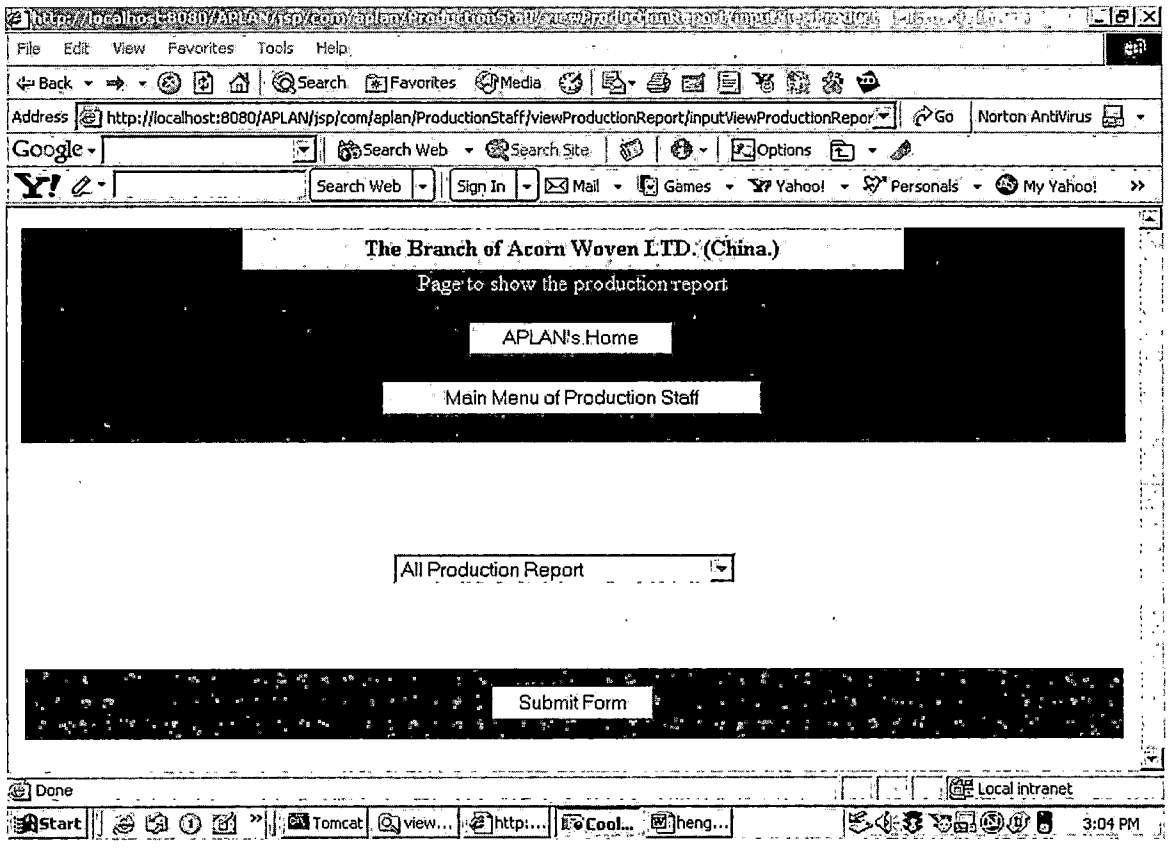


Figure 59. Select Type to View Production Report

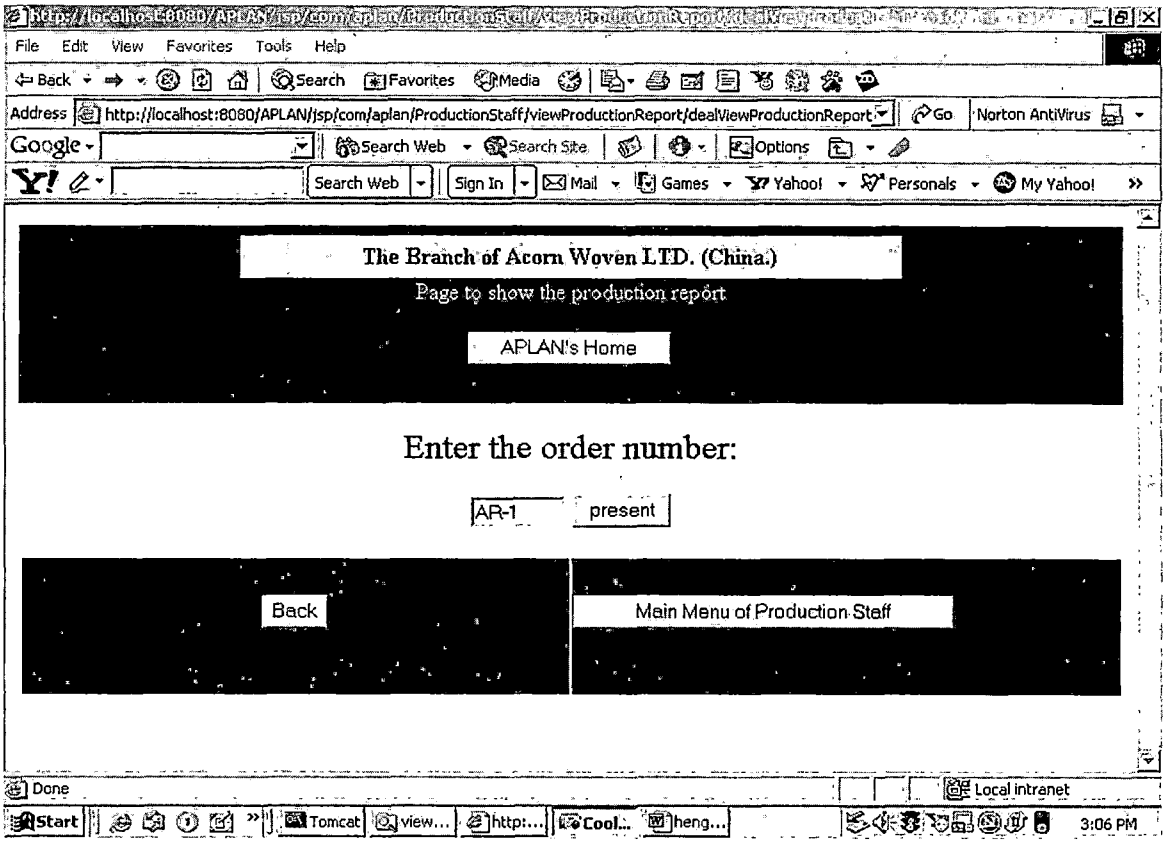
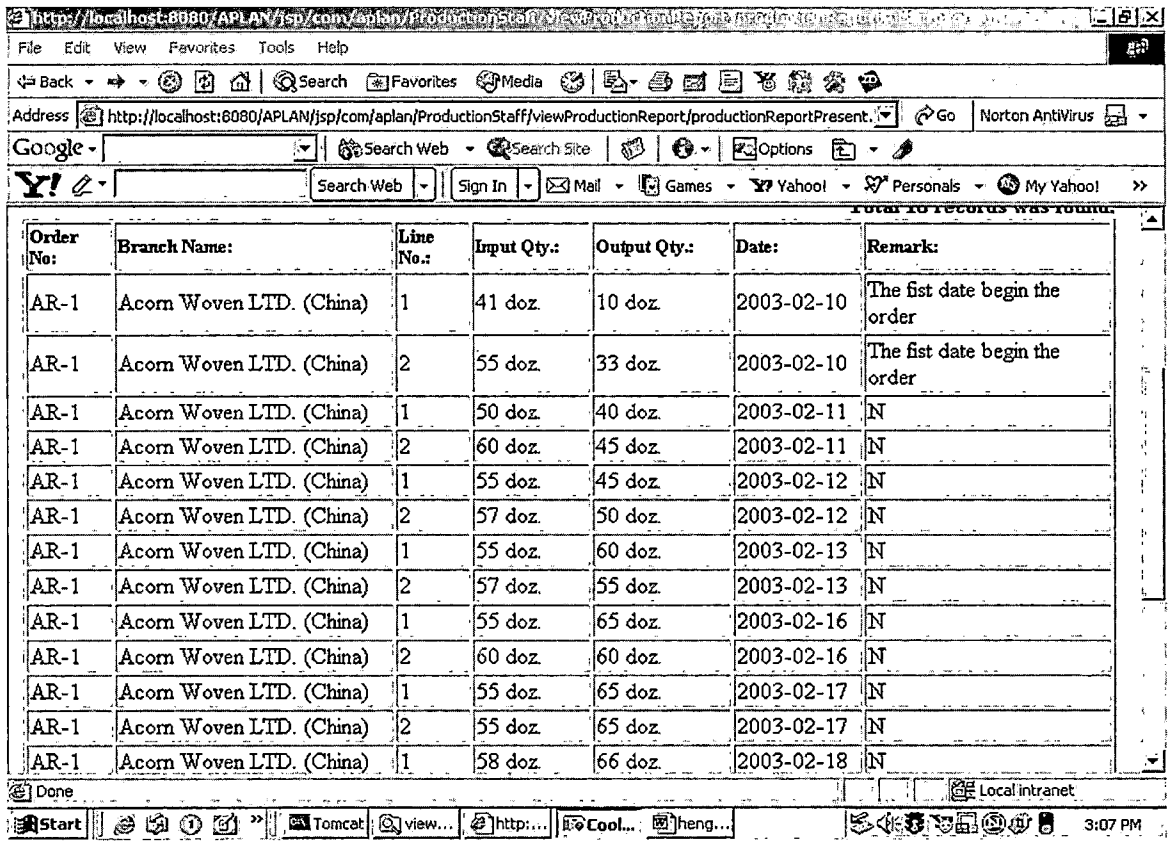


Figure 60. Enter Order to View Production Report



Order No:	Branch Name:	Line No.:	Input Qty.:	Output Qty.:	Date:	Remark:
AR-1	Acorn Woven LTD. (China)	1	41 doz.	10 doz.	2003-02-10	The fist date begin the order
AR-1	Acorn Woven LTD. (China)	2	55 doz.	33 doz.	2003-02-10	The fist date begin the order
AR-1	Acorn Woven LTD. (China)	1	50 doz.	40 doz.	2003-02-11	N
AR-1	Acorn Woven LTD. (China)	2	60 doz.	45 doz.	2003-02-11	N
AR-1	Acorn Woven LTD. (China)	1	55 doz.	45 doz.	2003-02-12	N
AR-1	Acorn Woven LTD. (China)	2	57 doz.	50 doz.	2003-02-12	N
AR-1	Acorn Woven LTD. (China)	1	55 doz.	60 doz.	2003-02-13	N
AR-1	Acorn Woven LTD. (China)	2	57 doz.	55 doz.	2003-02-13	N
AR-1	Acorn Woven LTD. (China)	1	55 doz.	65 doz.	2003-02-16	N
AR-1	Acorn Woven LTD. (China)	2	60 doz.	60 doz.	2003-02-16	N
AR-1	Acorn Woven LTD. (China)	1	55 doz.	65 doz.	2003-02-17	N
AR-1	Acorn Woven LTD. (China)	2	55 doz.	65 doz.	2003-02-17	N
AR-1	Acorn Woven LTD. (China)	1	58 doz.	66 doz.	2003-02-18	N

Figure 61. Page to Present Production Report

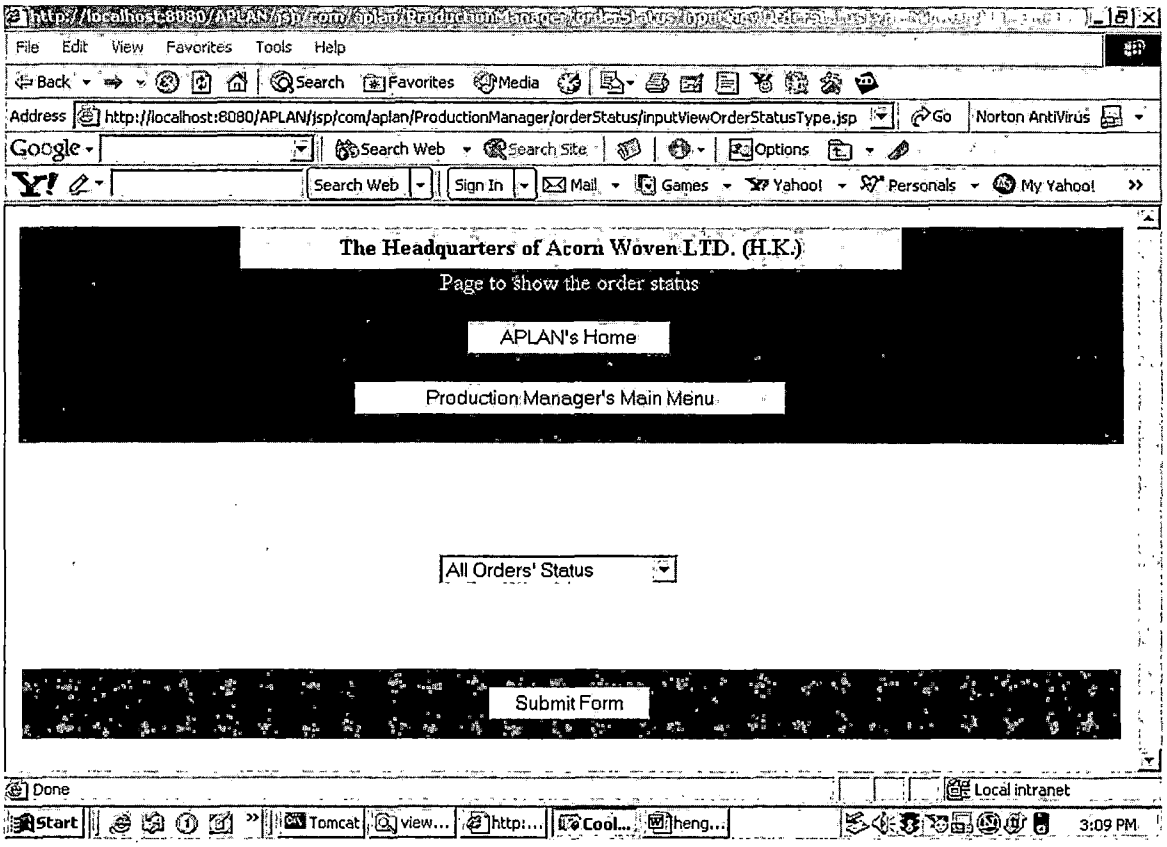


Figure 62. Select View Type of Order Status

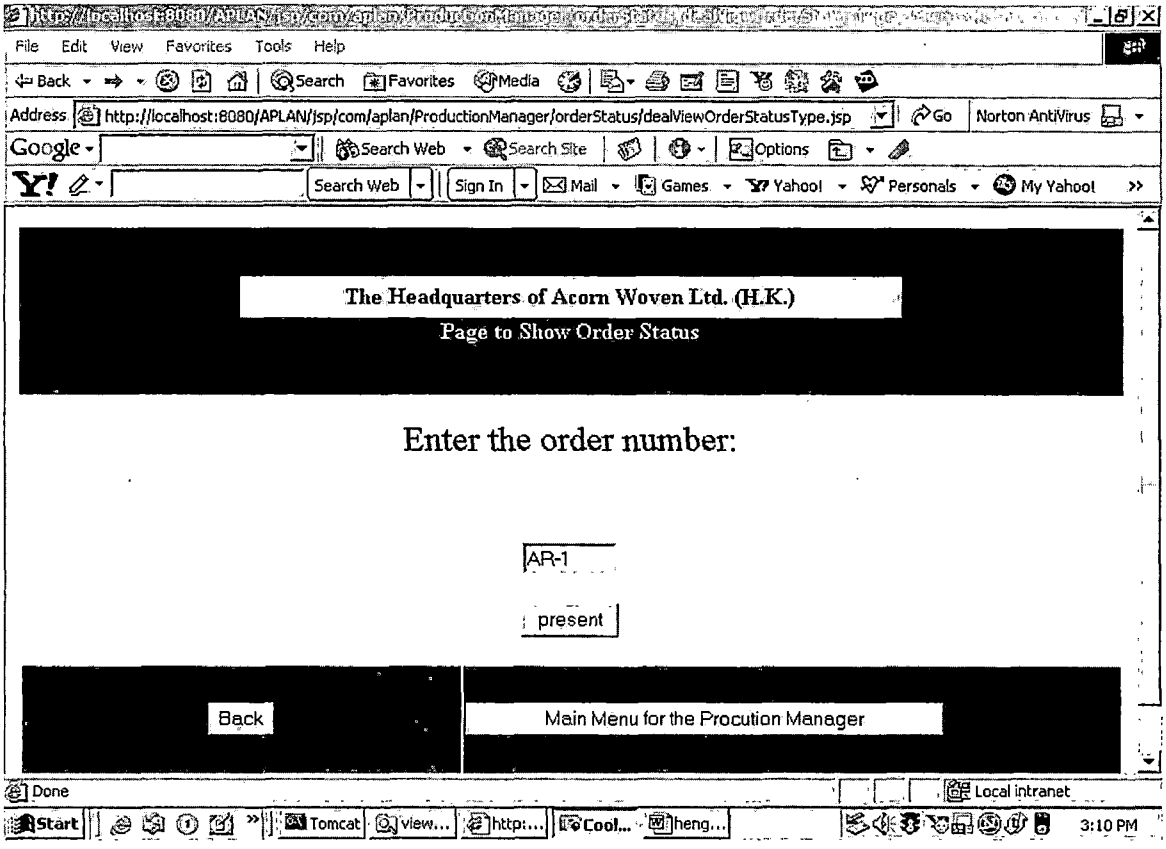


Figure 63. Enter Order to View Order Status

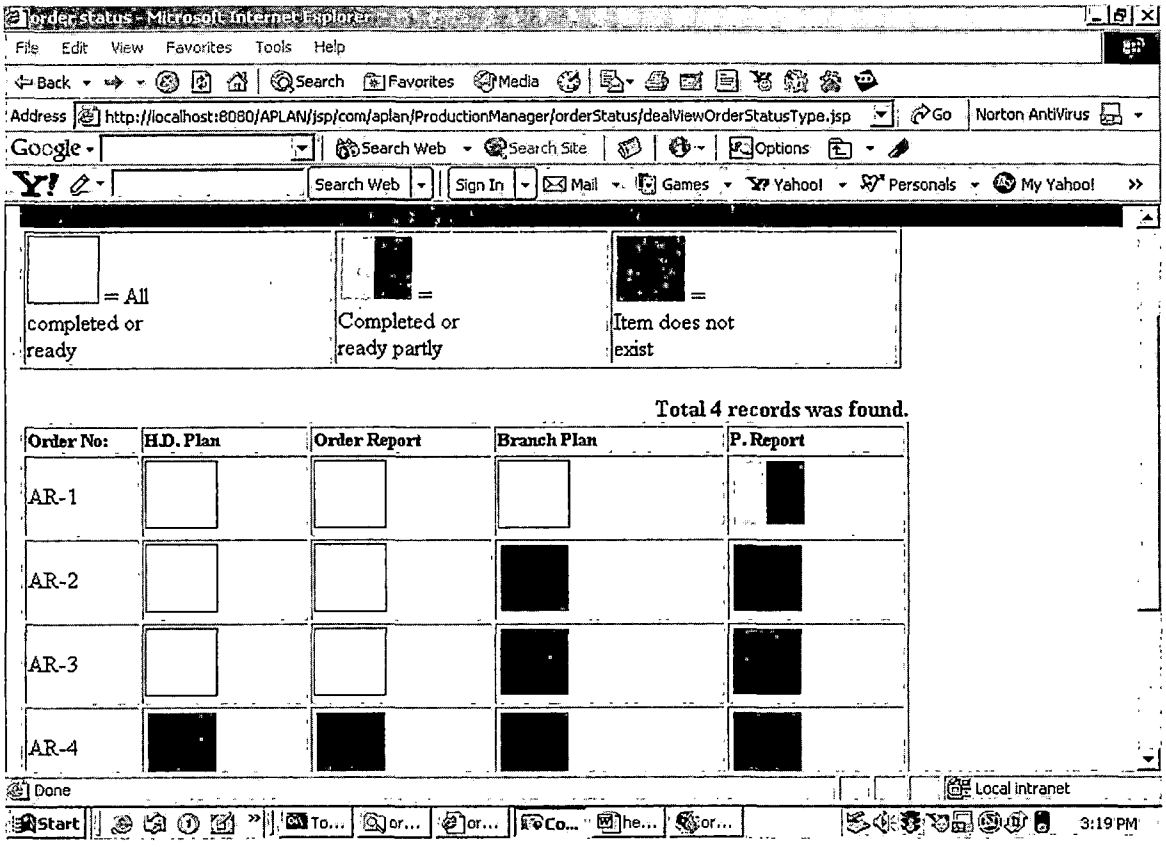


Figure 64. Page to Present Order Status

APPENDIX C

JAVA DESIGN

```
/** File Name:  Orderreport.java **/

package com.aplan.orderreport;

import java.sql.*;
import java.util.HashMap;
import com.aplan.users.AplanException;
import com.aplan.orders.Orders;

public class Orderreport {

    private String reportono, patternname, patternno, size;
    private int reportfqty, reportsqty, reportthdqty, reportfthqty;

    public void setReportono( String aono )
    {
        reportono = aono;
    }

    public String getReportono() { return reportono; }

    public void setPatternname( String aname)
    {
        patternname = aname;
    }

    public String getPatternname() { return patternname; }

    public void setPatternno( String ano )
    {
        patternno = ano;
    }

    public String getPatternno() { return patternno; }


    public void setSize(String asize)
    { size = asize;}

    public String getSize(){ return size;}

    public void setReportfqty(int afqty)
    { reportfqty = afqty;}

    public int getReportfqty() { return reportfqty;}

    public void setReportsqty(int asqty)
    { reportsqty = asqty;}

    public int getReportsqty() { return reportsqty;}

    public void setReportthdqty(int athdqty)
    { reportthdqty = athdqty;}

    public int getReportthdqty() { return reportthdqty;}

    public void setReportfthqty(int afthqty)
    { reportfthqty = afthqty;}

    public int getReportfthqty() { return reportfthqty;}

    public Orderreport() {
    }

    public Orderreport(String reportono, String patternname, String patternno,
                       String size, int reportfqty, int reportsqty, int reportthdqty,
                       int reportfthqty)

    { this.reportono = reportono;
      this.patternname = patternname;
```

```
    this.patternno = patternno;
    this.size = size;
    this.reportfqty = reportfqty;
    this.reportsqty = reportsqty;
    this.reportthdqty = reportthdqty;
    this.reportfthqty = reportfthqty;

}


public void insertOrderreport() throws AplanException
{
    Connection conn = null;
    Statement st = null;

    try
    {   Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        {   throw new AplanException ("The data base can't be connected");}

        String sql = "INSERT INTO orderreport VALUES " +
                     "(?, ?, ?, ?, ?, ?)";
       PreparedStatement statement = conn.prepareStatement(sql);
       statement.setString(1, reportono);
       statement.setString(2, patternname);
       statement.setString(3, patternno);

      statement.setString(4, size);
      statement.setInt(5, reportfqty);
      statement.setInt(6, reportsqty);

       statement.executeQuery();
    }
    catch (Exception e) {System.out.print("error");}
    finally
    {
        try{    conn.close(); }
        catch (Exception e)
        {   System.out.println("Exception while releasing DB connection"
            + e.getMessage());

       }
     }
}


public void deleteOrderreport(String aono) throws AplanException
{
    Connection conn = null;
    Statement st = null;
    ResultSet rs = null;
    try
    {
        Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");
        if (conn == null) { throw new AplanException ("The data base can't be
                            connected");}
        st = conn.createStatement();
        String sql = "DELETE FROM orderreport WHERE reportono = \'" + aono + "\'";
       rs = st.executeQuery(sql);
    }
    catch (Exception e) {System.out.print("error while delete an order report");}
    finally
    {
       try
       {
```

```
            conn.close();
        }
        catch (Exception e)
        {   System.out.println("Exception while releasing DB connection" +
            e.getMessage());

        }
    }
}


public void deleteOps(String aono, String apatternno, String asize)
                    throws AplanException
{
    Connection conn = null;
    Statement st = null;
    ResultSet rs = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        { throw new AplanException ("The data base can't be connected");}

        st = conn.createStatement();
        String sql = "DELETE FROM orderreport WHERE reportono = \'" + aono + "\'"
             + "and  patternno = \'" + apatternno + "\'" +  " and size = \'" +
             asize + "\'";
         rs = st.executeQuery(sql);

    }
    catch (Exception e) {System.out.print("error while delete a plan");}
    finally
    {
        try {  conn.close();}
        catch (Exception e)
        {    System.out.println("Exception while releasing DB connection" +
             e.getMessage());

        }
    }
}


public void updateOrderreport(String aorder, String apatternno, String asize)
{
    Connection conn = null;
    try
    {   Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        String sql = "UPDATE orderreport SET patternname=?, reportfqty=?,
                    reportsqty=?";
        PreparedStatement statement = conn.prepareStatement(sql);
        statement.setString(1, patternname);
        statement.setInt(2,reportfqty);
        statement.setInt(3,reportsqty);
        statement.executeQuery();
    }
    catch (Exception e) {System.out.print("error while update order report");}
    finally
    {
        try  {  conn.close(); }
        catch (Exception e)
        {   System.out.println("Exception while releasing DB connection"
           + e.getMessage());
```

```
        }
    }
}


private HashMap errorMessages = new HashMap();

public String getErrorType(String errorName) {
    return((String)errorMessages.get(errorName));
}

public void addErrorMessage(String errorName, String suggest) {
    errorMessages.put(errorName, suggest);
}


public boolean ifOrderreportFormError() throws AplanException
{
    errorMessages.clear();
    boolean error = false;
    if ((reportono == null) || (reportono.length() == 0)){
        addErrorMessage("reportono", "Order Number is essential.");
        error = true;
    }

    if ((patternno == null) || (patternno.length() == 0)) {
        addErrorMessage("patternno", "pattern number is essential.");
        error = true;
    }

    if ((size == null) || (size.length() == 0)) {
        addErrorMessage("size", "size is essential.");
        error = true;
   }

   return error;

}


public static Orderreport findOrderReportItem(String reportono, String patternno,
                          String size throws AplanException
{
    Connection conn = null;
    Statement st = null;
    String sql = "";
    Orderreport orderreport = null;
    try
    {   Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        {   throw new AplanException ("The data base can't be connected");}
        st = conn.createStatement();
        sql = "SELECT * FROM orderreport WHERE reportono = \'" + reportono+ "\'"
              + " and patternno=  \'" + patternno+ "\'" + " and size=\'" + size
              + "\'";
       ResultSet rs = st.executeQuery(sql);

        if (rs.next()) {
            orderreport = new Orderreport();
            orderreport.setPatternname(rs.getString("patternname"));
            orderreport.setReportfqty(rs.getInt("reportfqty"));
            orderreport.setReportsqty(rs.getInt("reportsqty"));

           orderreport.setReportfthqty(rs.getInt("reportfthqty"));
           orderreport.setReportthdqty(rs.getInt("reportthdqty"));
        }
        else { return null;}
        rs.close();
```

```
            st.close();
        }
        catch (Exception e)
        {
            System.out.println("System error in finding an order report item. "
            + e.getMessage());
        }
        finally
        {
            try
            {
                conn.close();
            }
            catch (Exception e)
            {
                System.out.println("Exception while releasing DB connection"
                + e.getMessage());
            }
        }
        return orderreport;
    }

    public static boolean findOrder(String aono) throws AplanException
    {
        Connection conn = null;
        Statement st = null;
        boolean find = true;

        try
        {   Class.forName("org.gjt.mm.mysql.Driver").newInstance();
            conn =
            DriverManager.getConnection("jdbc:mysql://localhost/aplan");

            if (conn == null)
            {   throw new AplanException ("The data base can't be connected");}

            st = conn.createStatement();
            String sql = "select * from orderreport where reportono  = \'" + aono
            + "\'";
            ResultSet rs = st.executeQuery(sql);

            if (!rs.next()) {
                find = false;
            }
            rs.close();
            st.close();
        }
        catch (Exception e)
        {
            System.out.println("System error in finding an order. " + e.getMessage());
        }
        finally
        {   try
            {
                conn.close();
            }
            catch (Exception e)
            {
                System.out.println("Exception while releasing DB connection"
                + e.getMessage());
            }
            return find;
        }
    }
}
```

Figure 65. Class of Order Report

```
/** File Name:  Hdplan.java **/

package com.aplan.hdplan;

import java.sql.*;
import java.util.HashMap;
import com.aplan.exceptions.AplanException;


public class Hdplan {

    private String salesid, ono, bname;
    public void setSalesid( String aSalesid )
    {
        salesid = aSalesid;
    }

    public String getSalesid() { return salesid; }

    public void setOno( String aOno )
    {
        ono = aOno;
    }


    public String getOno() { return ono; }

    public void setBname( String abname )
    {
        bname = abname;
    }

    public String getBname() { return bname; }

    public Hdplan() {}

    public Hdplan(String ono, String bname, String salesid)

    { this.ono = ono;
      this.bname = bname;
      this.salesid = salesid;

    }


    public static Hdplan findHdplanItem(String orderno) throws AplanException
    {
        Connection conn = null;
       Hdplan hdplan = null;

        try
        {   Class.forName("org.gjt.mm.mysql.Driver").newInstance();
            conn =
            DriverManager.getConnection("jdbc:mysql://localhost/aplan");

            if (conn == null)
            {   throw new AplanException ("The data base can't be connected");}
            String sql = "SELECT * FROM HDPLAN WHERE ono = " +
                         "(?)";
           PreparedStatement statement = conn.prepareStatement(sql);
          statement.setString(1, orderno);
          ResultSet rs = statement.executeQuery();
          if (rs.next()) {
             hdplan= new Hdplan();
             hdplan.setOno(orderno);
          }
          else
          {   return null;}
          rs.close();
          statement.close();
        }
```

```
    catch (Exception e)
    {
        System.out.println("System error in finding a Hd plan. " + e.getMessage());
    }
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection" +
                               e.getMessage());
        }
    }
    return hdplan;
}



public void insertPlan() throws AplanException{
    Connection conn = null;
    Statement st = null;


    try
    {   Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn = DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        { throw new AplanException ("The data base can't be connected");}
        String sql = "INSERT INTO hdplan VALUES " +
                 "(?, ?, ?)";
        PreparedStatement statement = conn.prepareStatement(sql);
        statement.setString(1, ono);
        statement.setString(2, bname);
        statement.setString(3, salesid);
        statement.executeQuery();
    }
    catch (Exception e) {System.out.print("error");}
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {    System.out.println("Exception while releasing DB connection"
            + e.getMessage());
        }
    }
}



public void deletePlan(String aorder) throws AplanException{
    Connection conn = null;
    Statement st = null;
    ResultSet rs = null;
    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");
        if (conn == null)
        {   throw new AplanException ("The data base can't be connected");}
            st = conn.createStatement();
            String sql = "DELETE FROM hdplan WHERE ono = \'" + aorder +"\'";
             rs = st.executeQuery(sql);
     }
```

```
     catch (Exception e) {System.out.print("error while delete a plan");}

     finally
     {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {    System.out.println("Exception while releasing DB connection"
                                + e.getMessage());
         }
    }
}


public void updatePlan(){
    Connection conn = null;
    Statement st = null;
    try{
        Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");
        String sql = "UPDATE hdplan SET " +
               "bname=? WHERE aono=?";
        PreparedStatement statement = conn.prepareStatement(sql);
        statement.setString(1, bname);

        statement.executeQuery();
    }
     catch (Exception e) {System.out.print("error while updating a plan");}

     finally
     {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {    System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
         }
    }
}


public static double calculateAbility(String coeff, String unittime, String cname)
                                    throws AplanException
{

    Connection conn = null;
     Statement stmt = null;
     ResultSet rs = null;
    int cability = 0;
    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
          conn =
          DriverManager.getConnection("jdbc:mysql://localhost/aplan");

          if (conn == null)
          { throw new AplanException ("The data base can't be connected");}
          stmt = conn.createStatement();
          String sql = "SELECT * FROM companiesofgroup where companyname = \'"
                     + cname + "\'";
         rs = stmt.executeQuery(sql);
         while(rs.next()){
              cability = rs.getInt("productability");
          }
         rs.close();
         stmt.close();
     }
```

```
        catch (Exception e) {System.out.print("error while updating a
                             productivity.");}

        finally
        {
            try
            {
                conn.close();
            }
            catch (Exception e)
            {    System.out.println("Exception while releasing DB connection"
                + e.getMessage());
             }
        }

        double tempCoeff = Double.parseDouble(coeff);
        double orderAbility = cability/(tempCoeff*60);

        return orderAbility;
    }


    private HashMap errorMessages = new HashMap();


    public String getErrorType(String errorName) {
        return((String)errorMessages.get(errorName));
    }


    public void addErrorMessage(String errorName, String suggest) {
        errorMessages.put(errorName, suggest);
    }


    public boolean checkHdplanForm() throws AplanException {
        errorMessages.clear();
        boolean error = false;
        if ((ono == null) || (ono.length() == 0)){
            addErrorMessage("ono", "order number is essential.");
            error = true;
        }

        if ((bname == null) || (bname.length() == 0)) {
            addErrorMessage("bname", "branch name is essential.");
            error = true;
        }

        if ((salesid == null) || (salesid.length() == 0)) {
            addErrorMessage("salesid", "sales' ID is essential.");
            error = true;
        }

        return error;

    }

}
```

Figure 66. Class of HD Plan

```
/** File Name:  Orders.java **/

package com.aplan.orders;

import java.sql.*;
import java.util.*;
import java.io.*;
import javax.servlet.http.*;
import java.util.HashMap;
import java.math.BigInteger;
import com.aplan.exceptions.*;
import javax.servlet.RequestDispatcher;


public class Orders {

    private String orderno, oname, buyerid, buyerono;
    private String odescribe, destination;
    private String unit, efdate,  esdate;
    private String  ethddate, efthdate;

    private int utime, totalqty;
    private int efqty = 0;
    private int esqty = 0;
    private int ethdqty = 0;
    private int efthqty = 0;

    public void setOrderno(String aorderno )
    {
        orderno = aorderno;
    }

    public String getOrderno() { return orderno; }

    public void setOname(String aoname )
    {
        oname = aoname;
    }

    public String getOname() { return oname; }


    public void setBuyerid(String abuyerid )
    {
        buyerid = abuyerid;
    }

    public String getBuyerid() { return buyerid; }

    public void setBuyerono(String abuyerono )
    {
        buyerono = abuyerono;
    }

    public String getBuyerono() { return buyerono; }

    public void setUtime(int autime )
    {
        utime = autime;
    }

    public int getUtime() { return utime; }

    public void setOdescribe(String aodescribe)
    {
        odescribe = aodescribe;
    }

    public String getOdescribe() { return odescribe; }
```

```
public void setDestination(String adestination )
{
    destination = adestination;
}

public String getDestination() { return destination; }

public void setTotalqty(int atotalqty)
{
    totalqty = atotalqty;
}

public int getTotalqty() { return totalqty; }

public void setUnit(String aunit)
{
    unit = aunit;
}

public String getUnit() { return unit; }


public void setEfdate(String aefdate)
{
    efdate = aefdate;
}

public String getEfdate() { return efdate; }


public void setEfqty(int aefqty)
{
    efqty = aefqty;
}

public int getEfqty() { return efqty; }

public void setEsdate(String aesdate)
{
    esdate = aesdate;
}

public String getEsdate() { return esdate; }


public void setEsqty(int aesqty)
{
    esqty = aesqty;
}

public int getEsqty() { return esqty; }

public void setEthddate(String aethddate)
{
    ethddate = aethddate;
}

public String getEthddate() { return ethddate; }


public void setEthdqty(int aethdqty)
{
    ethdqty = aethdqty;
}

public int getEthdqty() { return ethdqty; }

public void setEfthdate(String aefthdate)
{
    efthdate = aefthdate;
}
```

```
public String getEfthdate() { return efthdate; }


 public void setEfthqty(int aefthqty)
 {
     efthqty = aefthqty;
 }

 public int getEfthqty() { return efthqty; }


 public Orders() {}

 public Orders(String orderno, String oname, String buyerid, String buyerono,
               int utime, String odescribe,
               String destination, int totalqty, String unit, String efdate,
               int efqty, String esdate, int esqty, String ethddate, int ethdqty,
               String efthdate, int efthqty)
 { this.orderno = orderno;
   this.oname = oname;
   this.buyerid = buyerid;
   this.buyerono = buyerono;
   this.utime = utime;
   this.odescribe = odescribe;
   this.destination = destination;
   this.totalqty = totalqty;
   this.unit = unit;
   this.efdate = efdate;
   this.efqty = efqty;
   this.esdate = esdate;
   this.esqty = esqty;
   this.ethddate = ethddate;
   this.ethdqty = ethdqty;
   this.efthdate = efthdate;
   this.efthqty = efthqty;
 }


 public static Orders findOrder(String orderno) throws AplanException
 {
     Connection conn = null;

    Orders orders = null;

     try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
          conn =
          DriverManager.getConnection("jdbc:mysql://localhost/aplan");

          if (conn == null)
          { throw new AplanException ("The data base can't be connected");}
          String sql = "SELECT * FROM ORDERS WHERE orderno = " +
                    "(?)";
         PreparedStatement statement = conn.prepareStatement(sql);

         statement.setString(1, orderno);
         ResultSet rs = statement.executeQuery();

         if (rs.next()) {
              orders= new Orders();
             orders.setOname(rs.getString("oname"));
             orders.setBuyerid(rs.getString("buyerid"));
             orders.setBuyerono(rs.getString("buyerono"));
             orders.setUtime(rs.getInt("utime"));
             orders.setOdescribe(rs.getString("odescribe"));
             orders.setDestination(rs.getString("destination"));
             orders.setTotalqty(rs.getInt("totalqty"));
             orders.setUnit(rs.getString("unit"));
             orders.setEfdate(rs.getString("efdate"));
             orders.setEfqty(rs.getInt("efqty"));
             orders.setEsdate(rs.getString("esdate"));
             orders.setEsqty(rs.getInt("esqty"));
```

```
            orders.setEthddate(rs.getString("ethddate"));
            orders.setEthdqty(rs.getInt("ethdqty"));
            orders.setEfthdate(rs.getString("efthdate"));
            orders.setEfthqty(rs.getInt("efthqty"));

         }
         else { return null;}
          rs.close();
         statement.close();
    }
    catch (Exception e)
    {
        System.out.println("System error in finding an order. " + e.getMessage());
    }
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
        }
    }
    return orders;
}


public  void insertOrders() throws AplanException{
    Connection conn = null;
    Statement st = null;
    if (findOrder(getOrderno()) != null) {
        throw new AplanException ("duplicate orders");
    }
    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        { throw new AplanException ("The data base can't be connected");}

        String sql = "INSERT INTO orders VALUES " +
                  "(?, ?,  ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,?,?,?,?)";
         PreparedStatement statement = conn.prepareStatement(sql);
        statement.setString(1, orderno);
        statement.setString(2, oname);
         statement.setString(3, buyerid);
         statement.setString(4, buyerono);
         statement.setInt(5, utime);
         statement.setString(6, odescribe);
         statement.setString(7, destination);
         statement.setInt(8, totalqty);
         statement.setString(9, unit);
        statement.setString(10, efdate);
         statement.setInt(11, efqty);
         statement.setString(12, esdate);
         statement.setInt(13, esqty);
         statement.setString(14, ethddate);
         statement.setInt(15, ethdqty);
         statement.setString(16, efthdate);
         statement.setInt(17, efthqty);

         statement.executeQuery();
    }
    catch (Exception e)
    {
        System.out.println("System error in inserting an order. "
                           + e.getMessage());
```

```
    }
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
           + e.getMessage());
        }
    }
}


 public void deleteOrders(String aorder) throws AplanException{
     Connection conn = null;
     Statement st = null;
     ResultSet rs = null;

     try
     {   Class.forName("org.gjt.mm.mysql.Driver").newInstance();
         conn =
         DriverManager.getConnection("jdbc:mysql://localhost/aplan");

         if (conn == null)
         { throw new AplanException ("The data base can't be connected");}

         st = conn.createStatement();
         String sql = "DELETE FROM orders WHERE orderno = \'" + aorder+ "\'";
         rs = st.executeQuery(sql);
     }
    catch (Exception e)
    {
        System.out.println("System error while deleting an order. "
                           + e.getMessage());
    }
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
        }
    }
}


public void updateOrder() throws AplanException
{

    Connection conn = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");
        if (conn == null)
        { throw new AplanException ("The data base can't be connected");}

        String sql = "update orders set oname = ?, buyerid=?,buyerono=?, utime=?,
                     odescribe=?, destination=?, totalqty=?, unit=?, efdate=?,
                     efqty=?, esdate=?, esqty=?, ethddate=?, ethdqty=?,
                     efthdate=?, efthqty=? where  orderno=?";
        PreparedStatement statement = conn.prepareStatement(sql);
        statement.setString(1,oname);
```

```
            statement.setString(2,buyerid);
            statement.setString(3,buyerono);
            statement.setInt(4,utime);
            statement.setString(5,odescribe);
            statement.setString(6,destination);
            statement.setInt(7,totalqty);
            statement.setString(8,unit);
            statement.setString(9,efdate);
            statement.setInt(10,efqty);
            statement.setString(11,esdate);
            statement.setInt(12,esqty);
            statement.setString(13,ethddate);
            statement.setInt(14,ethdqty);
            statement.setString(15,efthdate);
            statement.setInt(16,efthqty);
            statement.setString(17,orderno);
            statement.executeQuery();
    }
        catch (Exception e)
        {
            System.out.println("System error while updating an order. "
                               + e.getMessage());
        }
        finally
        {
            try
            {
                conn.close();
            }
            catch (Exception e)
            {
                System.out.println("Exception while releasing DB connection"
                                   + e.getMessage());
            }
        }
    }


    public static String findUnit(String aorderno)
    {
        Connection conn = null;
        String aunit=null;
        String sql = "";
        Statement st = null;

        try
        {
            Class.forName("org.gjt.mm.mysql.Driver").newInstance();
            conn =
            DriverManager.getConnection("jdbc:mysql://localhost/aplan");

            if (conn == null)
            { throw new AplanException ("The data base can't be connected");}

            st = conn.createStatement();
            sql = "select unit from orders where orderno  = \'" + aorderno+ "\'";
            ResultSet rs = st.executeQuery(sql);

            while (rs.next()) { aunit =rs.getString("unit");}
        }
        catch (Exception e) {System.out.print("error when check the unit in orders ");}
        finally
        {
            try
            {
                conn.close();
            }
            catch (Exception e)
            {
                System.out.println("Exception while releasing DB connection"
                                   + e.getMessage());
```

```
        }
        return aunit;
    }
}

private HashMap errorMessages = new HashMap();


public String getErrorType(String errorName) {
    return((String)errorMessages.get(errorName));
}

public void addErrorMessage(String errorName, String suggest) {
    errorMessages.put(errorName, suggest);
}


public boolean orderFormCorrect()
{
    errorMessages.clear();
    boolean correct = true;

    if ((orderno==null) || (orderno.length() == 0)){
        addErrorMessage("orderno", "order# is essential.");
        correct = false;
    }
    if ((buyerid == null) || (buyerid.length() == 0)) {
        addErrorMessage("buyerid", "buyer ID is essential.");
        correct = false;
    }
     if ((buyerono == null) || (buyerono.length() == 0)) {
        addErrorMessage("buyerono", "buyer's No. is essential.");
        correct = false;
    }

     if (utime == 0)  {
        addErrorMessage("utime", "unit time is essential.");
        correct = false;
    }

    if ((destination== null) || (destination.length() == 0)) {
        addErrorMessage("destination", "destination is essential.");
        correct = false;
    }
     if (totalqty== 0)  {
        addErrorMessage("totalqty", "total quantity is essential.");
        correct = false;
    }

     if ((unit== null) || (unit.length() == 0)) {
        addErrorMessage("unit", "unit is essential.");
        correct = false;
    }
     if ((efdate== null) || (efdate.length() == 0)) {
        addErrorMessage("efdate", "first expected due date is essential.");
        correct = false;
    }
     if (efqty== 0)  {
        addErrorMessage("efqty", "first expected quantity is essential.");
        correct = false;
    }
     return correct;
}
}
```

Figure 67. Class of Orders

```
/** File Name:  Branchplan.java **/
package com.aplan.branchplan;

import java.sql.*;
import java.util.*;
import com.aplan.exceptions.AplanException;
import com.aplan.hdplan.Hdplan;
import com.aplan.orderreport.Orderreport;

public class Branchplan {

    private String bpono, bpcname, bplineno, bpfbegindate, bpfenddate, bpsbegindate;
    private String bpsenddate, bpthdbegindate, bpthdenddate, bpfthbegindate,
    bpfthenddate;
    private int bpfqty, bpsqty, bpthdqty, bpfthqty;

    public void setBpono( String abpono )
    {
        bpono = abpono;
    }

    public String getBpono() { return bpono; }

    public void setBpcname( String abpcname )
    {
        bpcname = abpcname;
    }

    public String getBpcname() { return bpcname; }

    public void setBplineno( String abplineno )
    {
        bplineno = abplineno;
    }

    public String getBplineno() { return bplineno; }

    public void setBpfqty(int abpfqty)
    {
        bpfqty = abpfqty;
    }

    public int getBpfqty() { return bpfqty; }

    public void setBpfbegindate(String abpfbegindate)
    {   bpfbegindate = abpfbegindate;
    }

    public String getBpfbegindate() { return bpfbegindate;}

    public void setBpfenddate(String abpfenddate)
    {   bpfenddate = abpfenddate;
    }

    public String getBpfenddate() { return bpfenddate;}


    public void setBpsqty(int abpsqty)
    {
        bpsqty = abpsqty;
    }

    public int getBpsqty() { return bpsqty; }

    public void setBpsbegindate(String abpsbegindate)
    {   bpsbegindate = abpsbegindate;
    }

    public String getBpsbegindate() { return bpsbegindate;}

    public void setBpsenddate(String abpsenddate)
```

```
{   bpsenddate = abpsenddate;
}

public String getBpsenddate() { return bpsenddate;}


public void setBpthdqty(int abpthdqty)
{
    bpthdqty = abpthdqty;
}

public int getBpthdqty() { return bpthdqty; }

public void setBpthdbegindate(String abpthdbegindate)
{   bpthdbegindate = abpthdbegindate;
}

public String getBpthdbegindate() { return bpthdbegindate;}

public void setBpthdenddate(String abpthdenddate)
{   bpthdenddate = abpthdenddate;
}

public String getBpthdenddate() { return bpthdenddate;}


public void setBpfthqty(int abpfthqty)
{
    bpfthqty = abpfthqty;
}

public int getBpfthqty() {return bpfthqty; }

public void setBpfthbegindate(String abpfthbegindate)
{   bpfthbegindate = abpfthbegindate;
}

public String getBpfthbegindate() { return bpfthbegindate;}

public void setBpfthenddate(String abpfthenddate)
{   bpfthenddate = abpfthenddate;
}

public String getBpfthenddate() { return bpfthenddate;}

public Branchplan() {
}

public Branchplan(String bpono, String bpcname, String bplineno, int bpfqty,
                  String bpfbegindate, String bpfenddate,int bpsqty,
                  String bpsbegindate, String bpsenddate,int bpthdqty,
                  String bpthdbegindate, String bpthdenddate,int bpfthqty,
                  String bpfthbegindate, String bpfthenddate)

{ this.bpono = bpono;
  this.bpcname = bpcname;
  this.bplineno = bplineno;
  this.bpfqty = bpfqty;
  this.bpfbegindate = bpfbegindate;
  this.bpfenddate = bpfenddate;
  this.bpsqty = bpsqty;
  this.bpsbegindate = bpsbegindate;
  this.bpsenddate = bpsenddate;
  this.bpthdqty = bpthdqty;
  this.bpthdbegindate = bpthdbegindate;
  this.bpthdenddate = bpthdenddate;
  this.bpfthqty = bpfthqty;
  this.bpfthbegindate = bpfthbegindate;
  this.bpfthenddate = bpfthenddate;

}
```

```
public static boolean findOrder(String abpono) throws AplanException
{
    Connection conn = null;
    Statement st = null;
    boolean find = true;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
          conn =
          DriverManager.getConnection("jdbc:mysql://localhost/aplan");

          if (conn == null)
          {   throw new AplanException ("The data base can't be connected");}

          st = conn.createStatement();
          String sql = "select * from branchplan where bpono  = \'" + abpono + "\'";
          ResultSet rs = st.executeQuery(sql);
          if (!rs.next())
          {
              find = false;
          }
          rs.close();
          st.close();
    }
    catch (Exception e)
    {
        System.out.println("System error in finding an order in branch plans. "
                           + e.getMessage());
    }
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
             + e.getMessage());
        }
        return find;
    }
}


public static double calLineAbility (String coeff, String unittime, String acname,
                                     String alineno) throws AplanException{
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
    int cability = 0;
    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        {   throw new AplanException ("The data base can't be connected");}
        stmt = conn.createStatement();
        String sql = "SELECT * FROM lineability where cname= \'"+ acname + "\'"
                     + "and lineno= \'"+ alineno + "\'";

        rs = stmt.executeQuery(sql);
        while(rs.next()){
            cability = rs.getInt("lineability");
        }
        rs.close();
        stmt.close();

    }
    catch (Exception e)
    {
        System.out.println("System error in finding the productivity in line. "
```

```
                        + e.getMessage());
    }
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                            + e.getMessage());
        }

      double tempCoeff = Double.parseDouble(coeff);
      double lineAbility = cability/(tempCoeff*60);
      return lineAbility;
    }
}

public void insertBranchPlan() throws AplanException{

    Connection conn = null;
    Statement st = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        {   throw new AplanException ("The data base can't be connected");}

        String sql = "INSERT INTO branchplan VALUES " +
                   "(?, ?,?,?,?,?,?,?,?,?,?,?,?,?, ?)";
         PreparedStatement statement = conn.prepareStatement(sql);
        statement.setString(1, bpono);
         statement.setString(2, bpcname);
         statement.setString(3, bplineno);
        statement.setInt(4,bpfqty);
        statement.setString(5,bpfbegindate);
        statement.setString(6,bpfenddate);
        statement.setInt(7,bpsqty);
        statement.setString(8,bpsbegindate);
        statement.setString(9,bpsenddate);
        statement.setInt(10,bpthdqty);
        statement.setString(11,bpthdbegindate);
        statement.setString(12,bpthdenddate);
        statement.setInt(13,bpfthqty);
        statement.setString(14,bpfthbegindate);
        statement.setString(15,bpfthenddate);
         statement.executeQuery();
    }
    catch (Exception e)
    {
        System.out.println("System error in insertng a branch plan. "
                        + e.getMessage());
    }
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                            + e.getMessage());
        }
    }

}
```

```
public void deleteBp(String abpono) throws AplanException{

    Connection conn = null;
    Statement st = null;
    ResultSet rs = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        {   throw new AplanException ("The data base can't be connected");}
        st = conn.createStatement();
        String sql = "DELETE FROM branchplan WHERE bpono = \'" + abpono + "\'";
        rs = st.executeQuery(sql);
    }
    catch (Exception e)
    {
        System.out.println("System error in deleting a branch plan. "
                           + e.getMessage());
    }
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
        }
    }

}


public void deleteBpItem (String abpono, String abpcname, String abplineno)
                          throws AplanException
{
    Connection conn = null;
    Statement st = null;
    ResultSet rs = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");
        if (conn == null)
        {   throw new AplanException ("The data base can't be connected");}

        st = conn.createStatement();
        String sql = "DELETE FROM branchplan WHERE bpono = \'" + abpono + "\'"
                     + "and  bpcname = \'"  + abpcname + "\'" + "and  bplineno= \'"
                   + abplineno + "\'";
        rs = st.executeQuery(sql);
    }
    catch (Exception e)
    {
        System.out.println("System error in deleting a branch plan item. "
                           + e.getMessage());
    }
    finally
    {
        try
        {
            conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
```

```
                                    + e.getMessage());
            }
        }

    }


    public void updateBranchPlan() throws AplanException
    {
        Connection conn = null;
        try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
            conn =
            DriverManager.getConnection("jdbc:mysql://localhost/aplan");
            if (conn == null)
            {   throw new AplanException ("The data base can't be connected");}


            String sql = "update branchplan set bpfqty = ?, bpfbegindate=?,
                          bpfenddate=?,bpsqty = ?, bpsbegindate=?, bpsenddate=?,
                          bpthdqty = ?, bpthdbegindate=?,bpthdenddate=?,bpfthqty = ?,
                          bpfthbegindate=?,bpfthenddate=? where  bpono=? and bpcname=?
                          and bplineno=?";
            PreparedStatement statement = conn.prepareStatement(sql);
            statement.setInt(1,bpfqty);
            statement.setString(2,bpfbegindate);
            statement.setString(3,bpfenddate);
            statement.setInt(4,bpsqty);
            statement.setString(5,bpsbegindate);
            statement.setString(6,bpsenddate);
            statement.setInt(7,bpthdqty);
            statement.setString(8,bpthdbegindate);
            statement.setString(9,bpthdenddate);
            statement.setInt(10,bpfthqty);
            statement.setString(11,bpfthbegindate);
            statement.setString(12,bpfthenddate);
            statement.setString(13,bpono);
            statement.setString(14,bpcname);
            statement.setString(15,bplineno);
            statement.executeQuery();

        }
        catch (Exception e)
        {
            System.out.println("System error in updating a branch plan. "
                               + e.getMessage());
        }
        finally
        {
            try
            {
                conn.close();
            }
            catch (Exception e)
            {
                System.out.println("Exception while releasing DB connection"
                                   + e.getMessage());
            }
        }
}

    private HashMap errorMessages = new HashMap();

    public String getErrorType(String errorName) {
        return((String)errorMessages.get(errorName));
    }


    public void addErrorMessage(String errorName, String suggest) {
        errorMessages.put(errorName, suggest);
    }
```

```
public boolean checkBranchPlanForm() throws AplanException{
    errorMessages.clear();
    boolean error = false;
    if ((bpono == null) || (bpono.length() == 0)){
        addErrorMessage("bpono", "order number is essential.");
        error = true;
    }

    if ((bpcname == null) || (bpcname.length() == 0)) {
        addErrorMessage("bpcname", "branch name is essential.");
        error = true;
    }


    if ((bplineno == null) || (bplineno.length() == 0)) {
        addErrorMessage("bplineno", "line No. is essential.");
        error = true;
    }

    if((bpfqty!=0)&&(bpfbegindate==null))
    {   addErrorMessage("bpfbegindate", "the first begin date is essential.");
        error = true;
    }
    if((bpfqty!=0)&&(bpfenddate==null))
    {   addErrorMessage("bpfenddate", "the first end date is essential.");
        error = true;
    }

    if((bpsqty!=0)&&(bpsbegindate==null))
    {   addErrorMessage("bpsbegindate", "the second begin date is essential.");
        error = true;
    }
    if((bpsqty!=0)&&(bpsenddate==null))
    {   addErrorMessage("bpsenddate", "the second end date is essential.");
        error = true;
    }

    return error;
}


public boolean ifUpdateBpError() throws AplanException{
    errorMessages.clear();
    boolean error = false;
    if ((bpono== null) || (bpono.length() == 0)){
        addErrorMessage("bpono", "Order Number is essential.");
        error = true;
    }

    if ((bpcname == null) || (bpcname.length() == 0)) {
        addErrorMessage("bpcname", "branch name is essential.");
        error = true;
    }


    if ((bplineno== null) || (bplineno.length() == 0)) {
        addErrorMessage("bplineno", "line number is essential.");
        error = true;
    }

    if (bpfqty !=0) {

        if((bpfbegindate == null)||(bpfbegindate.length()==0))
        {   addErrorMessage("bpfbegindate", "first begin date is essential.");
            error = true;
        }
        if((bpfenddate == null)||(bpfenddate.length()==0))
        {   addErrorMessage("bpfenddate", "first end date is essential.");
            error = true;
        }
```

```
        }
      return error;
    }
}
```

Figure 68. Class of Branch Plan

```
/** File Name Productionreport.java **/

package com.aplan.productionreport;

import java.sql.*;
import java.util.HashMap;
import com.aplan.exceptions.AplanException;
import com.aplan.branchplan.Branchplan;

public class Productionreport {

    private String prono,prcname, prlineno, prdate, prremark;

    private int prinput,proutput;
    private final double errorRange = 0.01;


    public void setProno( String aprono )
    {
       prono = aprono;
    }

    public String getProno() { return prono; }

    public String getPrcname() { return prcname;}


    public void setPrcname( String aprcname)
    {
        prcname = aprcname;
    }


    public String getPrlineno() {return prlineno;}

    public void setPrlineno (String aprlineno)
    {    prlineno = aprlineno;
    }


    public String getPrdate() { return prdate; }

    public void setPrdate(String aprdate)
    {    prdate = aprdate;
    }

    public String getPrremark() { return prremark; }

    public void setPrremark(String aprremark)
    {    prremark = aprremark;
    }


    public int getPrinput() { return prinput; }

    public void setPrinput(int aprinput)
    {    prinput = aprinput;
    }

    public int getProutput() { return proutput; }

    public void setProutput(int aproutput)
    {    proutput = aproutput;
    }



    public Productionreport() {}
```

```
public Productionreport(String prono, String prcname, String prlineno,
                        int prinput, int proutput, String prdate,String prremark)


{ this.prono = prono;
  this.prcname = prcname;
  this.prlineno = prlineno;
  this.prinput = prinput;
  this.proutput = proutput;
  this.prdate = prdate;
  this.prremark = prremark;

}


public static boolean find_order(String aprono) throws AplanException
{
    Connection conn = null;
    Statement st = null;
    boolean find = true;

    try {
        Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        {   throw new AplanException ("The data base can't be connected");}
        st = conn.createStatement();
        String sql = "select * from productionreport where prono = \'" + aprono
                     + "\'";
        ResultSet rs = st.executeQuery(sql);
        if (!rs.next()) {
           find = false;
        }
        rs.close();
        st.close();
    }
    catch (Exception e)
    {
        System.out.println("System error in finding a production report with a
                           specific order number. " + e.getMessage());
    }
    finally
    {
        try
        {    conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
        }
        return find;
    }
}


public static int TtlOutputInLine(String aprono, String aprcname,
                                  String aprlineno){

    Connection conn = null;
    int total = 0;
    String sql = "";
    Statement st = null;

    try
    {   Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");
```

```
        if (conn == null)
        {
            throw new AplanException ("The data base can't be connected");}
            st = conn.createStatement();
            sql = "select sum(proutput) AS totalqty from productionreport where
                  prono  = \'" + aprono + "\'" +" and prcname  = \'" + aprcname
                  + "\'" +" and prlineno  = \'" + aprlineno + "\'" + "group by
                  prono, prcname, prlineno";
            ResultSet rs = st.executeQuery(sql);

           while (rs.next()) { total =rs.getInt("totalqty");}

    }
    catch (Exception e)
    {
        System.out.println("System error in finding total output. "
                           + e.getMessage());
    }
    finally
    {
        try
        {    conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
        }
        return total;
    }
}


public static int TtlInputInLine(String aprono, String aprcname, String aprlineno){

    Connection conn = null;
    int total = 0;
    String sql = "";
    Statement st = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        {   throw new AplanException ("The data base can't be connected");}
        st = conn.createStatement();
        sql = "select sum(prinput) AS totalqty from productionreport where prono  =
              \'" + aprono + "\'" +" and prcname  = \'" + aprcname + "\'" +" and
              prlineno  = \'" + aprlineno + "\'" + "group by prono, prcname,
              prlineno";
        ResultSet rs = st.executeQuery(sql);

       while (rs.next()) { total =rs.getInt("totalqty");}

    }
    catch (Exception e)
    {
        System.out.println("System error in finding total input. "
                           + e.getMessage());
    }
    finally
    {
        try
        {    conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
```

```
        }
        return total;
    }
}


public void insertProductionReport() throws AplanException{

    Connection conn = null;
    Statement st = null;


    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        { throw new AplanException ("The data base can't be connected");}

        String sql = "INSERT INTO productionreport VALUES " +
                    "(?, ?,?,?,?,?,?)";
        PreparedStatement statement = conn.prepareStatement(sql);
        statement.setString(1, prono);
        statement.setString(2, prcname);
        statement.setString(3, prlineno);
        statement.setInt(4, prinput);
        statement.setInt(5, proutput);
        statement.setString(6, prdate);
        statement.setString(7, prremark);
        statement.executeQuery();
}
    catch (Exception e)
    {
        System.out.println("System error in inserting a production report. "
                           + e.getMessage());
    }
    finally
    {
        try
        {    conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
        }

    }
}


public void deleteProductionReport(String aprono) throws AplanException{
    Connection conn = null;
    Statement st = null;
    ResultSet rs = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        { throw new AplanException ("The data base can't be connected");}
        st = conn.createStatement();
        String sql = "DELETE FROM productionreport WHERE prono = \'" + aprono
                     + "\'";
        rs = st.executeQuery(sql);
    }
    catch (Exception e)
    {
```

```
        System.out.println("System error in deleting a production report. "
                           + e.getMessage());
    }
    finally
    {
        try
        {    conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
        }

    }
}


public void deleteProductionReportItem(String aprono, String aprcname,
            String aprlineno, String aprdate) throws AplanException{

    String sql = null;
    Connection conn = null;
    Statement st = null;
    ResultSet rs = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        { throw new AplanException ("The data base can't be connected");}

        st = conn.createStatement();
        sql = "DELETE FROM productionreport WHERE prono = \'" + aprono + "\'"
              + "and  prcname = \'" + aprcname + "\'" + "and prlineno = \'" +
              aprlineno + "\'" + "and prdate = \'" + aprdate + "\'";
        rs = st.executeQuery(sql);
}
    catch (Exception e)
    {
        System.out.println("System error in deleting a production report item. "
                           + e.getMessage());
    }
    finally
    {
        try
        {    conn.close();
        }
        catch (Exception e)
        {
            System.out.println("Exception while releasing DB connection"
                               + e.getMessage());
        }

    }
}


public void updateInput() throws AplanException{

    Connection conn = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
```

```
        { throw new AplanException ("The data base can't be connected");}

        String sql = "update productionreport set prinput = ? where  prono=?";
        PreparedStatement statement = conn.prepareStatement(sql);
       statement.setInt(1,prinput);
        statement.setString(2,prono);

        statement.executeQuery();
    }
    catch (Exception e) {System.out.print("error while update the production
                         report input item");}
   finally
   {
       try
       {    conn.close();
       }
       catch (Exception e)
       {
           System.out.println("Exception while releasing DB connection"
           + e.getMessage());
       }
   }
}


public void updateOutput() throws AplanException{

    Connection conn = null;

    try { Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        conn =
        DriverManager.getConnection("jdbc:mysql://localhost/aplan");

        if (conn == null)
        { throw new AplanException ("The data base can't be connected");}

        String sql = "update productionreport set proutput = ? where  prono=?";
        PreparedStatement statement = conn.prepareStatement(sql);

        statement.setInt(1,proutput);
        statement.setString(2,prono);

        statement.executeQuery();
    }
    catch (Exception e) {System.out.print("error while update the production
                        report output item");}
   finally
   {
       try
       {    conn.close();
       }
       catch (Exception e)
       {
           System.out.println("Exception while releasing DB connection"
                              + e.getMessage());
       }

   }
}


private HashMap errorMessages = new HashMap();

public String getErrorType(String errorName) {
   return((String)errorMessages.get(errorName));
}


public void addErrorMessage(String errorName, String suggest) {
   errorMessages.put(errorName, suggest); }
```

REFERENCES

[1] Yahoo Mobile Site. Web-address: http://sg.mobile.yahoo.com/www/whatis_java.php

[2] Sun's Web Site. Web-address: http://java.sun.com/products/jdbc/.

[3] Jose Annunziato and Stephanie F. Kaminaris, Sams Teach Yourself JavaServer Pages in 24 Hours, P6.

[4] Jose Annunziato and Stephanie F. Kaminaris, Sams Teach Yourself JavaServer Pages in 24 Hours, P18.

[5] MySQL Web Site. Web-address: http://www.mysql.org