



ERP System for Sanergy(pvt) Ltd

Chamarawarna Samaraweera

R0521558

Name of the supervisor

H.V. Ajith

2009/ 2010



**This dissertation is submitted in partial fulfilment of the requirement of the Degree
of Bachelor of Information Technology of the University of Colombo School of
Computing**

ABSTRACT

There is a potent need for the establishment of an ERP system as a solution for problems such as the manual maintenance of purchase order processes and inventories. The increasing rate of inaccuracies related to report writing, employee attendance records and payrolls have further given significance for the installation of this ERP system.

Throughout this project object oriented research methodologies were utilized. Hence lower maintenance cost, better quality and higher productivity can be obtained. The system was designed and modeled using Unified Modeling Language (UML) and it is developed using Java Programming language.. The tools used for UML Modeling, data base designing, report designing and user interface designing include Netbeans, Power Architect and pg Admin and IReport for Netbeans. JTables, JComboboxes and Jlists are used to diminish errors during data entry in user interfaces with the system.

The implementation of this ERP system brought many positive outcomes which include improving efficiency, increasing productivity and reducing the time needed to manage inventories. A prominent decrease in errors and inaccuracies related to stock issues and misinterpretations of data was also a commendable result. This ERP system fosters fast and quality reporting which emphasizes on analysis of previous information of presentation of high quality.

However the system has few aspects which have potential to be further developed. The ultimate objective of the project was to develop and produce an ERP system which proved to be successful subsequently to the formulation of requirements.

ACKNOWLEDGEMENT

This project is a result of my effort with the continuous support of our Supervisor, H.V. Ajith who guided my way throughout the design of this System as well as the project.

Also my heartfelt gratitude towards University of Colombo School of computing, Bachelor of Information Technology (External) for this great opportunity extended for myself to seek, learn and apply.

I take this opportunity to thank my family who lent me their hands giving me freedom and peace of mind to a better working environment setting up priority on my effort to fulfill my effort of system designing and documentation. I'm grateful for my family and for my friends for their various efforts of cooperation and interest taken over on my studies.

Last but not least, I would like to thank the lecturers for cooperating with me and helping and being kind enough to devote their time for me.

CONTENTS

ABSTRACT	I
ACKNOWLEDGEMENT	III
LIST OF FIGURES	VII
LIST OF TABLES	VIII
LIST OF ACRONYMS	IX
DECLARATION	X
CHAPTER 1 INTRODUCTION	1
1.1 MOTIVATION	1
1.1.1 PROBLEM DOMAIN AND NEED OF THE PROJECT --	1
1.2 OBJECTIVES AND SCOPE OF THE PROJECT	2
1.3 STRUCTURE OF THE DISSERTATION	3
CHAPTER 2 ANALYSIS	4
2.1 INTRODUCTION	4
2.1.1 CURRENT SYSTEM'S ANALYSIS	4
2.2 OTHER EQUIVALENT SYSTEMS STUDIED	5
2.3 REQUIREMENTS	6
2.3.1 USER REQUIREMENTS	6
2.3.2 SYSTEM REQUIREMENTS	6
2.4 USE CASE DIAGRAM	7
CHAPTER 3 DESIGN	11
3.1 INTRODUCTION	11
3.2 ALTERNATE SOLUTIONS	12
3.3 SELECTED SOLUTIONS	12
3.4 USER INTERFACE DESIGN	12
3.4.1 STRUCTURE OF THE USER INTERFACES	13
3.4.2 SAMPLE INTERFACES	14
3.5 DIAGRAMS	15
CHAPTER 4 IMPLEMENTATION	21
4.1 INTRODUCTION	21

4.2 IMPLEMENTATION ENVIRONMENT -----	21
4.2.1 DEVELOPMENT TOOLS -----	22
4.3 CODE AND MODULE STRUCTURE -----	24
4.3.1 USER INTERFACE CLASSES -----	24
4.3.2 CONTROLLER CLASSES -----	26
4.3.3 ENTITY CLASSES -----	28
CHAPTER 5 EVALUATION -----	30
5.1 TESTING -----	30
5.1.1 TEST PLANS -----	30
5.2 SAMPLE ACCEPTANCE TEST QUESTIONNAIRES	
WITH ANSWERS -----	33
CHAPTER 6 CONCLUSION -----	35
6.1 ASSESSMENT OF PROJECT -----	35
6.1.1 AGAINST NON-FUNCTIONAL REQUIREMENTS ---	35
6.1.2 AGAINST FUNCTIONAL REQUIREMENTS -----	35
6.2 FUTURE WORKS -----	36
6.3 KNOWLEDGE AND EXPERIENCE -----	36
6.4 SUMMERY -----	36
REFERENCES -----	37
APPENDIX A -----	38
A.1. INSTALLATION -----	38
A.2. COMPILATION AND EXECUTION -----	38
APPENDIX B -----	39
B.1. NOTES -----	39
B.2. GETTING STARTED WITH LOGIN -----	39
B.3. MAIN APPLICATION WINDOW -----	40
B.4. USER MANAGEMENT.	
(ONLY FOR ADMINISTRATOR ROLE) -----	41
B.5. MASTER DATE MANAGEMENT	
(ONLY FOR ADMINISTRATOR ROLE & OPERATOR ROLE) --	42

B.6. TRANSACTION MANAGEMENT	
(ONLY FOR ADMINISTRATOR ROLE & OPERATOR ROLE) --	46
B.7. REPORTS -----	49
B.8. PAYROLL MANAGEMENT -----	50
APPENDIX C -----	51
MAJOR CODES -----	51
C.1. SAMPLE ENTITY CLASSES CODES -----	51
C.2. SAMPLE CONTROLLER CLASSES CODES -----	67
C.3. DATABASE CREATION SCRIPT -----	94
APPENDIX D -----	106
CLIENT CERTIFICATE -----	106

List of Figures

Figure 2.1 Use case diagrams	7
Figure 4.1 User Screen	14
Figure 4.2 Manage Sales Order Screen	15
Figure 4.3 Class Roles	15
Figure 4.4 Class Suppliers	16
Figure 4.5 Class Customers	16
Figure 4.6 Class Products	17
Figure 4.7 Class SalesOrders	18
Figure 4.8 Class Users	19
Figure 4.9 Database Diagram	20
Figure B1 Login Screen	39
Figure B2 Main Application Window	40
Figure B3 User Management Screen	41
Figure B4 Product Management Screen	42
Figure B5 Product Bom Screen	43
Figure B6 Product Supplier Screen	43
Figure B7 Employee, Suppliers, Customers Screens	45
Figure B8 Transaction Management Screens	48
Figure B9 Report Screens	49
Figure B10 Payroll Management Screens	50

List of Tables

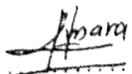
Table 2.1 Hardware Configurations	6
Table 2.2 Software Configurations	7
Table 2.3 Login Use Case View	8
Table 2.4 Maintenance Use Case View	8
Table 2.5 Product Input Use Case View	9
Table 2.6 Customer Input Use Case View	9
Table 2.7 Supplier Input Use Case View	10
Table 2.8 Sales Order Use Case View	10
Table 4.1 Hardware Configurations	21
Table 4.2 Software Configurations	22
Table 5.1 Login Test Case	30
Table 5.2 Product Test Case	31
Table 5.4 Customer Test Case	32
Table 5.5 Sales Order Test Case	32
Table 5.6 Manage Purchase Order Test Case	33
Table 5.7 Purchase Order Test Plans	33
Table 5.8 Actual Answers Summary	34

List of Acronyms

ERP	- Enterprise Resource Planning
KPI	- key performance indicators
EPF	- Employees Provident Fund
ETF	- Employees Trust Fund
BOM	- Bill Of Materials
API	- Application Programming Interface
OOA	- Object-Oriented analysis
OOD	- Object-oriented design
OOAD	- Object-oriented analysis and design
RUP	- Rational Unified Process
UML	- Unified Modeling Language
OMT	- Object Modeling Technique
OOSE	- Object-Oriented Software Engineering
UML	- Unified Modeling Language
GRN	- Goods Receive Notes
ORDBMS	- Object-relational database management system

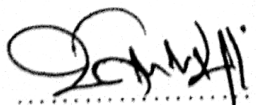
DECLARATION

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a Degree or Diploma in any University and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and for inter-library loans, and for the title and summary to be made available to outside organizations.


.....
C. Samaraweera
(Name of Candidate)

Date: 04/08/2010

Countersigned by:


.....
Ms. H.V. Ajith
(Name of Supervisor)

Date: 05/08/2010

SANERGY PRIVATE LTD
11/18A, School Avenue,
Mahindaramma Road,
Etal-Kotte.

INTRODUCTION

Chapter 1

1.1 MOTIVATION

Sanergy (pvt) Ltd is a company which produces many types of electrical equipments to the local market. There are around 50 workers in this company and it is now well established in Srilanka. One of the main product/service outcomes of this company is traffic lights. They used to buy basic ingredients such as transistors, resisters, IC's, bulbs from over seas (mostly from china) and produce a final outcome (traffic lights, road lamps, act.) locally.

The company plans to increase the number of customers and expand their services/products. Their workload will be higher in near future, thus they wanted to improve the efficiency of inventory control and purchase order processing. Currently, they have a manual system, which caused lot of ambiguities and some operations malfunctions. Some regular problems are listed below.

1.1.1 PROBLEM DOMAIN AND NEED OF THE PROJECT

Currently purchase order processes and the inventory are maintains manually so it result in many problems like out of stock issues and miss interprets of data. The report writing is time consuming and low in quality. Writing reports involves analysis of previous information and quality in presentation.

It takes considerable time period to complete the month end payroll and day to day attendance marking of the employees.

So, considering above disadvantages and ambiguities the requirement of this ERP system is obvious.

1.2 OBJECTIVES AND SCOPE OF THE PROJECT

There are few main objectives of the system which are listed below.

1. To build purchase order/Sales order management system for easy use.
2. Manage inventory times and products effectively.
3. Build proper payroll system.
4. Provide reporting facility for the above modules.

And also it is supposed to achieve following goals as well.

Providing an Effective Report Generation Process

Managers need various reports including stock lever, purchase order reports and etc. All the reports have to be in good quality in presentation and faster generation time.

Accuracy

Stock controlling, purchasing and payroll modules must be accurate as they are critical function of the company.

Quicker the Processing Time

System must react faster for the users.

User Friendliness and Easy to use

The system has to be providing with user friendly interfaces and the operations are to be easy to handle.

Security

Provide different access privileges for different parts of the overall system.

1.3 STRUCTURE OF THE DISSERTATION

The main divisions of this dissertation are pointed on the work done during the many ways of the scheme.

Each chapter will have the necessary details to help to understand the project with the help of suitable figures, graphs and reports. This section is helping to give the starting plan and the system and the observation of the project to the reader.

Second chapter will give the knowledge to follow with the things written. This will explain work in the project and will give the information about other similar systems and also will explain the literature review of this project.

The third chapter is on the necessary things wanted, details and design. It gives in brief of all the things to find the requirements and inform the systems necessary in detail. Then the type of work and the tools for the work arranged will be shown. All kind of methods, beginnings and what is expected at the beginning is given in detail. All named diagrams are given and explained. The most important design what expected will be discussed. Organized design will be in this chapter.

Fourth chapter is about the implementation. This will give main implementation details with suitable screen shots.

Fifth chapter is about on testing. It shows how to start testing and methods used in the project with all the major test cases and test results.

The last chapter gives you the conclusion of the project. There it will be discussed and estimated all the results of the project and will show failures and problems came across in the project. And will give suggestions for the future.

In addition, shown materials are available in the reference section. In appendices, System Documentation, User Documentation, Code Listing are available.

2.1 INTRODUCTION

During the analysis phase many requirement gatherings were carried through observation of the basic business process, interviews with users and questionnaires. At the beginning most of the documentations were gathered to identify the flow of documents. Then it was decided that process improvement was necessary.

2.1.1 CURRENT SYSTEM'S ANALYSIS

Currently purchase order processes and the inventory are maintained manually. So they have to out come many problems like out of stock issues and miss interprets of data. The report writing was time consuming and low in quality. They already were using some computer applications (i.e. Excel) to process purchase orders, payrolls, etc.

When they want to construct a product from ingredients they have to go through the current stock and order required components. Therefore it deals with lots of physical work and paper work which may cause human errors. This is same with the current payroll system too when they are calculating salaries each month it takes huge amount of time and paper work. Also the inquiring of heuristic information is difficult.

2.2 OTHER EQUIVALENT SYSTEMS STUDIED

Before we decide to design the system we used to study Openbravo ERP system [WWW10] as our primary source for ERP systems. It is completely developed in J2EE web container and can be configured to may database management systems.

Sales Order Management

Sales Management entitles all management of activities related to Customer Sales, such as Sales Orders, Pricing, Shipping, Invoicing and Reporting. It is possible to link documents such as order documents, delivery notes and invoice, in any order required. If considered unnecessary, they can be disregarded. The capacities of the EDI (Electronic Data Interchange) and the integration with PDAs (Personal Digital Assistant) provide access beyond the physical limits of a company. [WWW1]

Purchase Order Management

A ‘Purchase Order’ is a document that specifies products and/or services ordered from a specific vendor, as well as the price, terms and conditions. Purchase Orders are usually delivered according to contractual agreements with a supplier, specifying payment terms, delivery dates, item identification, quantities, and freight terms and all other obligations and conditions. A ‘Purchase Order’ can be entered manually or easily created by copying an existing Purchase Order. An automated method to create Purchase Orders is by running a program to convert the proposals from a ‘Purchasing Plan’, generated by MRP, to ‘Purchase Orders’. [WWW2]

Warehouse Management

Fundamentally, warehouse management is the physical management of inventory. In some cases, inventory may be stored for an extended time. In other warehouses that implement just in time practices inventory experiences rapid turn over and the warehouse functions as a distribution center. A warehouse can contain raw materials, work-in-process inventory, finished goods, supplies, and possibly repair parts. As with other elements in the any distribution system, the objective of the warehouse is to minimize costs and maximize customer service [WWW3]

2.3 REQUIREMENTS

Project requirements include user requirements and system requirements.

2.3.1 USER REQUIREMENTS

Sanergy (pvt) Ltd is currently handling their main functions manually but the efficiency of the current manual system is very poor. Therefore it was decided to develop an ERP system for the company which handles Purchase orders, Sales orders, Product inventory and payroll. These operations are day to day operations of the Sanergy (pvt) Ltd. Other than that the system should be very efficient and reliable which caused to satisfy their needs. For that it was introduces this ERP system for the Sanergy (pvt) Ltd.

2.3.2 SYSTEM REQUIREMENTS

For the usage of ERP system, the following resources are required.

Hardware Requirements

A standard PC with following configuration (Table 2.1) will be necessary for system implementation.

Processor Clock Speed	Intel Pentium IV(Recommended) 2.6GHz (minimum)
Memory	2GB
HDD	40GB
VGA Memory	32MB
Monitor	Minimum Resolution 800 * 600
Printer	Dot matrix printer A4 and Pos Printer for salary slip printing.
Network Card	Network interface card
Modem	Any Internet connectivity modem

Table 2.1 Hardware Configurations

Software Requirements

Following software (Table 2.2) necessary for the software implementation

Operating System	Windows (XP or above) Or Unix
Programming Language Requirement	JRE 1.6
Database Requirements	PostgreSQL 8.3 Database server

Table 2.2 Software Configurations

2.4 USE CASE DIAGRAM

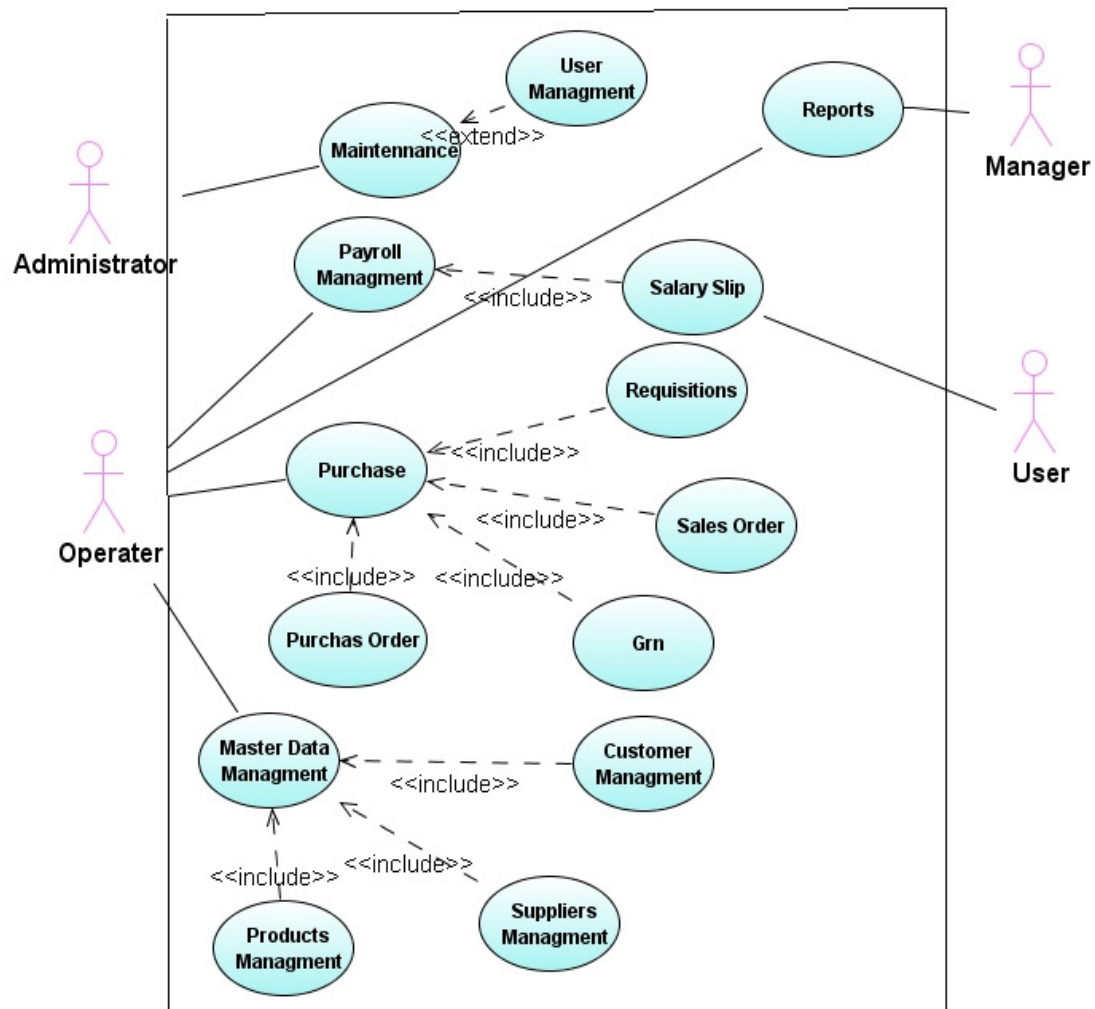


Figure 2.1 Use case diagrams

Use case views have described using the Use Case diagram (Figure 2.1) illustrates above. Main actors to the system are Administrator, Operator, Manager and User. Moreover Login (Table 2.3), Maintenance (Table 2.4), Product Input (Table 2.5), Customer Input (Table 2.6), Supplier Input (Table 2.7) and Sales Order (Table 2.8) Use cases are described below.

<i>Use Case Name</i>	<i>Login</i>
<i>Description</i>	<i>This use case describes the login procedure of the use</i>
<i>Pre-condition</i>	<i>Before accessing the system user has to register and get the user ID and password from Administrator/Manager</i>
<i>Main flow of events</i>	<ol style="list-style-type: none"> 1. This use case starts when the application starts 2. User can access the system after a successful login.
<i>Alternative flow</i>	<ol style="list-style-type: none"> 1. If the password or User ID is invaded then error message will be displayed.
<i>Post-Condition</i>	<i>System Access</i>

Table 2.3 Login Use Case View

<i>Use Case Name</i>	<i>Maintenance</i>
<i>Description</i>	<i>Maintain the User Administration</i>
<i>Pre-condition</i>	<i>Administrator level user should be properly logged in.</i>
<i>Main flow of events</i>	<ol style="list-style-type: none"> 1. Start when Administrator level user logged in. 2. Select Add, Update or View as needed.
<i>Alternative flow</i>	<ol style="list-style-type: none"> 1. If the password or User ID is invaded then error message will be displayed. 2. Entered Details not sufficient.
<i>Post-Condition</i>	<i>Add Modify or View Users in the system.</i>

Table 2.4 Maintenance Use Case View

<i>Use Case Name</i>	<i>Product Input</i>
<i>Description</i>	<i>Input Product Details in to the System</i>
<i>Pre-condition</i>	<i>User level should be an Operator or an Administrator.</i>
<i>Main flow of events</i>	<i>1. Start when User selects Products.</i> <i>2. All the necessary Product details must be entered.</i> <i>3. Select Add, Update or View as needed.</i>
<i>Alternative flow</i>	<i>1. Invalid Product code.</i> <i>2. Entered Information not sufficient.</i>
<i>Post-Condition</i>	<i>Do necessary Product Details changes in the System.</i>

Table 2.5 Product Input Use Case View

<i>Use Case Name</i>	<i>Customer Input</i>
<i>Description</i>	<i>Input Customer Details in to the System</i>
<i>Pre-condition</i>	<i>User level should be an Operator or an Administrator.</i>
<i>Main flow of events</i>	<i>1. Start when User selects Customers.</i> <i>2. All the necessary Customer details must be entered.</i> <i>3. Select Add, Update or View as needed.</i>
<i>Alternative flow</i>	<i>1. Invalid Customer code.</i> <i>2. Entered Information not sufficient.</i>
<i>Post-Condition</i>	<i>Do necessary Customer Details changes in the System.</i>

Table 2.6 Customer Input Use Case View

<i>Use Case Name</i>	<i>Supplier Input</i>
<i>Description</i>	<i>Input Supplier Details in to the System</i>
<i>Pre-condition</i>	<i>User level should be an Operator or an Administrator.</i>
<i>Main flow of events</i>	<i>1. Start when User selects Suppliers.</i> <i>2. All the necessary Supplier details must be entered.</i> <i>3. Select Add, Update or View as needed.</i>
<i>Alternative flow</i>	<i>1. Invalid Supplier code.</i> <i>2. Entered Information not sufficient.</i>
<i>Post-Condition</i>	<i>Do necessary Supplier Details changes in the System.</i>

Table 2.7 Supplier Input Use Case View

<i>Use Case Name</i>	<i>Sales Order</i>
<i>Description</i>	<i>Competes a Sales Order.</i>
<i>Pre-condition</i>	<i>User level should be an Operator or an Administrator.</i>
<i>Main flow of events</i>	<i>1. Start when User Select Sales Order.</i> <i>2. All the necessary details of the Sales Orders must be entered.</i> <i>3. Save the details and generates Sales Order Reference Number.</i> <i>4. Select Add, Update or View as needed.</i>
<i>Alternative flow</i>	<i>1. Invalid Sales Order Date.</i> <i>2. Entered Information not sufficient</i>
<i>Post-Condition</i>	<i>Add, Modify or View Sales Orders in the system</i>

Table 2.8 Sales Order Use Case View

3.1 INTRODUCTION

During the design stage, some of the recursive requirement gathering were been happened. Some designed models had to update and correct during the design. Most of the interviews were carried out with company managers on the designed diagrams. There Regular changes to the design model and model have met requirements correct at the end.

Methodologies

Object oriented design methodologies were used rapidly in this project. So, lower maintenance cost, better quality and higher productivity can be achieved.

Unified Modeling Language (UML) was used to design and model the system. The Unified Modeling Language (UML) is a standard modeling language for software a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system the easy. Here the most important part was to develop the model, because a model is the basic foundation to the software system.

Tools

UML Modeling - Netbeans

Database Design – Power Architect and pgAdmin

Report Design – I Report for Netbeans

User Interface Design - Netbeans

3.2 ALTERNATE SOLUTIONS

Before we design the system we consider some alternative solutions for our requiems and able to find few alternate solutions. We can develop this system using Microsoft technologies like VB.net with SQL server. Also we clearly noticed that this application can be written as php web application with mysql database server. Both above solutions are refused because first one is not platform independent and second one because of company currently not planed to go online business. So keeps things simple and cost effective way we decided to build the solution using java with PostgreSQL database server.

3.3 SELECTED SOLUTION

It was decides to develop a ERP system with following features

1. Sales Management
2. Purchase Management
3. Stock Control
4. Payroll Management

Although some existing ERP solutions are matching with our requirements they are too much complex and need heavy resources. The solution we decided was built in a standalone java program which is accessible via company network.

To keep things clearly separated we decided to use M.V.C (Model, View and Controller) design pattern. To keep things perfectly matched we decided to use PostgreSQL as our database server and all selected technologies and tools are open source, up to dated and freely available.

Next section will go through all those Model, View and Controller Designs.

3.4 USER INTERFACE DESIGN

Full User interface Illustrations are enclosed and described in the Appendix C. i.e, “User documentation” and “Reports”. All the reports could be produced as hard copies by printing. All the interfaces are user friendly and provide useful messages during the User interaction. User can select the focus of a control by using the tab key or mouse clicking.

Tab orders are designed to minimize the use of the mouse and convenience with user data entry. Most of the validations are happening control events itself.

Most of the time Users are provided with a set of available data that could be selected to their choice and necessity .Therefore mostly JTables JComboboxes and JLists are used in the system to minimize the errors during data entry.

All the Users are divided in to four levels (i.e. Administrator, Operator, Manager and User). Administrator level users have access to all interfaces, and have access to the User management interfaces, where you can add new users, change user details such as user level, password, etc.

Operator Level user can access all the other interfaces other than the administration interface.

Manager Level user can access only reporting interfaces.

Basic User Level Users can only print their salary slips.

3.4.1 STRUCTURE OF THE USER INTERFACES

1. Transactions

- 1.1 Manage Sales Orders
- 1.2 Manage Purchase Orders
- 1.3 Manage GRN
- 1.4 Requisition

2. Master Data Management

- 2.1 Products
- 2.2 Employee
- 2.3 Suppliers
- 2.4 Customers

3. Maintenance

- 3.1 User

4. Reports

4.1 Sales Order Report

4.2 Purchase Order Report

4.3 GRN Report

4.4 Out Of Stock Report

4.5 Employee Attendance Report

5. Payroll

5.1 Manage Payroll

5.2 My Salary Slip

5.3 Mark Attendance

6. Help

6.1 User Manual

6.2 About

7. Exit

7.1 Logout Current Users

7.2 Exit the Application

3.4.2 SAMPLE INTERFACES

Only User Screen (Figure 4.1) and Manage Sales Order Screen (Figure 4.2) are shown here to reduce the space required

UserID	User Name	Role
1	Nihal	Administrator

Person: 1-Nihal New Password: Role: Administrator Done

Done

Figure 4.1 User Screen

Figure 4.2 Manage Sales Order Screen

3.5 DIAGRAMS

Class Model

Only Roles (Figure 4.3), Suppliers (Figure 4.4), Customers (Figure 4.5), Products (Figure 4.6), SalesOrders (Figure 4.7), and Users (Figure 4.8) classes are shown here to reduce the space required.

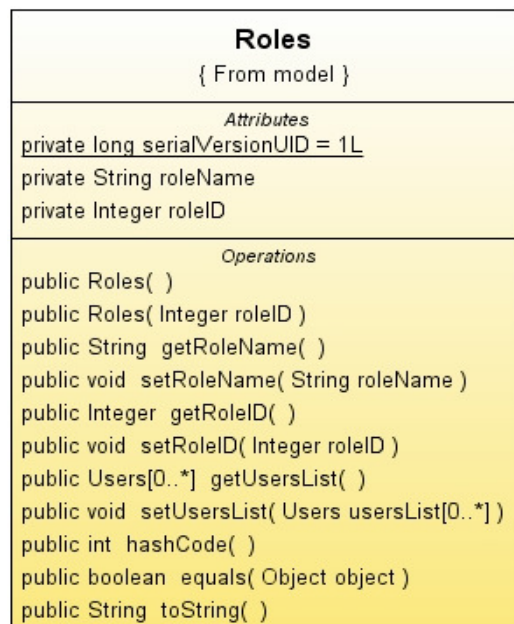


Figure 4.3 Class Roles

Suppliers { From model }
<i>Attributes</i>
<pre> private long serialVersionUID = 1L private String supplierName private Integer supplierID </pre>
<i>Operations</i>
<pre> public Suppliers() public Suppliers(Integer supplierID) public Suppliers(Integer supplierID, String supplierName) public String getSupplierName() public void setSupplierName(String supplierName) public Integer getSupplierID() public void setSupplierID(Integer supplierID) public ProductSuppliers[0..*] getProductSuppliersList() public void setProductSuppliersList(ProductSuppliers productSuppliersList[0..*]) public Users getUsers() public void setUsers(Users users) public Contacts getContacts() public void setContacts(Contacts contacts) public GRN[0..*] getGRNList() public void setGRNList(GRN gRNList[0..*]) public PurchasOrders[0..*] getPurchasOrdersList() public void setPurchasOrdersList(PurchasOrders purchasOrdersList[0..*]) public int hashCode() public boolean equals(Object object) public String toString() </pre>

Figure 4.4 Class Suppliers

Customer { From model }
<i>Attributes</i>
<pre> private long serialVersionUID = 1L private Integer customerID private String customerName </pre>
<i>Operations</i>
<pre> public Customer() public Customer(Integer customerID) public Integer getCustomerID() public void setCustomerID(Integer customerID) public String getCustomerName() public void setCustomerName(String customerName) public SalesOrders[0..*] getSalesOrdersList() public void setSalesOrdersList(SalesOrders salesOrdersList[0..*]) public Contacts getContacts() public void setContacts(Contacts contacts) public int hashCode() public boolean equals(Object object) public String toString() </pre>

Figure 4.5 Class Customers

Products { From model }
<p><i>Attributes</i></p> <pre> private long serialVersionUID = 1L private Integer productID private String code private String name private BigDecimal price private Short isPurchas private Short isProduction private byte hardwareDesign[0..*] private byte softwareDesign[0..*] private Integer stockLevel </pre>
<p><i>Operations</i></p> <pre> public Products() public Products(Integer productID) public Integer getProductID() public void setProductID(Integer productID) public String getCode() public void setCode(String code) public String getName() public void setName(String name) public BigDecimal getPrice() public void setPrice(BigDecimal price) public Short getIsPurchas() public void setIsPurchas(Short isPurchas) public Short getIsProduction() public void setIsProduction(Short isProduction) public byte[0..*] getHardwareDesign() public void setHardwareDesign(byte hardwareDesign[0..*]) public byte[0..*] getSoftwareDesign() public void setSoftwareDesign(byte softwareDesign[0..*]) public Integer getStockLevel() public void setStockLevel(Integer stockLevel) public GRNLine[0..*] getGRNLineList() public void setGRNLineList(GRNLine gRNLineList[0..*]) public PurchasOrdersLines[0..*] getPurchasOrdersLinesList() public void setPurchasOrdersLinesList(PurchasOrdersLines purchasOrdersLinesList[0..*]) public ProductSuppliers[0..*] getProductSuppliersList() public void setProductSuppliersList(ProductSuppliers productSuppliersList[0..*]) public ProductBinCard[0..*] getProductBinCardList() public void setProductBinCardList(ProductBinCard productBinCardList[0..*]) public ProductIngredients[0..*] getProductIngredientsList() public void setProductIngredientsList(ProductIngredients productIngredientsList[0..*]) public ProductIngredients[0..*] getProductIngredientsList1() public void setProductIngredientsList1(ProductIngredients productIngredientsList1[0..*]) public SalesOrderLines[0..*] getSalesOrderLinesList() public void setSalesOrderLinesList(SalesOrderLines salesOrderLinesList[0..*]) public int hashCode() public boolean equals(Object object) public String toString() </pre>

Figure 4.6 Class Products

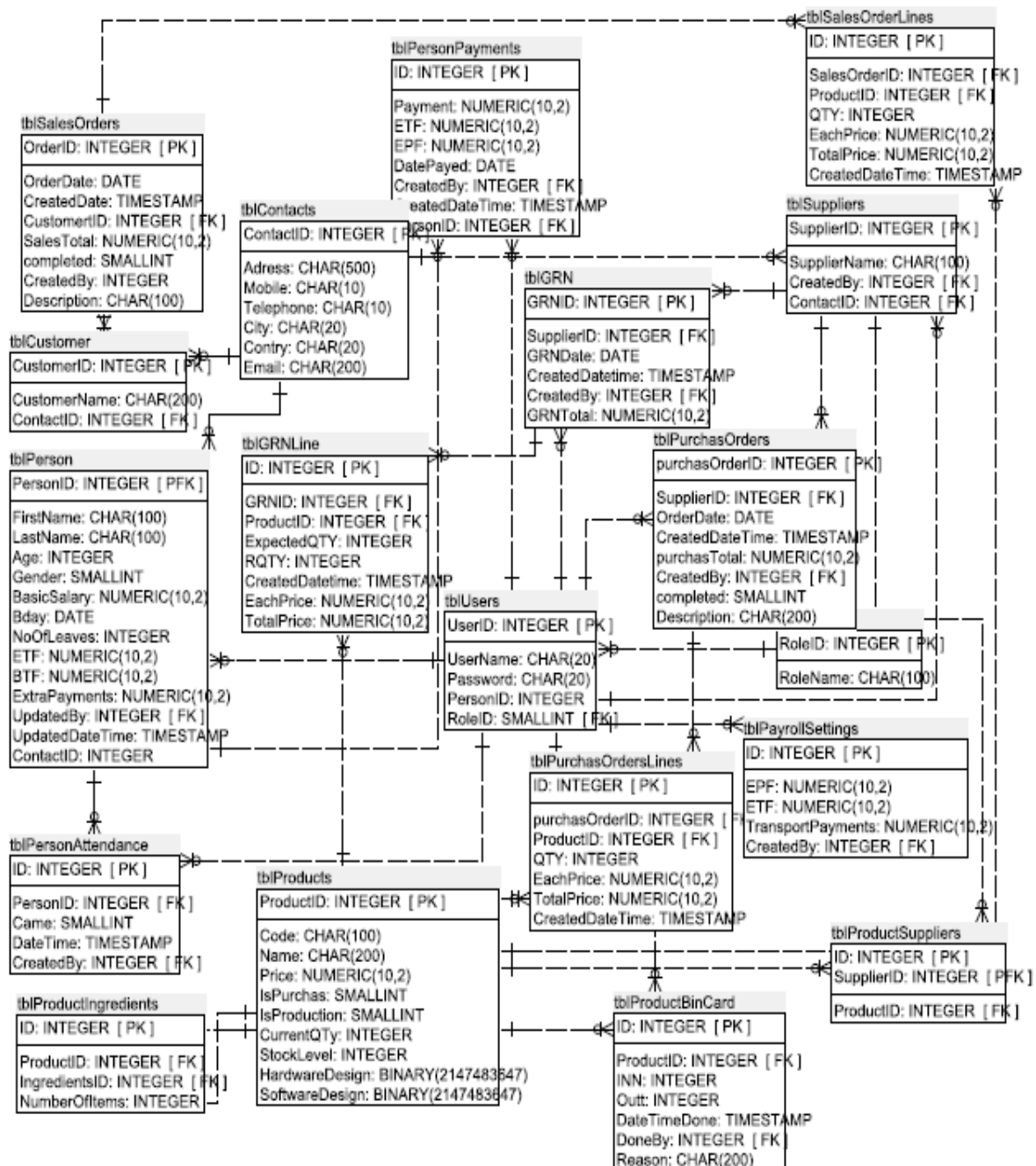
SalesOrders { From model }
<i>Attributes</i>
<pre> private long serialVersionUID = 1L private Integer orderID private Date orderDate private Date createdDate private BigDecimal salesTotal private Integer createdBy private String description private Short completed </pre>
<i>Operations</i>
<pre> public SalesOrders() public SalesOrders(Integer orderID) public Integer getOrderID() public void setOrderID(Integer orderID) public Date getOrderDate() public void setOrderDate(Date orderDate) public Date getCreatedDate() public void setCreatedDate(Date createdDate) public BigDecimal getSalesTotal() public void setSalesTotal(BigDecimal salesTotal) public Integer getCreatedBy() public void setCreatedBy(Integer createdBy) public String getDescription() public void setDescription(String description) public Short getCompleted() public void setCompleted(Short completed) public Customer getCustomer() public void setCustomer(Customer customer) public SalesOrderLines[0..*] getSalesOrderLinesList() public void setSalesOrderLinesList(SalesOrderLines salesOrderLinesList[0..*]) public int hashCode() public boolean equals(Object object) public String toString() </pre>

Figure 4.7 Class SalesOrders

Users { From model }
<i>Attributes</i> <pre> private long serialVersionUID = 1L private String userName private String password private Integer personID private Integer userID </pre>
<i>Operations</i> <pre> public Users() public Users(Integer userID) public Users(Integer userID, String userName, String password) public String getUserName() public void setUserName(String userName) public String getPassword() public void setPassword(String password) public Integer getPersonID() public void setPersonID(Integer personID) public Integer getUserID() public void setUserID(Integer userID) public PersonAttendance[0..*] getPersonAttendanceList() public void setPersonAttendanceList(PersonAttendance personAttendanceList[0..*]) public Person[0..*] getPersonList() public void setPersonList(Person personList[0..*]) public Roles getRoles() public void setRoles(Roles roles) public Suppliers[0..*] getSuppliersList() public void setSuppliersList(Suppliers suppliersList[0..*]) public GRN[0..*] getGRNList() public void setGRNList(GRN gRNList[0..*]) public PersonPayments[0..*] getPersonPaymentsList() public void setPersonPaymentsList(PersonPayments personPaymentsList[0..*]) public ProductBinCard[0..*] getProductBinCardList() public void setProductBinCardList(ProductBinCard productBinCardList[0..*]) public PayrollSettings[0..*] getPayrollSettingsList() public void setPayrollSettingsList(PayrollSettings payrollSettingsList[0..*]) public PurchasOrders[0..*] getPurchasOrdersList() public void setPurchasOrdersList(PurchasOrders purchasOrdersList[0..*]) public int hashCode() public boolean equals(Object object) public String toString() </pre>

Figure 4.8 Class Users

Date base diagram (Figure 4.9) which is designed by using SQL Power Architect for entire system is shown bellow



IMPLEMENTATION

Chapter

4

4.1 INTRODUCTION

Implementation was the most time consuming part of this project. It was used new language tools, which were unfamiliar. A Lot of literature reviews were been carried to learn new areas. Netbeans, Power Architect, Java Persistence API, Jasper Reports and I Report plug-in were very interesting to learn as they consist of easy and dynamic user friendly features.

4.2 IMPLEMENTATION ENVIRONMENT

Hardware and software implementation environment is been illustrated in the Figure 4.1 and Figure 4.2 respectively.

Processor Clock Speed	Intel Pentium IV(Recommended) 2.6GHz (minimum)
Memory	2GB
HDD	40GB
VGA Memory	32MB
Monitor	Minimum Resolution 800 * 600
Printer	Dot matrix printer A4 and Pos Printer for salary slip printing.
Network Card	Network interface card
Modem	Any Internet connectivity modem

Table 4.1 Hardware Configurations

Operating System	Windows (XP or above) Or Unix
Programming Language Requirement	JRE 1.6
Database Requirements	PostgreSQL 8.3 Database server

Table 4.2 Software Configurations

4.2.1 DEVELOPMENT TOOLS

Netbeans IDE 6.9

The Netbeans IDE is an award-winning integrated development environment available for Windows, Mac, Linux, and Solaris. The Netbeans project consists of an open-source IDE and an application platform that enable developers to rapidly create web, enterprise, desktop, and mobile applications using the Java platform, as well as JavaFX, PHP, JavaScript and Ajax, Ruby and Ruby on Rails, Groovy and Grails, and C/C++.

[WWW4]

PostgreSQL 8.3

PostgreSQL is an object-relational database management system (ORDBMS) based on POSTGRES, Version 4.2, developed at the University of California at Berkeley Computer Science Department. POSTGRES pioneered many concepts that only became available in some commercial database systems much later.

PostgreSQL is an open-source descendant of this original Berkeley code. It supports a large part of the SQL standard and offers many modern features.

[WWW8]

pgAdmin III

pgAdmin III is a comprehensive PostgreSQL database design and management system for Unix and Windows systems. It is freely available under the terms of the Artistic License and may be redistributed provided the terms of the License are adhered to. The project is managed by the pgAdmin Development Team.

[WWW9]

SQL Power Architect

Data Architects, DBA's, Analysts and Designers rely on Data Modeling tools to facilitate and simplify their data Modeling efforts, while maximizing the use of their resources. The SQL Power Architect allows these busy highly technical resources to perform this most intricate part of their job in a fraction of the time. [WWW5]

Java Persistence API

In fact, the Java community has produced numerous object-oriented approaches to data persistence: EJB, JDO, Hibernate, and Toplink are all worthy solutions that have tackled this problem. The Java Persistence API, or JPA, is a standard persistence API introduced as part of the Java EE 5 platform. The JPA specification was first introduced as part of JSR 220: EJB 3.0, with the goal of simplifying the EJB entity beans programming model. Although it all started with entity beans and is packaged with Java EE 5.0, JPA can be used outside the container in a Java SE environment. [WWW6]

Jasper Reports

Jasper Reports is the world's most popular open source reporting engine. It is entirely written in Java and it is able to use data coming from any kind of data source and produce pixel-perfect documents that can be viewed, printed or exported in a variety of document formats including HTML, PDF, Excel, Open Office and Word. [WWW7]

iReport

iReport is the free, open source report designer for Jasper Reports. Create very sophisticated layouts containing charts, images, sub reports, crosstabs and much more. Access your data through JDBC, Table Models, JavaBeans, XML, Hibernate, CSV, and custom sources. Then publish your reports as PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX, or Open Office. [WWW7]

4.3 CODE AND MODULE STRUCTURE

To make clear separation we have used four java packages

1. `erp.model` – To store all entity classes
2. `erp.controller` – To store all controller classes
3. `erp.gui.Interface` – To store all interface classes
2. `erp.controller.exceptions` – To store all exception classes
4. `erp.gui.InterfaceHelper` – To store interface helper classes like GUI validations

4.3.1 USER INTERFACE CLASSES

All the User interfaces of the system have separate interface class and adding with prefix of “frm” for their names. User interfaces only contain presentation logics of the application.

Some Of the classes are listed bellow.

frmCustomer

frmLogin

frmMainFrame

frmManageGRN

frmManagePayroll

And as a demonstrator purposes initialization (`initComponents` function) of `frmLogin` classis bellow and all interfaces are designs using Netbeans GUI designer.

```
private void initComponents() {  
    jButton1 = new javax.swing.JButton();  
    jLabel2 = new javax.swing.JLabel();  
    jPasswordField1 = new javax.swing.JPasswordField();  
    jLabel1 = new javax.swing.JLabel();  
    jTextField1 = new javax.swing.JTextField();  
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);  
    jButton1.setText("Login");  
    jButton1.addActionListener(new java.awt.event.ActionListener() {  
        public void actionPerformed(java.awt.event.ActionEvent evt) {  
            jButton1ActionPerformed(evt);  
        }  
    });  
}
```

```

    }
});
jLabel2.setText("Password");
jLabel1.setText("User Name");
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 400, Short.MAX_VALUE)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel1)
                .addComponent(jLabel2))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jPasswordField1, javax.swing.GroupLayout.DEFAULT_SIZE,
310, Short.MAX_VALUE)
                .addComponent(jTextField1, javax.swing.GroupLayout.DEFAULT_SIZE, 310,
Short.MAX_VALUE)))
                .addComponent(jButton1, javax.swing.GroupLayout.Alignment.TRAILING))
            .addContainerGap())
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 102, Short.MAX_VALUE)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()

```

```

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel1)
    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
    .addComponent(jLabel2)
    .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jButton1)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
pack();
}

```

4.3.2 CONTROLLER CLASSES

All the actions are performing under controller classes. So, most of the classes have relationships with these classes. Most part of the controller classes are generated from Netbeans by using JPA control class generations and have added extra methods when needed.

For the demonstrator purposes create customer [create(Customer customer)] function of the CustomerJpaController class is shown here.

```

public void create(Customer customer) throws PreexistingEntityException, Exception {
    if (customer.getSalesOrdersList() == null) {
        customer.setSalesOrdersList(new ArrayList<SalesOrders>());
    }
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
    }
}

```

```

    Contacts contacts = customer.getContacts();
    if (contacts != null) {
        contacts = em.getReference(contacts.getClass(), contacts.getContactID());
        customer.setContacts(contacts);
    }

    List<SalesOrders> attachedSalesOrdersList = new ArrayList<SalesOrders>();
    for (SalesOrders salesOrdersListSalesOrdersToAttach : customer.getSalesOrdersList()) {
        salesOrdersListSalesOrdersToAttach
em.getReference(salesOrdersListSalesOrdersToAttach.getClass(),
salesOrdersListSalesOrdersToAttach.getOrderID());
        attachedSalesOrdersList.add(salesOrdersListSalesOrdersToAttach);
    }
    customer.setSalesOrdersList(attachedSalesOrdersList);
    em.persist(customer);
    if (contacts != null) {
        contacts.getCustomerList().add(customer);
        contacts = em.merge(contacts);
    }
    for (SalesOrders salesOrdersListSalesOrders : customer.getSalesOrdersList()) {
        Customer oldCustomerOfSalesOrdersListSalesOrders
salesOrdersListSalesOrders.getCustomer();
        salesOrdersListSalesOrders.setCustomer(customer);
        salesOrdersListSalesOrders = em.merge(salesOrdersListSalesOrders);
        if (oldCustomerOfSalesOrdersListSalesOrders != null) {

oldCustomerOfSalesOrdersListSalesOrders.getSalesOrdersList().remove(salesOrdersListSalesOr
ders);

        oldCustomerOfSalesOrdersListSalesOrders
em.merge(oldCustomerOfSalesOrdersListSalesOrders);
        }
    }
    em.getTransaction().commit();
} catch (Exception ex) {
    if (findCustomer(customer.getCustomerID()) != null) {

```

```

        throw new PreexistingEntityException("Customer " + customer + " already exists.", ex);
    }
    throw ex;
} finally {
    if (em != null) {
        em.close();
    }
}
}
}

```

4.3.3 ENTITY CLASSES

Entity classes represent the entities of the system. Entities can be change through controller classes. All the Entity classes are generated by using the Netbean's Entity class generator and it was very useful to reduce the development time.

For the demonstrator purposes getter and setter functions of Customer class are shown here.

```

public Integer getCustomerID() {
    return customerID;
}
public void setCustomerID(Integer customerID) {
    this.customerID = customerID;
}
public String getCustomerName() {
    return customerName;
}
public void setCustomerName(String customerName) {
    this.customerName = customerName;
}
public List<SalesOrders> getSalesOrdersList() {
    return salesOrdersList;
}
public void setSalesOrdersList(List<SalesOrders> salesOrdersList) {

```

```

        this.salesOrdersList = salesOrdersList;
    }
    public Contacts getContacts() {
        return contacts;
    }
    public void setContacts(Contacts contacts) {
        this.contacts = contacts;
    }
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Customer)) {
            return false;
        }
        Customer other = (Customer) object;
        if ((this.customerID == null && other.customerID != null) || (this.customerID != null &&
!this.customerID.equals(other.customerID))) {
            return false;
        }
        return true;
    }
}

```

5.1 TESTING

Sanergy (pvt) Ltd used this system parallel to the current manual system and the testing was satisfied their most important requirements. Throughout the development of the project carried JUnit testing and it was used vast range of test cases. After implementation, functional level as well as the user level test carried.

5.1.1 TEST PLANS

Login (Table 5.1), Product (Table 5.2), Suppliers (Table 5.3), Customer (Table 5.4), Manage Purchase Order (Table 5.5) and Purchase Order (Table 5.6) Test plans are described bellow.

Name	Login	Screen	login	Tested date	
Test Case		Input		Expected Result	Status
Click OK Button		Enter Valid User ID and password		Login to the system	
		Enter Invalid User ID or password		Prompt Message "Invalid User ID or Password"	
		Enter Administrator Level User Id		Full Access	
		Enter Operator Level User Id		Full Access without Administration options.	
		Enter Manager Level User ID		Access only to Reports and Pay Slips.	
		Enter Basic User Lever User ID		Access only to Pay Slips.	
Click Close Button				Exit from the Login Screen	

Table 5.1 Login Test Case

Name		Screen	Products	Tested date	
Test Case		Input		Expected Result	Status
Add option and Done Button		Enter New Product Details Enter Existing Product Details Enter Insufficient Product Detail		Product Details will be saved and message will be displayed saying "Product Details Saved Successfully" Prompt Message "Product Already Exist" Prompt Message "Please Fill Necessary Fields"	
Edit option and Done Button		Enter Existing Product Details Enter Insufficient Product Details		Product Details will be updated and message will be displayed saying "Product Details Updated Successfully" Prompt Message "Please Fill Necessary Fields"	
Delete option and Done Button		Enter Existing Product Details		Delete Record	

Table 5.2 Product Test Case

Name		Screen	Suppliers	Tested date	
Test Case		Input		Expected Result	Status
Add option and Done Button		Enter New Suppliers Details Enter Existing Suppliers Details Enter Insufficient Suppliers Details		Suppliers Details will be saved and message will be displayed saying "Suppliers Details Saved Successfully" Prompt Message "Suppliers Already Exist" Prompt Message "Please Fill Necessary Fields"	
Edit option and Done Button		Enter Existing Suppliers Details Enter Insufficient Suppliers Details		Suppliers Details will be updated and message will be displayed saying "Suppliers Details Updated Successfully" Prompt Message "Please Fill Necessary Fields"	
Delete option and Done Button		Enter Existing Suppliers Details		Delete Record	

Table 5.3 Suppliers Test Case

Name		Screen	Customers	Tested date	
Test Case		Input		Expected Result	Status
Add option and Done Button		Enter New Customers Details		Customers Details will be saved and message will be displayed saying "Customers Details Saved Successfully"	
		Enter Existing Customers Details		Prompt Message "Customers Already Exist"	
		Enter Insufficient Customers Details		Prompt Message "Please Fill Necessary Fields"	
Edit option and Done Button		Enter Existing Customers Details		Customers Details will be updated and message will be displayed saying "Customers Details Updated Successfully"	
		Enter Insufficient Customers Details		Prompt Message "Please Fill Necessary Fields"	
Delete option and Done Button		Enter Existing Customers Details		Delete Record	

Table 5.4 Customer Test Case

Name		Screen	Sales Order	Tested date	
Test Case		Input		Expected Result	Status
Click Add Button		Enter Customer Name and Product		Details will be Added to the Table	
		Enter Insufficient Product		Prompt Message "Please Check Again"	
Click Create Button				Create Sales Order and Prompt Message "Sales Oder Created Successfully"	

Table 5.5 Sales Order Test Case

Name		Screen	Manage Purchase Order	Tested date	
Test Case		Input		Expected Result	Status
Click Save Button		Enter Existing Purchase Order Details		Sales Purchase Details will be updated and message will be displayed saying “Purchase Order Updated Successfully”	
		Enter Insufficient Purchase Order Details		Prompt Message “Please Fill Necessary Fields”	
Click Delete Button		Enter Existing Purchase Order Details		Delete Record	
Click Complete Purchase Order Button				Prompt Message “Successfully Completed Purchase Order”	

Table 5.6 Manage Purchase Order Test Case

Name		Screen	Purchase Order	Tested date	
Test Case		Input		Expected Result	Status
Click Add Button		Enter Supplier Name and Product		Details will be Added to the Table	
		Enter Insufficient Product		Prompt Message “Please Check Again”	
Click Create Button				Create Purchase Order and Prompt Message “Purchase Oder Created Successfully”	

Table 5.7 Purchase Order Test Plans

5.2 SAMPLE ACCEPTANCE TEST QUESTIONNAIRES WITH ANSWERS

All the functions are tested with using large amount of test data according to the test plans which are provided.

All reports are presented accurate results and quality output.

Finally, 13 Internal Uses were been given evaluation questionnaire on this system for future improvements. Actual user answers are summarize on the Table 5.8

Answers

1. What is your opinion regarding the reports of the system?

- | | | | |
|------|--------------------|-----|--------------------|
| i. | Yes, Very useful | ii. | Yes, Enough |
| iii. | Needs improvements | iv. | No, Not much |
| v. | Not at all | | |

2. Do you feel this system has the requirements you expected?

- | | | | |
|------|-----------------------|-----|--------------------|
| i. | Not at all | ii. | No, Not much |
| iii. | A little bit | iv. | Fairly complicated |
| v. | Yes, Very much | | |

3. Do you feel that the system is user friendly?

- | | | | |
|------|----------------------|-----|------------------|
| i. | Yes of course | ii. | Yes, but not all |
| iii. | Not much | iv. | Not at all |

4. Do you require further system demonstrations?

- | | | | |
|------|----------------|-----|------------------------|
| i. | Yes, Very mush | ii. | No, enough |
| iii. | Not decided | iv. | No, but not all |
| v. | Not at all | | |

5. Comments

Actual Answers Summary

Question No	Number of people - Option i	Number of people - Option ii	Number of people - Option iii	Number of people - Option iv	Number of people - Option v
1	1	8	4	0	0
2	2	3	0	1	7
3	10	2	1	0	0
4	3	2	3	5	0

Table 5.8 Actual Answers Summary

CONCLUSION

Chapter

6

6.1 ASSESSMENT OF PROJECT

6.1.1 AGAINST NON-FUNCTIONAL REQUIREMENTS

Maintainability

The code is been written clearly and has sufficiently comments so that it can be easily updated or extended. Also most of the tools we used for the development and design are up to date and open source.

Usability and User Friendliness

The screens were been designed to improve the user friendliness. Is has been designed to be pleasant to user, has a rich reporting content and a high level of interactivity.

6.1.2 AGAINST FUNCTIONAL REQUIREMENTS

The evaluation of the system against the set of functional requirements (Sales Orders, Purchase Orders, Requisitions, GRN, Payroll, Reports, Etc.) is useful as a way of accessing the amount of progress that made during the system.

Before they shifted to the ERP system, they will give three months period to test it with the manual work they are currently using. At the end of writing this document, the test period is not over. During the trial period of the system, they gave positive feedback. Most of the requirements they were expected were been met by the ERP system, after the trial period, I am suppose to give a questionnaire to check the system quality and effectiveness to the system users.

6.2 FUTURE WORKS

The system has few areas, which have been identified as to be further developed. The company accounts and production management could be handled from ERP system and it is not currently supported. This features may required by the company in future

6.3 KNOWLEDGE AND EXPERIENCE

Developer got a good knowledge in JPA, as it was a new emerging API. And also the developer was able to fine-tune the knowledge in Java, Netbeans, Jasper Reports, Uml and PostgreSQL when designing an ERP System for a company. It was a great challenge and great valuable experience gained by the developer, as that was the first complete software system designed by the developer for a company.

6.4 SUMMERY

The aim of the project is to produce an ERP system. The project aim has been fulfilled. All the requirements were formulated and the system is designed and produced.

.

References

- [WWW1] Functional Documentation/Sales Management, Openbravo
http://wiki.openbravo.com/wiki/Functional_Documentation/Sales_Management
- [WWW2] Functional Documentation/Procurement Management, Openbravo
http://wiki.openbravo.com/wiki/Functional_Documentation/Procurement_Management
- [WWW3] Functional Documentation/Warehouse Management, Openbravo
http://wiki.openbravo.com/wiki/Functional_Documentation/Warehouse_Management
- [WWW4] Netbeans IDE <http://www.netbeans.org>
- [WWW5] SQL Power Architect, <http://www.sqlpower.ca>
- [WWW6] Java Persistence API, <http://www.javaworld.com>
- [WWW7] Jasper Reports and iReports , <http://www.jasperforge.org>
- [WWW8] PostgreSQL, <http://www.postgresql.org/>
- [WWW9] pgAdmin III , <http://www.pgadmin.org/>
- [WWW10] Openbravo, <http://www.openbravo.com/>

APPENDIX A

System Documentation

A.1. INSTALLATION

1. Install JRE (Java Runtime Environment) from the From CDROM > Install Folder.
2. Install PostgreSQL 8.3 from the CDROM > Install Folder.
3. Import Database backup file(CDROM > DB) by using pgAdmin III
4. Copy ERP Folder (CDROM > ERP) To any location of your Hard Drive.
5. Create the Desktop shortcut for the ERPS.jar file.
6. Install POS and A4 printer.

Entire installation process in explained on the video tutorials section of the CDROM > Video Tutorials.

If you are under UNIX system, then you have to download, install and configure Respective JRE and PostgreSQL

A.2. COMPILATION AND EXECUTION

Double click on the Created shortcut of the ERPS.jar file, for the program execution.

For the compilation purpose use the entire project folder from CDROM>ERP PROJECT.

1. Open the project folder on Netbeans.
2. Do any changes that needed.
3. Click on Build.

APPENDIX B

USER DOCUMENTATION

In this section, it will briefly explain how a user can cooperate with the ERP system. For better understanding please go through the video tutorials section (CDROM > Video Tutorials) on the CDROM.

B.1 NOTES

- Always log out from the system, when complete your session
- Use Tab key to move to the next data entry position.
- Red color data entry fields are mandatory.

B.2 GETTING STARTED WITH LOGIN

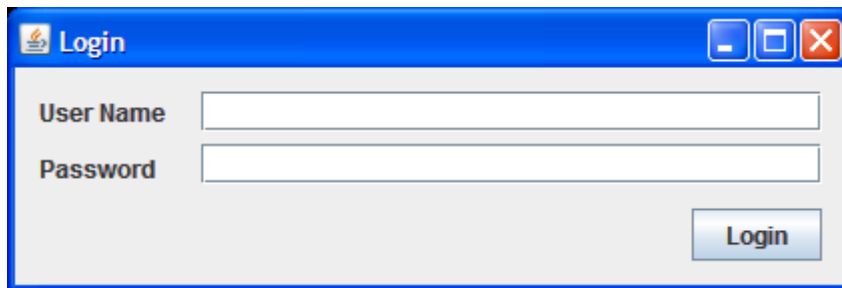


Figure B1 Login Screen

1. Provide both user name and password and click on the Login button.
2. You can try up to three times if u fails to provide correct details on third time application will exit.
3. Successful login will show Main Application window with appropriate menu items for the user.

B.3 MAIN APPLICATION WINDOW

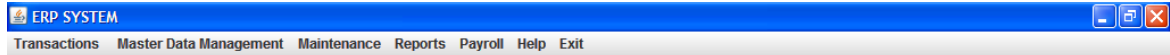
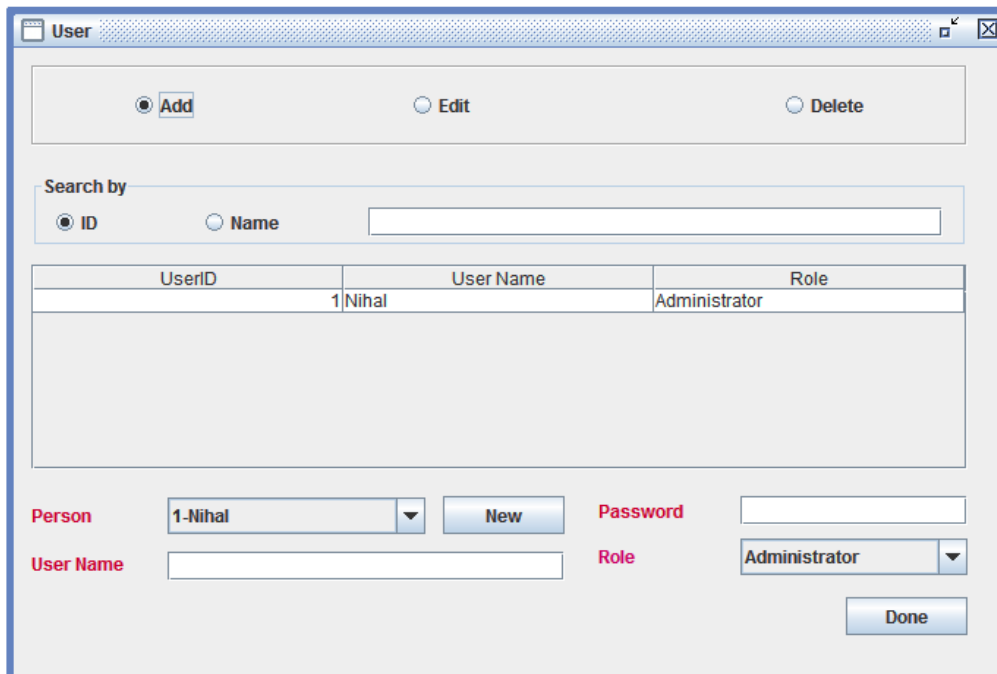


Figure B2 Main Application Window

1. Based on the login user role menu items will be load for the user.

Role Name	Menu(s)
Administrator	All menus
Operator	Transactions, Master Data Management, Reports, Payroll, Help, Exit
Manager	Reports, Help, Exit
User	Report Menu with only My Pay Slip Menu Item, Help, Exit

B.4 USER MANAGEMENT. (ONLY FOR ADMINISTRATOR ROLE)



The screenshot shows a 'User' management window. At the top, there are three radio buttons: 'Add' (selected), 'Edit', and 'Delete'. Below this is a 'Search by' section with two radio buttons: 'ID' (selected) and 'Name', followed by a text input field. A table displays user information with columns 'UserID', 'User Name', and 'Role'. The first row shows '1', 'Nihal', and 'Administrator'. Below the table, there are input fields for 'Person' (a dropdown menu showing '1-Nihal'), 'User Name', 'Password', and 'Role' (a dropdown menu showing 'Administrator'). A 'New' button is next to the 'Person' dropdown, and a 'Done' button is at the bottom right.

UserID	User Name	Role
1	Nihal	Administrator

Figure B3 User Management Screen

Select Add/Edit/Delete option buttons as needed.

Add

- I. Select the person from the drop down menu. If person is not existing then you have to add new person using New button. Person Screen will be under Master Data Management section.
- II. Provide all required Fields. (Red colored)
- III. Click the Done Button.

Edit

Search the user.

- I. Select search option ID or Name
- II. Type the respective search value in the sear text field.

Editing

- I, Click on the required user on the table which you want to update.
- II. Do the required changes.
- III. Click the Done Button.

Delete

Search the user.

- I. Select search option ID or Name
- II. Type the respective search value in the sear text field.

Deleting

- I, Click on the required user on the table which you want to Delete.
- II. Click the Done Button.

B.5 MASTER DATE MANAGEMENT. (ONLY FOR ADMINISTRATOR ROLE AND OPERATOR ROLE)

Products

The screenshot shows a software window titled "Products". At the top, there are three radio buttons: "Add" (selected), "Edit", and "Delete". Below this is a "Search by" section with two radio buttons: "ID" (selected) and "Name", followed by a text input field. In the center is a table with the following data:

ProductID	Code	Name	Purchas	Production	Price
1	B001	LED-RED	<input checked="" type="checkbox"/>	<input type="checkbox"/>	20

Below the table, there are several input fields and buttons. On the left, there are labels for "Code", "Name", "Price", "HW Design", "SW Design", and "Required Stock Level", each followed by a text input field. On the right, there are two radio buttons: "Production" (selected) and "Purchas". At the bottom right, there are three buttons: "Set BOM", "Define Suppliers", and "Done".

Figure B4 Product Management Screen

Select Add/Edit/Delete option buttons as needed.

Add

- I. Provide all required Fields. (Red colored)
- II. Click the Done Button.

Edit

Search.

- I. Select search option ID or Name
- II. Type the respective search value in the sear text field.

Editing

- I, Click the required row on the table which you want to update.
- II. Do the required changes.
- III. Click the Done Button.

Delete

Search.

- I. Select search option ID or Name
- II. Type the respective search value in the sear text field.

Deleting

I, Click the required row on the table which you want to delete.

II. Click the Done Button.

Set Bom Button will navigate to **Product Bom Screen** where you can define ingredients of the production products.

Define Supplier Button will navigate to **Product Supplier Screen** where you can define ingredients of the production products.

Both Screens are straight foreword user can select required data from the list and add them. To remove currently selected item user can select the required row from the table and click Remove Button

Product Bom Screen

ID	ProductID	Name	QTY
1	03	RED-LED	10

Figure B5 Product Bom Screen

Product Supplier Screen

SupplierID	Supplier Name
1	Supplier-ANC

Figure B6 Product Supplier Screen

Employee, Suppliers, Customers

Person

☒ Add ☐ Edit ☐ Delete

Search by
☒ ID ☐ First Name ☐ Last Name

Person...	First Name	Last Name	Age	Salary	A
1	Amal	Fernando	21	1,000	

First Name

Last Name

Age

Basic salary

Allowances

Mobile

Phone

City

Address

Country

Email

Done

Supplier

☒ Add ☐ Edit ☐ Delete

Search by
☒ ID ☐ Name

SupplierID	Supplier Name	Mobile	Phone	Email
1	Supplier-ANC	0777-898563	01122191456	officer@abc.com

Name

Address

Mobile

Phone

City

Country

Email

Done

Customers

☒ Add ☐ Edit ☐ Delete

Search by ☒ ID ☐ Name

CustomerID	Customer Name	Mobile	Phone	Email
1	Nihal	077156231	01122191555	nihal@test.com

Customer Name

Address

Mobile

Phone

City

Country

Email

Done

Figure B7 Employee, Suppliers, Customers Screens

Select Add/Edit/Delete option buttons as needed.

Add

- I. Provide all required Fields. (Red colored)
- II. Click the Done Button.

Edit

Search.

- I. Select search option ID or Name
- II. Type the respective search value in the sear text field.

Editing

- I, Click the required row on the table which you want to update.
- II. Do the required changes.
- III. Click the Done Button.

Delete

Search.

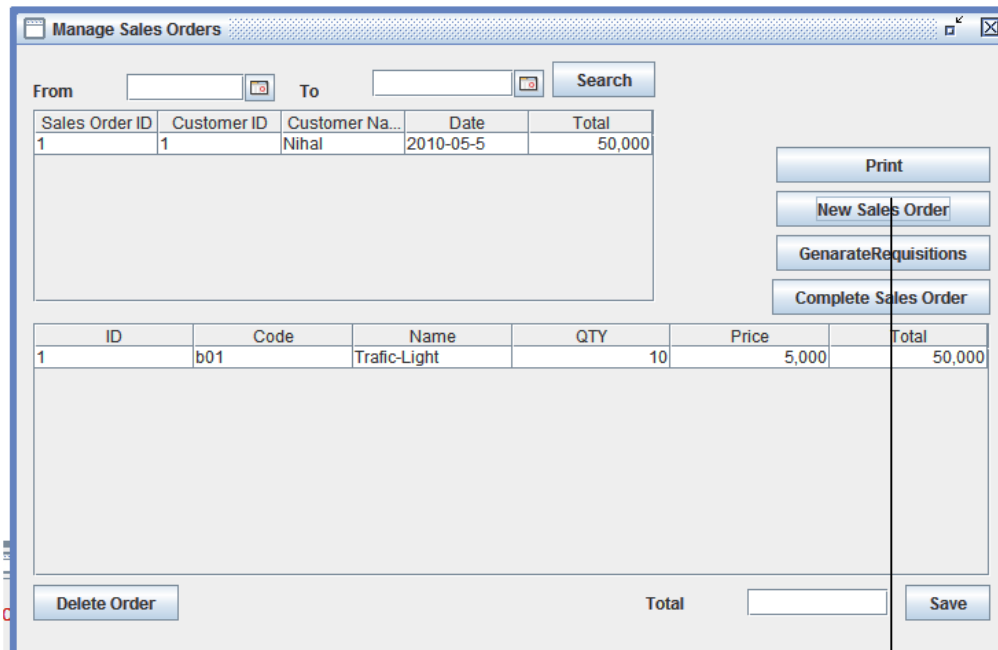
- I. Select search option ID or Name
- II. Type the respective search value in the sear text field.

Deleting

- I, Click the required row on the table which you want to delete.
- II. Click the Done Button.

B.6 TRANSACTION MANAGEMENT. (ONLY FOR ADMINISTRATOR ROLE AND OPERATOR ROLE)

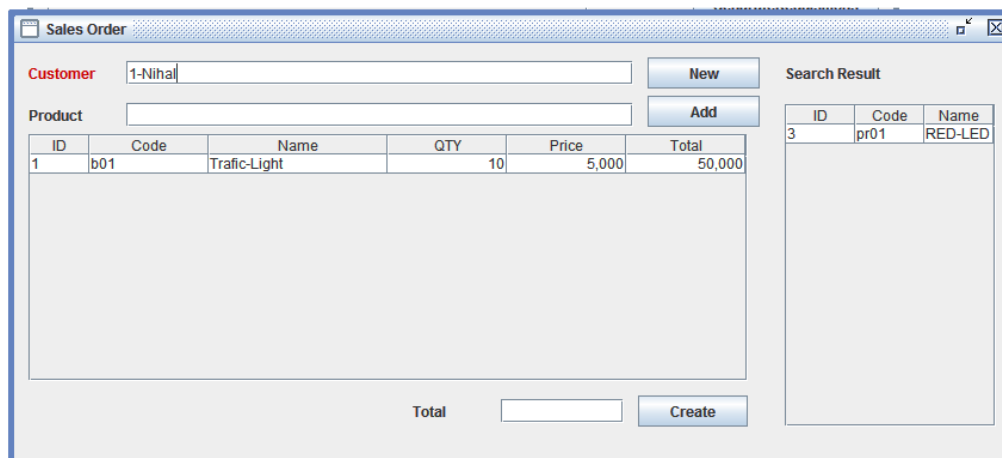
Only Screens are displayed here please refer to video tutorial section to understand this section of the system.



The 'Manage Sales Orders' window features a search bar at the top with 'From' and 'To' date pickers and a 'Search' button. Below the search bar is a table with one row: Sales Order ID 1, Customer ID 1, Customer Name Nihal, Date 2010-05-5, and Total 50,000. To the right of this table are four buttons: 'Print', 'New Sales Order', 'Generate Requisitions', and 'Complete Sales Order'. At the bottom left is a 'Delete Order' button. At the bottom right is a 'Total' label followed by an empty text box and a 'Save' button. A large table in the center contains one row: ID 1, Code b01, Name Traffic-Light, QTY 10, Price 5,000, and Total 50,000.

Sales Order ID	Customer ID	Customer Na...	Date	Total
1	1	Nihal	2010-05-5	50,000

ID	Code	Name	QTY	Price	Total
1	b01	Traffic-Light	10	5,000	50,000



The 'Sales Order' window has a 'Customer' field with '1-Nihal' and a 'New' button. Below it is a 'Product' field and an 'Add' button. On the right is a 'Search Result' table with one row: ID 3, Code pr01, Name RED-LED. At the bottom left is a 'Total' label followed by an empty text box and a 'Create' button. A large table in the center contains one row: ID 1, Code b01, Name Traffic-Light, QTY 10, Price 5,000, and Total 50,000.

ID	Code	Name	QTY	Price	Total
1	b01	Traffic-Light	10	5,000	50,000

ID	Code	Name
3	pr01	RED-LED

Manage Purchase Orders

From To

purchase Order ID	Supplier ID	Supplier Name	Date	Total
1	1	Suppoer-ANC	2010-05-5	20

ID	Code	Name	QTY	Price	Total
3	pr01	RED-LED	10	2	20

Total

Purchas Order

Supplier

Product

ID	Code	Name	QTY	Price	Total	Supplier
3	pr01	RED-LED	10	2	20	1-Supplier-ANC

Search Result

ID	Code	Name
3	pr01	RED-...

Total

Manage GRN

From To

purchase Order ID	Supplier ID	Supplier Name	Date	Total
1	1	Suppoer-ANC	2010-05-5	20

ID	Code	Name	Reruired QTY	QTY
3	pr01	RED-LED	10	10

Total

Requisition

Sales Order

Products For Production Products To Purchas ☒ All ☐ For Selected Production Product Select The Supplier For Product

ID	Code	Namne	QTY
1	b01	Trafic-LI...	10

ID	Code	Name	QTY	Supplier
3	pr01	LED-RED	100	1-Supplier-ANC

1-Supplier-ANC

Figure B8 Transaction Management Screens

B.7 REPORTS

Only Screens are displayed here please refer to video tutorial section to understand this section of the system.

The figure displays four separate report screens from a software application, each with a title bar and a close button. The 'Sales Order Report' and 'Purchase Order Report' screens include radio buttons for 'All' (selected), 'Not Completed', and 'Completed'. They also feature 'From Date' and 'To Date' text boxes with calendar icons, a dropdown menu for 'Customer' or 'Supplier' set to 'ALL', and a 'Load' button. The 'Attendance Report' screen has radio buttons for 'From Date' (selected), 'To Date', and 'Person', with a 'Load' button. The 'GRN Report' screen has radio buttons for 'All' (selected), 'Not Completed', and 'Completed', with 'From Date', 'To Date' text boxes, a 'Supplier' dropdown set to 'ALL', and a 'Load' button.

Sales Order Report

☒ All ☐ Not Completed ☐ Completed

From Date To Date

Customer ALL

Purchase Order Report

☒ All ☐ Not Completed ☐ Completed

From Date To Date

Supplier ALL

Attendance Report

☐ From Date To Date

Person ALL

GRN Report

☒ All ☐ Not Completed ☐ Completed

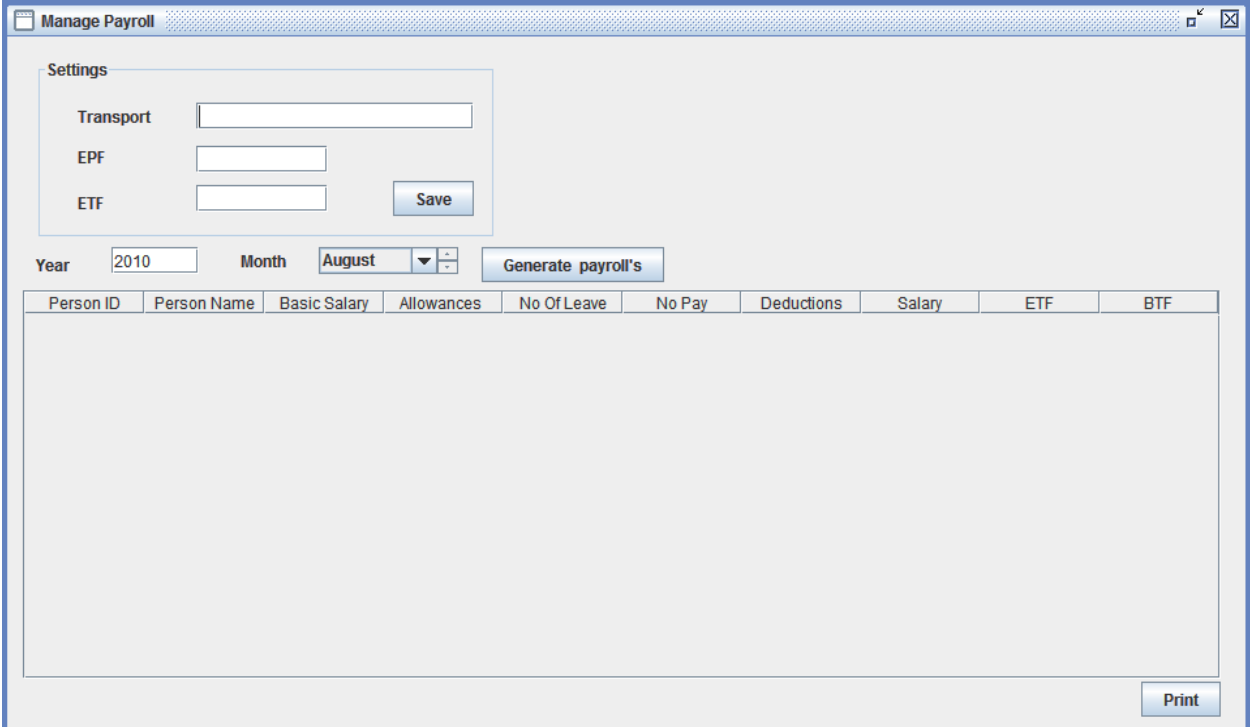
From Date To Date

Supplier ALL

Figure B9 Report Screens

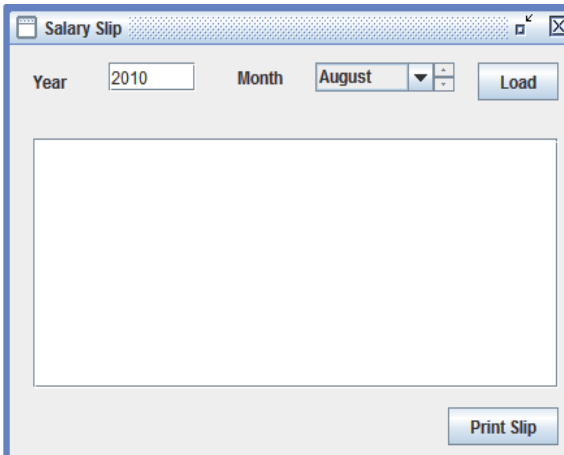
B.8 PAYROLL MANAGEMENT

Only Screens are displayed here please refer to video tutorial section to understand this section of the system.

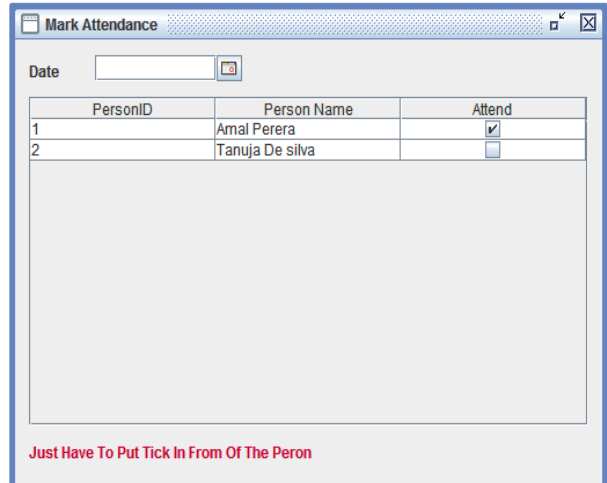


The 'Manage Payroll' window contains a 'Settings' section with input fields for 'Transport', 'EPF', and 'ETF', and a 'Save' button. Below this, there are 'Year' (2010) and 'Month' (August) dropdowns, and a 'Generate payroll's' button. A table with 10 columns is displayed: Person ID, Person Name, Basic Salary, Allowances, No Of Leave, No Pay, Deductions, Salary, ETF, and BTF. A 'Print' button is located at the bottom right.

Person ID	Person Name	Basic Salary	Allowances	No Of Leave	No Pay	Deductions	Salary	ETF	BTF
-----------	-------------	--------------	------------	-------------	--------	------------	--------	-----	-----



The 'Salary Slip' window features 'Year' (2010) and 'Month' (August) dropdowns, and a 'Load' button. A large empty rectangular area is provided for the slip content. A 'Print Slip' button is at the bottom right.



The 'Mark Attendance' window includes a 'Date' input field. It contains a table with 3 columns: PersonID, Person Name, and Attend. The table has two rows of data. A red message at the bottom states: 'Just Have To Put Tick In From Of The Peron'.

PersonID	Person Name	Attend
1	Amal Perera	<input checked="" type="checkbox"/>
2	Tanuja De silva	<input type="checkbox"/>

Just Have To Put Tick In From Of The Peron

Figure B10 Payroll Management Screens

APPENDIX C

MAJOR CODES

C. 1. SAMPLE ENTITY CLASSES CODES

```
/*  
 * To change this template, choose Tools | Templates  
 * and open the template in the editor.  
 */
```

```
package erp.model;
```

```
import java.io.Serializable;  
import java.math.BigDecimal;  
import java.util.Date;  
import java.util.List;  
import javax.persistence.Basic;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.JoinColumn;  
import javax.persistence.ManyToOne;  
import javax.persistence.NamedQueries;  
import javax.persistence.NamedQuery;  
import javax.persistence.OneToMany;  
import javax.persistence.OneToOne;  
import javax.persistence.Table;  
import javax.persistence.Temporal;  
import javax.persistence.TemporalType;
```

```
/**  
 *  
 * @author Administrator  
 */  
@Entity  
@Table(name = "tblPerson")
```

```

@NamedQueries({
    @NamedQuery(name = "Person.findAll", query = "SELECT p FROM Person p"),
    @NamedQuery(name = "Person.findByFirstName", query = "SELECT p FROM Person p WHERE
p.firstName = :firstName"),
    @NamedQuery(name = "Person.findByLastName", query = "SELECT p FROM Person p WHERE
p.lastName = :lastName"),
    @NamedQuery(name = "Person.findByAge", query = "SELECT p FROM Person p WHERE p.age =
:age"),
    @NamedQuery(name = "Person.findByGender", query = "SELECT p FROM Person p WHERE p.gender
= :gender"),
    @NamedQuery(name = "Person.findByBasicSalary", query = "SELECT p FROM Person p WHERE
p.basicSalary = :basicSalary"),
    @NamedQuery(name = "Person.findByBday", query = "SELECT p FROM Person p WHERE p.bday =
:bday"),
    @NamedQuery(name = "Person.findByEtf", query = "SELECT p FROM Person p WHERE p.etf = :etf"),
    @NamedQuery(name = "Person.findByBtf", query = "SELECT p FROM Person p WHERE p.btf = :btf"),
    @NamedQuery(name = "Person.findByExtraPayments", query = "SELECT p FROM Person p WHERE
p.extraPayments = :extraPayments"),
    @NamedQuery(name = "Person.findByUpdatedDateTime", query = "SELECT p FROM Person p
WHERE p.updatedDateTime = :updatedDateTime"),
    @NamedQuery(name = "Person.findByPersonID", query = "SELECT p FROM Person p WHERE
p.personID = :personID"),
    @NamedQuery(name = "Person.findByContactID", query = "SELECT p FROM Person p WHERE
p.contactID = :contactID"),
    @NamedQuery(name = "Person.findByNoOfLeaves", query = "SELECT p FROM Person p WHERE
p.noOfLeaves = :noOfLeaves"))
public class Person implements Serializable {
    private static final long serialVersionUID = 1L;
    @Basic(optional = false)
    @Column(name = "FirstName")
    private String firstName;
    @Basic(optional = false)
    @Column(name = "LastName")
    private String lastName;
    @Basic(optional = false)
    @Column(name = "Age")
    private int age;
    @Basic(optional = false)

```

```

@Column(name = "Gender")
private short gender;

@Column(optional = false)
@Column(name = "BasicSalary")
private BigDecimal basicSalary;

@Column(optional = false)
@Column(name = "Bday")
@Temporal(TemporalType.DATE)
private Date bday;

@Column(name = "ETF")
private BigDecimal etf;

@Column(name = "BTF")
private BigDecimal btf;

@Column(name = "ExtraPayments")
private BigDecimal extraPayments;

@Column(name = "UpdatedDateTime")
@Temporal(TemporalType.TIMESTAMP)
private Date updatedDateTime;

@Id
@Column(optional = false)
@Column(name = "PersonID")
private Integer personID;

@Column(name = "ContactID")
private Integer contactID;

@Column(name = "NoOfLeaves")
private Integer noOfLeaves;

@OneToMany(mappedBy = "person")
private List<PersonAttendance> personAttendanceList;

@JoinColumn(name = "UpdatedBy", referencedColumnName = "UserID")
@ManyToOne
private Users users;

@JoinColumn(name = "PersonID", referencedColumnName = "ContactID", insertable = false,
updatable = false)
@OneToOne(optional = false)
private Contacts contacts;

@OneToMany(mappedBy = "person")
private List<PersonPayments> personPaymentsList;

```

```

public Person() {
}

public Person(Integer personID) {
    this.personID = personID;
}

public Person(Integer personID, String firstName, String lastName, int age, short gender, BigDecimal
basicSalary, Date bday) {
    this.personID = personID;
    this.firstName = firstName;
    this.lastName = lastName;
    this.age = age;
    this.gender = gender;
    this.basicSalary = basicSalary;
    this.bday = bday;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public int getAge() {
    return age;
}

```



```
public void setAge(int age) {  
    this.age = age;  
}  
  
public short getGender() {  
    return gender;  
}  
  
public void setGender(short gender) {  
    this.gender = gender;  
}  
  
public BigDecimal getBasicSalary() {  
    return basicSalary;  
}  
  
public void setBasicSalary(BigDecimal basicSalary) {  
    this.basicSalary = basicSalary;  
}  
  
public Date getBday() {  
    return bday;  
}  
  
public void setBday(Date bday) {  
    this.bday = bday;  
}  
  
public BigDecimal getEtf() {  
    return etf;  
}  
  
public void setEtf(BigDecimal etf) {  
    this.etf = etf;  
}  
  
public BigDecimal getBtf() {  
    return btf;  
}
```

```

}

public void setBtf(BigDecimal btf) {
    this.btf = btf;
}

public BigDecimal getExtraPayments() {
    return extraPayments;
}

public void setExtraPayments(BigDecimal extraPayments) {
    this.extraPayments = extraPayments;
}

public Date getUpdatedDateTime() {
    return updatedDateTime;
}

public void setUpdatedDateTime(Date updatedDateTime) {
    this.updatedDateTime = updatedDateTime;
}

public Integer getPersonID() {
    return personID;
}

public void setPersonID(Integer personID) {
    this.personID = personID;
}

public Integer getContactID() {
    return contactID;
}

public void setContactID(Integer contactID) {
    this.contactID = contactID;
}

```

```

public Integer getNoOfLeaves() {
    return noOfLeaves;
}

public void setNoOfLeaves(Integer noOfLeaves) {
    this.noOfLeaves = noOfLeaves;
}

public List<PersonAttendance> getPersonAttendanceList() {
    return personAttendanceList;
}

public void setPersonAttendanceList(List<PersonAttendance> personAttendanceList) {
    this.personAttendanceList = personAttendanceList;
}

public Users getUsers() {
    return users;
}

public void setUsers(Users users) {
    this.users = users;
}

public Contacts getContacts() {
    return contacts;
}

public void setContacts(Contacts contacts) {
    this.contacts = contacts;
}

public List<PersonPayments> getPersonPaymentsList() {
    return personPaymentsList;
}

public void setPersonPaymentsList(List<PersonPayments> personPaymentsList) {
    this.personPaymentsList = personPaymentsList;
}

```

```

    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (personID != null ? personID.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        //TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Person)) {
            return false;
        }
        Person other = (Person) object;
        if ((this.personID == null && other.personID != null) || (this.personID != null &&
!this.personID.equals(other.personID))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "erp.model.Person[personID=" + personID + "]";
    }

}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package erp.model;

```

```

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;
import java.util.List;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

/**
 *
 * @author Administrator
 */
@Entity
@Table(name = "tblGRN")
@NamedQueries({
    @NamedQuery(name = "GRN.findAll", query = "SELECT g FROM GRN g"),
    @NamedQuery(name = "GRN.findByGrnid", query = "SELECT g FROM GRN g WHERE g.grnid = :grnid"),
    @NamedQuery(name = "GRN.findByGRNDate", query = "SELECT g FROM GRN g WHERE g.gRNDate = :gRNDate"),
    @NamedQuery(name = "GRN.findByCreatedDatetime", query = "SELECT g FROM GRN g WHERE g.createdDatetime = :createdDatetime"),
    @NamedQuery(name = "GRN.findByGRNTotal", query = "SELECT g FROM GRN g WHERE g.gRNTotal = :gRNTotal"))
public class GRN implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @Basic(optional = false)

```

```

@Column(name = "GRNID")
private Integer grnid;
@Column(name = "GRNDate")
@Temporal(TemporalType.DATE)
private Date gRNDate;
@Column(name = "CreatedDatetime")
@Temporal(TemporalType.TIMESTAMP)
private Date createdDatetime;
@Basic(optional = false)
@Column(name = "GRNTotal")
private BigDecimal gRNTotal;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "grn")
private List<GRNLine> gRNLineList;
@JoinColumn(name = "CreatedBy", referencedColumnName = "UserID")
@ManyToOne
private Users users;
@JoinColumn(name = "SupplierID", referencedColumnName = "SupplierID")
@ManyToOne
private Suppliers suppliers;

public GRN() {
}

public GRN(Integer grnid) {
    this.grnid = grnid;
}

public GRN(Integer grnid, BigDecimal gRNTotal) {
    this.grnid = grnid;
    this.gRNTotal = gRNTotal;
}

public Integer getGrnid() {
    return grnid;
}

public void setGrnid(Integer grnid) {
    this.grnid = grnid;
}

```

```

}

public Date getGRNDate() {
    return gRNDate;
}

public void setGRNDate(Date gRNDate) {
    this.gRNDate = gRNDate;
}

public Date getCreatedDatetime() {
    return createdDatetime;
}

public void setCreatedDatetime(Date createdDatetime) {
    this.createdDatetime = createdDatetime;
}

public BigDecimal getGRNTotal() {
    return gRNTotal;
}

public void setGRNTotal(BigDecimal gRNTotal) {
    this.gRNTotal = gRNTotal;
}

public List<GRNLine> getGRNLineList() {
    return gRNLineList;
}

public void setGRNLineList(List<GRNLine> gRNLineList) {
    this.gRNLineList = gRNLineList;
}

public Users getUsers() {
    return users;
}

```

```

public void setUsers(Users users) {
    this.users = users;
}

public Suppliers getSuppliers() {
    return suppliers;
}

public void setSuppliers(Suppliers suppliers) {
    this.suppliers = suppliers;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (grnid != null ? grnid.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not set
    if (!(object instanceof GRN)) {
        return false;
    }
    GRN other = (GRN) object;
    if ((this.grnid == null && other.grnid != null) || (this.grnid != null &&
!this.grnid.equals(other.grnid))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "erp.model.GRN[grnid=" + grnid + "]";
}

```



```

}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package erp.model;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;
import java.util.List;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

/**
 *
 * @author Administrator
 */
@Entity
@Table(name = "tblSalesOrders")
@NamedQueries({
    @NamedQuery(name = "SalesOrders.findAll", query = "SELECT s FROM SalesOrders s"),
    @NamedQuery(name = "SalesOrders.findByOrderID", query = "SELECT s FROM SalesOrders s
WHERE s.orderID = :orderID"),

```

```

    @NamedQuery(name = "SalesOrders.findByOrderDate", query = "SELECT s FROM SalesOrders s
WHERE s.orderDate = :orderDate"),
    @NamedQuery(name = "SalesOrders.findByCreatedDate", query = "SELECT s FROM SalesOrders s
WHERE s.createdDate = :createdDate"),
    @NamedQuery(name = "SalesOrders.findBySalesTotal", query = "SELECT s FROM SalesOrders s
WHERE s.salesTotal = :salesTotal"),
    @NamedQuery(name = "SalesOrders.findByCreatedBy", query = "SELECT s FROM SalesOrders s
WHERE s.createdBy = :createdBy"),
    @NamedQuery(name = "SalesOrders.findByDescription", query = "SELECT s FROM SalesOrders s
WHERE s.description = :description"),
    @NamedQuery(name = "SalesOrders.findByCompleted", query = "SELECT s FROM SalesOrders s
WHERE s.completed = :completed"))})
public class SalesOrders implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "OrderID")
    private Integer orderID;
    @Column(name = "OrderDate")
    @Temporal(TemporalType.DATE)
    private Date orderDate;
    @Column(name = "CreatedDate")
    @Temporal(TemporalType.TIMESTAMP)
    private Date createdDate;
    @Column(name = "SalesTotal")
    private BigDecimal salesTotal;
    @Column(name = "CreatedBy")
    private Integer createdBy;
    @Column(name = "Description")
    private String description;
    @Column(name = "completed")
    private Short completed;
    @JoinColumn(name = "CustomertID", referencedColumnName = "CustomerID")
    @ManyToOne
    private Customer customer;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "salesOrders")
    private List<SalesOrderLines> salesOrderLinesList;

```

```
public SalesOrders() {  
    }  
  
public SalesOrders(Integer orderID) {  
    this.orderID = orderID;  
}  
  
public Integer getOrderID() {  
    return orderID;  
}  
  
public void setOrderID(Integer orderID) {  
    this.orderID = orderID;  
}  
  
public Date getOrderDate() {  
    return orderDate;  
}  
  
public void setOrderDate(Date orderDate) {  
    this.orderDate = orderDate;  
}  
  
public Date getCreatedDate() {  
    return createdDate;  
}  
  
public void setCreatedDate(Date createdDate) {  
    this.createdDate = createdDate;  
}  
  
public BigDecimal getSalesTotal() {  
    return salesTotal;  
}  
  
public void setSalesTotal(BigDecimal salesTotal) {  
    this.salesTotal = salesTotal;  
}
```

```

public Integer getCreatedBy() {
    return createdBy;
}

public void setCreatedBy(Integer createdBy) {
    this.createdBy = createdBy;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public Short getCompleted() {
    return completed;
}

public void setCompleted(Short completed) {
    this.completed = completed;
}

public Customer getCustomer() {
    return customer;
}

public void setCustomer(Customer customer) {
    this.customer = customer;
}

public List<SalesOrderLines> getSalesOrderLinesList() {
    return salesOrderLinesList;
}

public void setSalesOrderLinesList(List<SalesOrderLines> salesOrderLinesList) {

```

```

        this.salesOrderLinesList = salesOrderLinesList;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (orderId != null ? orderId.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof SalesOrders)) {
            return false;
        }
        SalesOrders other = (SalesOrders) object;
        if ((this.orderID == null && other.orderID != null) || (this.orderID != null &&
!this.orderID.equals(other.orderID))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "erp.model.SalesOrders[orderId=" + orderId + "]";
    }

}

```

C.2. SAMPLE CONTROLLER CLASSES CODES

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package erp.controller;

import erp.controller.exceptions.NonexistentEntityException;
import erp.controller.exceptions.PreexistingEntityException;

```

```

import erp.model.Customer;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.persistence.EntityNotFoundException;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import erp.model.Contacts;
import erp.model.SalesOrders;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Administrator
 */
public class CustomerJpaController {

    public CustomerJpaController() {
        emf = Persistence.createEntityManagerFactory("ERPSPU");
    }
    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    public void create(Customer customer) throws PreexistingEntityException, Exception {
        if (customer.getSalesOrdersList() == null) {
            customer.setSalesOrdersList(new ArrayList<SalesOrders>());
        }
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            Contacts contacts = customer.getContacts();
            if (contacts != null) {
                contacts = em.getReference(contacts.getClass(), contacts.getContactID());
                customer.setContacts(contacts);
            }
            List<SalesOrders> attachedSalesOrdersList = new ArrayList<SalesOrders>();
            for (SalesOrders salesOrdersListSalesOrdersToAttach : customer.getSalesOrdersList()) {
                salesOrdersListSalesOrdersToAttach =
em.getReference(salesOrdersListSalesOrdersToAttach.getClass(),
salesOrdersListSalesOrdersToAttach.getOrderID());
                attachedSalesOrdersList.add(salesOrdersListSalesOrdersToAttach);
            }
            customer.setSalesOrdersList(attachedSalesOrdersList);
            em.persist(customer);
            if (contacts != null) {
                contacts.getCustomerList().add(customer);
                contacts = em.merge(contacts);
            }
            for (SalesOrders salesOrdersListSalesOrders : customer.getSalesOrdersList()) {

```

```

        Customer oldCustomerOfSalesOrdersListSalesOrders =
salesOrdersListSalesOrders.getCustomer();
        salesOrdersListSalesOrders.setCustomer(customer);
        salesOrdersListSalesOrders = em.merge(salesOrdersListSalesOrders);
        if (oldCustomerOfSalesOrdersListSalesOrders != null) {

oldCustomerOfSalesOrdersListSalesOrders.getSalesOrdersList().remove(salesOrdersListSalesOrders);
        oldCustomerOfSalesOrdersListSalesOrders =
em.merge(oldCustomerOfSalesOrdersListSalesOrders);
        }
    }
    em.getTransaction().commit();
} catch (Exception ex) {
    if (findCustomer(customer.getCustomerID()) != null) {
        throw new PreexistingEntityException("Customer " + customer + " already exists.", ex);
    }
    throw ex;
} finally {
    if (em != null) {
        em.close();
    }
}
}
}

```

```

public void edit(Customer customer) throws NonexistentEntityException, Exception {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        Customer persistentCustomer = em.find(Customer.class, customer.getCustomerID());
        Contacts contactsOld = persistentCustomer.getContacts();
        Contacts contactsNew = customer.getContacts();
        List<SalesOrders> salesOrdersListOld = persistentCustomer.getSalesOrdersList();
        List<SalesOrders> salesOrdersListNew = customer.getSalesOrdersList();
        if (contactsNew != null) {
            contactsNew = em.getReference(contactsNew.getClass(), contactsNew.getContactID());
            customer.setContacts(contactsNew);
        }
        List<SalesOrders> attachedSalesOrdersListNew = new ArrayList<SalesOrders>();
        for (SalesOrders salesOrdersListNewSalesOrdersToAttach : salesOrdersListNew) {
            salesOrdersListNewSalesOrdersToAttach =
em.getReference(salesOrdersListNewSalesOrdersToAttach.getClass(),
salesOrdersListNewSalesOrdersToAttach.getOrderID());
            attachedSalesOrdersListNew.add(salesOrdersListNewSalesOrdersToAttach);
        }
        salesOrdersListNew = attachedSalesOrdersListNew;
        customer.setSalesOrdersList(salesOrdersListNew);
        customer = em.merge(customer);
        if (contactsOld != null && !contactsOld.equals(contactsNew)) {
            contactsOld.getCustomerList().remove(customer);
            contactsOld = em.merge(contactsOld);
        }
        if (contactsNew != null && !contactsNew.equals(contactsOld)) {
            contactsNew.getCustomerList().add(customer);
            contactsNew = em.merge(contactsNew);
        }
    }
}

```

```

        for (SalesOrders salesOrdersListOldSalesOrders : salesOrdersListOld) {
            if (!salesOrdersListNew.contains(salesOrdersListOldSalesOrders)) {
                salesOrdersListOldSalesOrders.setCustomer(null);
                salesOrdersListOldSalesOrders = em.merge(salesOrdersListOldSalesOrders);
            }
        }
        for (SalesOrders salesOrdersListNewSalesOrders : salesOrdersListNew) {
            if (!salesOrdersListOld.contains(salesOrdersListNewSalesOrders)) {
                Customer oldCustomerOfSalesOrdersListNewSalesOrders =
salesOrdersListNewSalesOrders.getCustomer();
                salesOrdersListNewSalesOrders.setCustomer(customer);
                salesOrdersListNewSalesOrders = em.merge(salesOrdersListNewSalesOrders);
                if (oldCustomerOfSalesOrdersListNewSalesOrders != null &&
!oldCustomerOfSalesOrdersListNewSalesOrders.equals(customer)) {

oldCustomerOfSalesOrdersListNewSalesOrders.getSalesOrdersList().remove(salesOrdersListNewSalesOrd
ers);

                oldCustomerOfSalesOrdersListNewSalesOrders =
em.merge(oldCustomerOfSalesOrdersListNewSalesOrders);
            }
        }
        em.getTransaction().commit();
    } catch (Exception ex) {
        String msg = ex.getMessage();
        if (msg == null || msg.length() == 0) {
            Integer id = customer.getCustomerID();
            if (findCustomer(id) == null) {
                throw new NonexistentEntityException("The customer with id " + id + " no longer exists.");
            }
        }
        throw ex;
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public void destroy(Integer id) throws NonexistentEntityException {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        Customer customer;
        try {
            customer = em.getReference(Customer.class, id);
            customer.getCustomerID();
        } catch (EntityNotFoundException enfe) {
            throw new NonexistentEntityException("The customer with id " + id + " no longer exists.", enfe);
        }
        Contacts contacts = customer.getContacts();
        if (contacts != null) {
            contacts.getCustomerList().remove(customer);
            contacts = em.merge(contacts);
        }
    }
}

```



```

        List<SalesOrders> salesOrdersList = customer.getSalesOrdersList();
        for (SalesOrders salesOrdersListSalesOrders : salesOrdersList) {
            salesOrdersListSalesOrders.setCustomer(null);
            salesOrdersListSalesOrders = em.merge(salesOrdersListSalesOrders);
        }
        em.remove(customer);
        em.getTransaction().commit();
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public List<Customer> findCustomerEntities() {
    return findCustomerEntities(true, -1, -1);
}

public List<Customer> findCustomerEntities(int maxResults, int firstResult) {
    return findCustomerEntities(false, maxResults, firstResult);
}

private List<Customer> findCustomerEntities(boolean all, int maxResults, int firstResult) {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(Customer.class));
        Query q = em.createQuery(cq);
        if (!all) {
            q.setMaxResults(maxResults);
            q.setFirstResult(firstResult);
        }
        return q.getResultList();
    } finally {
        em.close();
    }
}

public Customer findCustomer(Integer id) {
    EntityManager em = getEntityManager();
    try {
        return em.find(Customer.class, id);
    } finally {
        em.close();
    }
}

public int getCustomerCount() {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        Root<Customer> rt = cq.from(Customer.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    } finally {

```

```

        em.close();
    }
}

}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package erp.controller;

import erp.controller.exceptions.NonexistentEntityException;
import erp.controller.exceptions.PreexistingEntityException;
import erp.model.GRNLine;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.persistence.EntityNotFoundException;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import erp.model.Products;
import erp.model.GRN;

/**
 *
 * @author Administrator
 */
public class GRNLineJpaController {

    public GRNLineJpaController() {
        emf = Persistence.createEntityManagerFactory("ERPSPU");
    }
    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    public void create(GRNLine GRNLine) throws PreexistingEntityException, Exception {
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            Products products = GRNLine.getProducts();
            if (products != null) {
                products = em.getReference(products.getClass(), products.getProductID());
                GRNLine.setProducts(products);
            }
            GRN grn = GRNLine.getGrn();
            if (grn != null) {

```

```

        grn = em.getReference(grn.getClass(), grn.getGrnid());
        GRNLine.setGrn(grn);
    }
    em.persist(GRNLine);
    if (products != null) {
        products.getGRNLineList().add(GRNLine);
        products = em.merge(products);
    }
    if (grn != null) {
        grn.getGRNLineList().add(GRNLine);
        grn = em.merge(grn);
    }
    em.getTransaction().commit();
} catch (Exception ex) {
    if (findGRNLine(GRNLine.getId()) != null) {
        throw new PreexistingEntityException("GRNLine " + GRNLine + " already exists.", ex);
    }
    throw ex;
} finally {
    if (em != null) {
        em.close();
    }
}
}
}

```

```

public void edit(GRNLine GRNLine) throws NonexistentEntityException, Exception {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        GRNLine persistentGRNLine = em.find(GRNLine.class, GRNLine.getId());
        Products productsOld = persistentGRNLine.getProducts();
        Products productsNew = GRNLine.getProducts();
        GRN grnOld = persistentGRNLine.getGrn();
        GRN grnNew = GRNLine.getGrn();
        if (productsNew != null) {
            productsNew = em.getReference(productsNew.getClass(), productsNew.getProductID());
            GRNLine.setProducts(productsNew);
        }
        if (grnNew != null) {
            grnNew = em.getReference(grnNew.getClass(), grnNew.getGrnid());
            GRNLine.setGrn(grnNew);
        }
        GRNLine = em.merge(GRNLine);
        if (productsOld != null && !productsOld.equals(productsNew)) {
            productsOld.getGRNLineList().remove(GRNLine);
            productsOld = em.merge(productsOld);
        }
        if (productsNew != null && !productsNew.equals(productsOld)) {
            productsNew.getGRNLineList().add(GRNLine);
            productsNew = em.merge(productsNew);
        }
        if (grnOld != null && !grnOld.equals(grnNew)) {
            grnOld.getGRNLineList().remove(GRNLine);
            grnOld = em.merge(grnOld);
        }
    }
}

```

```

        if (grnNew != null && !grnNew.equals(grnOld)) {
            grnNew.getGRNLineList().add(GRNLine);
            grnNew = em.merge(grnNew);
        }
        em.getTransaction().commit();
    } catch (Exception ex) {
        String msg = ex.getLocalizedMessage();
        if (msg == null || msg.length() == 0) {
            Integer id = GRNLine.getId();
            if (findGRNLine(id) == null) {
                throw new NonexistentEntityException("The grNLine with id " + id + " no longer exists.");
            }
        }
        throw ex;
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public void destroy(Integer id) throws NonexistentEntityException {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        GRNLine GRNLine;
        try {
            GRNLine = em.getReference(GRNLine.class, id);
            GRNLine.getId();
        } catch (EntityNotFoundException enfe) {
            throw new NonexistentEntityException("The GRNLine with id " + id + " no longer exists.", enfe);
        }
        Products products = GRNLine.getProducts();
        if (products != null) {
            products.getGRNLineList().remove(GRNLine);
            products = em.merge(products);
        }
        GRN grn = GRNLine.getGrn();
        if (grn != null) {
            grn.getGRNLineList().remove(GRNLine);
            grn = em.merge(grn);
        }
        em.remove(GRNLine);
        em.getTransaction().commit();
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public List<GRNLine> findGRNLineEntities() {
    return findGRNLineEntities(true, -1, -1);
}

```

```

public List<GRNLine> findGRNLineEntities(int maxResults, int firstResult) {
    return findGRNLineEntities(false, maxResults, firstResult);
}

private List<GRNLine> findGRNLineEntities(boolean all, int maxResults, int firstResult) {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(GRNLine.class));
        Query q = em.createQuery(cq);
        if (!all) {
            q.setMaxResults(maxResults);
            q.setFirstResult(firstResult);
        }
        return q.getResultList();
    } finally {
        em.close();
    }
}

public GRNLine findGRNLine(Integer id) {
    EntityManager em = getEntityManager();
    try {
        return em.find(GRNLine.class, id);
    } finally {
        em.close();
    }
}

public int getGRNLineCount() {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        Root<GRNLine> rt = cq.from(GRNLine.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    } finally {
        em.close();
    }
}

}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package erp.controller;

import erp.controller.exceptions.IllegalOrphanException;
import erp.controller.exceptions.NonexistentEntityException;
import erp.controller.exceptions.PreexistingEntityException;
import erp.model.Person;
import javax.persistence.EntityManager;

```

```

import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.persistence.EntityNotFoundException;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import erp.model.Users;
import erp.model.Contacts;
import erp.model.PersonAttendance;
import java.util.ArrayList;
import java.util.List;
import erp.model.PersonPayments;

/**
 *
 * @author Administrator
 */
public class PersonJpaController {

    public PersonJpaController() {
        emf = Persistence.createEntityManagerFactory("ERPSPU");
    }
    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    public void create(Person person) throws IllegalOrphanException, PreexistingEntityException, Exception
    {
        if (person.getPersonAttendanceList() == null) {
            person.setPersonAttendanceList(new ArrayList<PersonAttendance>());
        }
        if (person.getPersonPaymentsList() == null) {
            person.setPersonPaymentsList(new ArrayList<PersonPayments>());
        }
        List<String> illegalOrphanMessages = null;
        Contacts contactsOrphanCheck = person.getContacts();
        if (contactsOrphanCheck != null) {
            Person oldPersonOfContacts = contactsOrphanCheck.getPerson();
            if (oldPersonOfContacts != null) {
                if (illegalOrphanMessages == null) {
                    illegalOrphanMessages = new ArrayList<String>();
                }
                illegalOrphanMessages.add("The Contacts " + contactsOrphanCheck + " already has an item of
type Person whose contacts column cannot be null. Please make another selection for the contacts field.");
            }
        }
        if (illegalOrphanMessages != null) {
            throw new IllegalOrphanException(illegalOrphanMessages);
        }
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            Users users = person.getUsers();

```

```

    if (users != null) {
        users = em.getReference(users.getClass(), users.getUserID());
        person.setUsers(users);
    }
    Contacts contacts = person.getContacts();
    if (contacts != null) {
        contacts = em.getReference(contacts.getClass(), contacts.getContactID());
        person.setContacts(contacts);
    }
    List<PersonAttendance> attachedPersonAttendanceList = new ArrayList<PersonAttendance>();
    for (PersonAttendance personAttendanceListPersonAttendanceToAttach :
person.getPersonAttendanceList()) {
        personAttendanceListPersonAttendanceToAttach =
em.getReference(personAttendanceListPersonAttendanceToAttach.getClass(),
personAttendanceListPersonAttendanceToAttach.getId());
        attachedPersonAttendanceList.add(personAttendanceListPersonAttendanceToAttach);
    }
    person.setPersonAttendanceList(attachedPersonAttendanceList);
    List<PersonPayments> attachedPersonPaymentsList = new ArrayList<PersonPayments>();
    for (PersonPayments personPaymentsListPersonPaymentsToAttach :
person.getPersonPaymentsList()) {
        personPaymentsListPersonPaymentsToAttach =
em.getReference(personPaymentsListPersonPaymentsToAttach.getClass(),
personPaymentsListPersonPaymentsToAttach.getId());
        attachedPersonPaymentsList.add(personPaymentsListPersonPaymentsToAttach);
    }
    person.setPersonPaymentsList(attachedPersonPaymentsList);
    em.persist(person);
    if (users != null) {
        users.getPersonList().add(person);
        users = em.merge(users);
    }
    if (contacts != null) {
        contacts.setPerson(person);
        contacts = em.merge(contacts);
    }
    for (PersonAttendance personAttendanceListPersonAttendance : person.getPersonAttendanceList())
    {
        Person oldPersonOfPersonAttendanceListPersonAttendance =
personAttendanceListPersonAttendance.getPerson();
        personAttendanceListPersonAttendance.setPerson(person);
        personAttendanceListPersonAttendance = em.merge(personAttendanceListPersonAttendance);
        if (oldPersonOfPersonAttendanceListPersonAttendance != null) {

oldPersonOfPersonAttendanceListPersonAttendance.getPersonAttendanceList().remove(personAttendanceL
istPersonAttendance);
            oldPersonOfPersonAttendanceListPersonAttendance =
em.merge(oldPersonOfPersonAttendanceListPersonAttendance);
        }
    }
    for (PersonPayments personPaymentsListPersonPayments : person.getPersonPaymentsList()) {
        Person oldPersonOfPersonPaymentsListPersonPayments =
personPaymentsListPersonPayments.getPerson();
        personPaymentsListPersonPayments.setPerson(person);
        personPaymentsListPersonPayments = em.merge(personPaymentsListPersonPayments);
        if (oldPersonOfPersonPaymentsListPersonPayments != null) {

```

```

oldPersonOfPersonPaymentsListPersonPayments.remove(personPaymentsListPer
sonPayments);
    oldPersonOfPersonPaymentsListPersonPayments =
em.merge(oldPersonOfPersonPaymentsListPersonPayments);
    }
    }
    em.getTransaction().commit();
} catch (Exception ex) {
    if (findPerson(person.getPersonID()) != null) {
        throw new PreexistingEntityException("Person " + person + " already exists.", ex);
    }
    throw ex;
} finally {
    if (em != null) {
        em.close();
    }
}
}
}

```

```

public void edit(Person person) throws IllegalOrphanException, NonexistentEntityException, Exception {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        Person persistentPerson = em.find(Person.class, person.getPersonID());
        Users usersOld = persistentPerson.getUsers();
        Users usersNew = person.getUsers();
        Contacts contactsOld = persistentPerson.getContacts();
        Contacts contactsNew = person.getContacts();
        List<PersonAttendance> personAttendanceListOld = persistentPerson.getPersonAttendanceList();
        List<PersonAttendance> personAttendanceListNew = person.getPersonAttendanceList();
        List<PersonPayments> personPaymentsListOld = persistentPerson.getPersonPaymentsList();
        List<PersonPayments> personPaymentsListNew = person.getPersonPaymentsList();
        List<String> illegalOrphanMessages = null;
        if (contactsNew != null && !contactsNew.equals(contactsOld)) {
            Person oldPersonOfContacts = contactsNew.getPerson();
            if (oldPersonOfContacts != null) {
                if (illegalOrphanMessages == null) {
                    illegalOrphanMessages = new ArrayList<String>();
                }
                illegalOrphanMessages.add("The Contacts " + contactsNew + " already has an item of type
Person whose contacts column cannot be null. Please make another selection for the contacts field.");
            }
        }
        if (illegalOrphanMessages != null) {
            throw new IllegalOrphanException(illegalOrphanMessages);
        }
        if (usersNew != null) {
            usersNew = em.getReference(usersNew.getClass(), usersNew.getUserID());
            person.setUsers(usersNew);
        }
        if (contactsNew != null) {
            contactsNew = em.getReference(contactsNew.getClass(), contactsNew.getContactID());
            person.setContacts(contactsNew);
        }
    }
}

```



```

        List<PersonAttendance> attachedPersonAttendanceListNew = new
ArrayList<PersonAttendance>();
        for (PersonAttendance personAttendanceListNewPersonAttendanceToAttach :
personAttendanceListNew) {
            personAttendanceListNewPersonAttendanceToAttach =
em.getReference(personAttendanceListNewPersonAttendanceToAttach.getClass(),
personAttendanceListNewPersonAttendanceToAttach.getId());
            attachedPersonAttendanceListNew.add(personAttendanceListNewPersonAttendanceToAttach);
        }
        personAttendanceListNew = attachedPersonAttendanceListNew;
        person.setPersonAttendanceList(personAttendanceListNew);
        List<PersonPayments> attachedPersonPaymentsListNew = new ArrayList<PersonPayments>();
        for (PersonPayments personPaymentsListNewPersonPaymentsToAttach : personPaymentsListNew)
        {
            personPaymentsListNewPersonPaymentsToAttach =
em.getReference(personPaymentsListNewPersonPaymentsToAttach.getClass(),
personPaymentsListNewPersonPaymentsToAttach.getId());
            attachedPersonPaymentsListNew.add(personPaymentsListNewPersonPaymentsToAttach);
        }
        personPaymentsListNew = attachedPersonPaymentsListNew;
        person.setPersonPaymentsList(personPaymentsListNew);
        person = em.merge(person);
        if (usersOld != null && !usersOld.equals(usersNew)) {
            usersOld.getPersonList().remove(person);
            usersOld = em.merge(usersOld);
        }
        if (usersNew != null && !usersNew.equals(usersOld)) {
            usersNew.getPersonList().add(person);
            usersNew = em.merge(usersNew);
        }
        if (contactsOld != null && !contactsOld.equals(contactsNew)) {
            contactsOld.setPerson(null);
            contactsOld = em.merge(contactsOld);
        }
        if (contactsNew != null && !contactsNew.equals(contactsOld)) {
            contactsNew.setPerson(person);
            contactsNew = em.merge(contactsNew);
        }
        for (PersonAttendance personAttendanceListOldPersonAttendance : personAttendanceListOld) {
            if (!personAttendanceListNew.contains(personAttendanceListOldPersonAttendance)) {
                personAttendanceListOldPersonAttendance.setPerson(null);
                personAttendanceListOldPersonAttendance =
em.merge(personAttendanceListOldPersonAttendance);
            }
        }
        for (PersonAttendance personAttendanceListNewPersonAttendance : personAttendanceListNew) {
            if (!personAttendanceListOld.contains(personAttendanceListNewPersonAttendance)) {
                Person oldPersonOfPersonAttendanceListNewPersonAttendance =
personAttendanceListNewPersonAttendance.getPerson();
                personAttendanceListNewPersonAttendance.setPerson(person);
                personAttendanceListNewPersonAttendance =
em.merge(personAttendanceListNewPersonAttendance);
                if (oldPersonOfPersonAttendanceListNewPersonAttendance != null &&
!oldPersonOfPersonAttendanceListNewPersonAttendance.equals(person)) {

```

```

oldPersonOfPersonAttendanceListNewPersonAttendance.getPersonAttendanceList().remove(personAttendanceListNewPersonAttendance);
        oldPersonOfPersonAttendanceListNewPersonAttendance =
em.merge(oldPersonOfPersonAttendanceListNewPersonAttendance);
    }
}
}
for (PersonPayments personPaymentsListOldPersonPayments : personPaymentsListOld) {
    if (!personPaymentsListNew.contains(personPaymentsListOldPersonPayments)) {
        personPaymentsListOldPersonPayments.setPerson(null);
        personPaymentsListOldPersonPayments =
em.merge(personPaymentsListOldPersonPayments);
    }
}
for (PersonPayments personPaymentsListNewPersonPayments : personPaymentsListNew) {
    if (!personPaymentsListOld.contains(personPaymentsListNewPersonPayments)) {
        Person oldPersonOfPersonPaymentsListNewPersonPayments =
personPaymentsListNewPersonPayments.getPerson();
        personPaymentsListNewPersonPayments.setPerson(person);
        personPaymentsListNewPersonPayments =
em.merge(personPaymentsListNewPersonPayments);
        if (oldPersonOfPersonPaymentsListNewPersonPayments != null &&
!oldPersonOfPersonPaymentsListNewPersonPayments.equals(person)) {

oldPersonOfPersonPaymentsListNewPersonPayments.getPersonPaymentsList().remove(personPaymentsListNewPersonPayments);
        oldPersonOfPersonPaymentsListNewPersonPayments =
em.merge(oldPersonOfPersonPaymentsListNewPersonPayments);
    }
}
}
em.getTransaction().commit();
} catch (Exception ex) {
    String msg = ex.getLocalizedMessage();
    if (msg == null || msg.length() == 0) {
        Integer id = person.getPersonID();
        if (findPerson(id) == null) {
            throw new NonexistentEntityException("The person with id " + id + " no longer exists.");
        }
    }
    throw ex;
} finally {
    if (em != null) {
        em.close();
    }
}
}

public void destroy(Integer id) throws NonexistentEntityException {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        Person person;
        try {

```

```

        person = em.getReference(Person.class, id);
        person.getPersonID();
    } catch (EntityNotFoundException enfe) {
        throw new NonexistentEntityException("The person with id " + id + " no longer exists.", enfe);
    }
    Users users = person.getUsers();
    if (users != null) {
        users.getPersonList().remove(person);
        users = em.merge(users);
    }
    Contacts contacts = person.getContacts();
    if (contacts != null) {
        contacts.setPerson(null);
        contacts = em.merge(contacts);
    }
    List<PersonAttendance> personAttendanceList = person.getPersonAttendanceList();
    for (PersonAttendance personAttendanceListPersonAttendance : personAttendanceList) {
        personAttendanceListPersonAttendance.setPerson(null);
        personAttendanceListPersonAttendance = em.merge(personAttendanceListPersonAttendance);
    }
    List<PersonPayments> personPaymentsList = person.getPersonPaymentsList();
    for (PersonPayments personPaymentsListPersonPayments : personPaymentsList) {
        personPaymentsListPersonPayments.setPerson(null);
        personPaymentsListPersonPayments = em.merge(personPaymentsListPersonPayments);
    }
    em.remove(person);
    em.getTransaction().commit();
} finally {
    if (em != null) {
        em.close();
    }
}
}

public List<Person> findPersonEntities() {
    return findPersonEntities(true, -1, -1);
}

public List<Person> findPersonEntities(int maxResults, int firstResult) {
    return findPersonEntities(false, maxResults, firstResult);
}

private List<Person> findPersonEntities(boolean all, int maxResults, int firstResult) {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(Person.class));
        Query q = em.createQuery(cq);
        if (!all) {
            q.setMaxResults(maxResults);
            q.setFirstResult(firstResult);
        }
        return q.getResultList();
    } finally {
        em.close();
    }
}

```

```

    }

    public Person findPerson(Integer id) {
        EntityManager em = getEntityManager();
        try {
            return em.find(Person.class, id);
        } finally {
            em.close();
        }
    }

    public int getPersonCount() {
        EntityManager em = getEntityManager();
        try {
            CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
            Root<Person> rt = cq.from(Person.class);
            cq.select(em.getCriteriaBuilder().count(rt));
            Query q = em.createQuery(cq);
            return ((Long) q.getSingleResult()).intValue();
        } finally {
            em.close();
        }
    }
}

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package erp.controller;

import erp.controller.exceptions.IllegalOrphanException;
import erp.controller.exceptions.NonexistentEntityException;
import erp.controller.exceptions.PreexistingEntityException;
import erp.model.Products;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.persistence.EntityNotFoundException;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import erp.model.GRNLine;
import java.util.ArrayList;
import java.util.List;
import erp.model.PurchasOrdersLines;
import erp.model.ProductSuppliers;
import erp.model.ProductBinCard;
import erp.model.ProductIngredients;
import erp.model.SalesOrderLines;

/**
 *

```

```

* @author Administrator
*/
public class ProductsJpaController {

    public ProductsJpaController() {
        emf = Persistence.createEntityManagerFactory("ERPSPU");
    }
    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    public void create(Products products) throws PreexistingEntityException, Exception {
        if (products.getGRNLineList() == null) {
            products.setGRNLineList(new ArrayList<GRNLine>());
        }
        if (products.getPurchasOrdersLinesList() == null) {
            products.setPurchasOrdersLinesList(new ArrayList<PurchasOrdersLines>());
        }
        if (products.getProductSuppliersList() == null) {
            products.setProductSuppliersList(new ArrayList<ProductSuppliers>());
        }
        if (products.getProductBinCardList() == null) {
            products.setProductBinCardList(new ArrayList<ProductBinCard>());
        }
        if (products.getProductIngredientsList() == null) {
            products.setProductIngredientsList(new ArrayList<ProductIngredients>());
        }
        if (products.getProductIngredientsListI() == null) {
            products.setProductIngredientsListI(new ArrayList<ProductIngredients>());
        }
        if (products.getSalesOrderLinesList() == null) {
            products.setSalesOrderLinesList(new ArrayList<SalesOrderLines>());
        }
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            List<GRNLine> attachedGRNLineList = new ArrayList<GRNLine>();
            for (GRNLine GRNLineListGRNLineToAttach : products.getGRNLineList()) {
                GRNLineListGRNLineToAttach = em.getReference(GRNLineListGRNLineToAttach.getClass(),
GRNLineListGRNLineToAttach.getId());
                attachedGRNLineList.add(GRNLineListGRNLineToAttach);
            }
            products.setGRNLineList(attachedGRNLineList);
            List<PurchasOrdersLines> attachedPurchasOrdersLinesList = new
ArrayList<PurchasOrdersLines>();
            for (PurchasOrdersLines purchasOrdersLinesListPurchasOrdersLinesToAttach :
products.getPurchasOrdersLinesList()) {
                purchasOrdersLinesListPurchasOrdersLinesToAttach =
em.getReference(purchasOrdersLinesListPurchasOrdersLinesToAttach.getClass(),
purchasOrdersLinesListPurchasOrdersLinesToAttach.getId());
                attachedPurchasOrdersLinesList.add(purchasOrdersLinesListPurchasOrdersLinesToAttach);
            }
            products.setPurchasOrdersLinesList(attachedPurchasOrdersLinesList);
        }
    }
}

```

```

        List<ProductSuppliers> attachedProductSuppliersList = new ArrayList<ProductSuppliers>();
        for (ProductSuppliers productSuppliersListProductSuppliersToAttach :
products.getProductSuppliersList()) {
            productSuppliersListProductSuppliersToAttach =
em.getReference(productSuppliersListProductSuppliersToAttach.getClass(),
productSuppliersListProductSuppliersToAttach.getId());
            attachedProductSuppliersList.add(productSuppliersListProductSuppliersToAttach);
        }
        products.setProductSuppliersList(attachedProductSuppliersList);
        List<ProductBinCard> attachedProductBinCardList = new ArrayList<ProductBinCard>();
        for (ProductBinCard productBinCardListProductBinCardToAttach :
products.getProductBinCardList()) {
            productBinCardListProductBinCardToAttach =
em.getReference(productBinCardListProductBinCardToAttach.getClass(),
productBinCardListProductBinCardToAttach.getId());
            attachedProductBinCardList.add(productBinCardListProductBinCardToAttach);
        }
        products.setProductBinCardList(attachedProductBinCardList);
        List<ProductIngredients> attachedProductIngredientsList = new ArrayList<ProductIngredients>();
        for (ProductIngredients productIngredientsListProductIngredientsToAttach :
products.getProductIngredientsList()) {
            productIngredientsListProductIngredientsToAttach =
em.getReference(productIngredientsListProductIngredientsToAttach.getClass(),
productIngredientsListProductIngredientsToAttach.getId());
            attachedProductIngredientsList.add(productIngredientsListProductIngredientsToAttach);
        }
        products.setProductIngredientsList(attachedProductIngredientsList);
        List<ProductIngredients> attachedProductIngredientsList1 = new
ArrayList<ProductIngredients>();
        for (ProductIngredients productIngredientsList1ProductIngredientsToAttach :
products.getProductIngredientsList1()) {
            productIngredientsList1ProductIngredientsToAttach =
em.getReference(productIngredientsList1ProductIngredientsToAttach.getClass(),
productIngredientsList1ProductIngredientsToAttach.getId());
            attachedProductIngredientsList1.add(productIngredientsList1ProductIngredientsToAttach);
        }
        products.setProductIngredientsList1(attachedProductIngredientsList1);
        List<SalesOrderLines> attachedSalesOrderLinesList = new ArrayList<SalesOrderLines>();
        for (SalesOrderLines salesOrderLinesListSalesOrderLinesToAttach :
products.getSalesOrderLinesList()) {
            salesOrderLinesListSalesOrderLinesToAttach =
em.getReference(salesOrderLinesListSalesOrderLinesToAttach.getClass(),
salesOrderLinesListSalesOrderLinesToAttach.getId());
            attachedSalesOrderLinesList.add(salesOrderLinesListSalesOrderLinesToAttach);
        }
        products.setSalesOrderLinesList(attachedSalesOrderLinesList);
        em.persist(products);
        for (GRNLine GRNLineListGRNLine : products.getGRNLineList()) {
            Products oldProductsOfGRNLineListGRNLine = GRNLineListGRNLine.getProducts();
            GRNLineListGRNLine.setProducts(products);
            GRNLineListGRNLine = em.merge(GRNLineListGRNLine);
            if (oldProductsOfGRNLineListGRNLine != null) {
                oldProductsOfGRNLineListGRNLine.getGRNLineList().remove(GRNLineListGRNLine);
                oldProductsOfGRNLineListGRNLine = em.merge(oldProductsOfGRNLineListGRNLine);
            }
        }
    }
}

```

```

        for (PurchasOrdersLines purchasOrdersLinesListPurchasOrdersLines :
products.getPurchasOrdersLinesList()) {
            Products oldProductsOfPurchasOrdersLinesListPurchasOrdersLines =
purchasOrdersLinesListPurchasOrdersLines.getProducts();
            purchasOrdersLinesListPurchasOrdersLines.setProducts(products);
            purchasOrdersLinesListPurchasOrdersLines =
em.merge(purchasOrdersLinesListPurchasOrdersLines);
            if (oldProductsOfPurchasOrdersLinesListPurchasOrdersLines != null) {

oldProductsOfPurchasOrdersLinesListPurchasOrdersLines.getPurchasOrdersLinesList().remove(purchasO
rdersLinesListPurchasOrdersLines);
                oldProductsOfPurchasOrdersLinesListPurchasOrdersLines =
em.merge(oldProductsOfPurchasOrdersLinesListPurchasOrdersLines);
            }
        }
        for (ProductSuppliers productSuppliersListProductSuppliers : products.getProductSuppliersList()) {
            Products oldProductsOfProductSuppliersListProductSuppliers =
productSuppliersListProductSuppliers.getProducts();
            productSuppliersListProductSuppliers.setProducts(products);
            productSuppliersListProductSuppliers = em.merge(productSuppliersListProductSuppliers);
            if (oldProductsOfProductSuppliersListProductSuppliers != null) {

oldProductsOfProductSuppliersListProductSuppliers.getProductSuppliersList().remove(productSuppliersLi
stProductSuppliers);
                oldProductsOfProductSuppliersListProductSuppliers =
em.merge(oldProductsOfProductSuppliersListProductSuppliers);
            }
        }
        for (ProductBinCard productBinCardListProductBinCard : products.getProductBinCardList()) {
            Products oldProductsOfProductBinCardListProductBinCard =
productBinCardListProductBinCard.getProducts();
            productBinCardListProductBinCard.setProducts(products);
            productBinCardListProductBinCard = em.merge(productBinCardListProductBinCard);
            if (oldProductsOfProductBinCardListProductBinCard != null) {

oldProductsOfProductBinCardListProductBinCard.getProductBinCardList().remove(productBinCardListPr
oductBinCard);
                oldProductsOfProductBinCardListProductBinCard =
em.merge(oldProductsOfProductBinCardListProductBinCard);
            }
        }
        for (ProductIngredients productIngredientsListProductIngredients :
products.getProductIngredientsList()) {
            Products oldProductsOfProductIngredientsListProductIngredients =
productIngredientsListProductIngredients.getProducts();
            productIngredientsListProductIngredients.setProducts(products);
            productIngredientsListProductIngredients =
em.merge(productIngredientsListProductIngredients);
            if (oldProductsOfProductIngredientsListProductIngredients != null) {

oldProductsOfProductIngredientsListProductIngredients.getProductIngredientsList().remove(productIngre
dientsListProductIngredients);
                oldProductsOfProductIngredientsListProductIngredients =
em.merge(oldProductsOfProductIngredientsListProductIngredients);
            }
        }
    }
}

```

```

        for (ProductIngredients productIngredientsList1ProductIngredients :
products.getProductIngredientsList1()) {
            Products oldProducts1OfProductIngredientsList1ProductIngredients =
productIngredientsList1ProductIngredients.getProduct1();
            productIngredientsList1ProductIngredients.setProducts1(products);
            productIngredientsList1ProductIngredients =
em.merge(productIngredientsList1ProductIngredients);
            if (oldProducts1OfProductIngredientsList1ProductIngredients != null) {

oldProducts1OfProductIngredientsList1ProductIngredients.getProductIngredientsList1().remove(productIn
gredientsList1ProductIngredients);
                oldProducts1OfProductIngredientsList1ProductIngredients =
em.merge(oldProducts1OfProductIngredientsList1ProductIngredients);
            }
        }
        for (SalesOrderLines salesOrderLinesListSalesOrderLines : products.getSalesOrderLinesList()) {
            Products oldProductsOfSalesOrderLinesListSalesOrderLines =
salesOrderLinesListSalesOrderLines.getProduct1();
            salesOrderLinesListSalesOrderLines.setProducts(products);
            salesOrderLinesListSalesOrderLines = em.merge(salesOrderLinesListSalesOrderLines);
            if (oldProductsOfSalesOrderLinesListSalesOrderLines != null) {

oldProductsOfSalesOrderLinesListSalesOrderLines.getSalesOrderLinesList().remove(salesOrderLinesListS
alesOrderLines);
                oldProductsOfSalesOrderLinesListSalesOrderLines =
em.merge(oldProductsOfSalesOrderLinesListSalesOrderLines);
            }
        }
        em.getTransaction().commit();
    } catch (Exception ex) {
        if (findProducts(products.getProductID()) != null) {
            throw new PreexistingEntityException("Products " + products + " already exists.", ex);
        }
        throw ex;
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

```

```

public void edit(Products products) throws IllegalOrphanException, NonexistentEntityException,
Exception {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        Products persistentProducts = em.find(Products.class, products.getProductID());
        List<GRNLine> GRNLineListOld = persistentProducts.getGRNLineList();
        List<GRNLine> GRNLineListNew = products.getGRNLineList();
        List<PurchasOrdersLines> purchasOrdersLinesListOld =
persistentProducts.getPurchasOrdersLinesList();
        List<PurchasOrdersLines> purchasOrdersLinesListNew = products.getPurchasOrdersLinesList();
        List<ProductSuppliers> productSuppliersListOld = persistentProducts.getProductSuppliersList();
        List<ProductSuppliers> productSuppliersListNew = products.getProductSuppliersList();
        List<ProductBinCard> productBinCardListOld = persistentProducts.getProductBinCardList();
    }
}

```



```

        List<ProductBinCard> productBinCardListNew = products.getProductBinCardList();
        List<ProductIngredients> productIngredientsListOld =
persistentProducts.getProductIngredientsList();
        List<ProductIngredients> productIngredientsListNew = products.getProductIngredientsList();
        List<ProductIngredients> productIngredientsList1Old =
persistentProducts.getProductIngredientsList1();
        List<ProductIngredients> productIngredientsList1New = products.getProductIngredientsList1();
        List<SalesOrderLines> salesOrderLinesListOld = persistentProducts.getSalesOrderLinesList();
        List<SalesOrderLines> salesOrderLinesListNew = products.getSalesOrderLinesList();
        List<String> illegalOrphanMessages = null;
        for (GRNLine GRNLineListOldGRNLine : GRNLineListOld) {
            if (!GRNLineListNew.contains(GRNLineListOldGRNLine)) {
                if (illegalOrphanMessages == null) {
                    illegalOrphanMessages = new ArrayList<String>();
                }
                illegalOrphanMessages.add("You must retain GRNLine " + GRNLineListOldGRNLine + "
since its products field is not nullable.");
            }
        }
        for (PurchasOrdersLines purchasOrdersLinesListOldPurchasOrdersLines :
purchasOrdersLinesListOld) {
            if (!purchasOrdersLinesListNew.contains(purchasOrdersLinesListOldPurchasOrdersLines)) {
                if (illegalOrphanMessages == null) {
                    illegalOrphanMessages = new ArrayList<String>();
                }
                illegalOrphanMessages.add("You must retain PurchasOrdersLines " +
purchasOrdersLinesListOldPurchasOrdersLines + " since its products field is not nullable.");
            }
        }
        for (SalesOrderLines salesOrderLinesListOldSalesOrderLines : salesOrderLinesListOld) {
            if (!salesOrderLinesListNew.contains(salesOrderLinesListOldSalesOrderLines)) {
                if (illegalOrphanMessages == null) {
                    illegalOrphanMessages = new ArrayList<String>();
                }
                illegalOrphanMessages.add("You must retain SalesOrderLines " +
salesOrderLinesListOldSalesOrderLines + " since its products field is not nullable.");
            }
        }
        if (illegalOrphanMessages != null) {
            throw new IllegalOrphanException(illegalOrphanMessages);
        }
        List<GRNLine> attachedGRNLineListNew = new ArrayList<GRNLine>();
        for (GRNLine GRNLineListNewGRNLineToAttach : GRNLineListNew) {
            GRNLineListNewGRNLineToAttach =
em.getReference(GRNLineListNewGRNLineToAttach.getClass(),
GRNLineListNewGRNLineToAttach.getId());
            attachedGRNLineListNew.add(GRNLineListNewGRNLineToAttach);
        }
        GRNLineListNew = attachedGRNLineListNew;
        products.setGRNLineList(GRNLineListNew);
        List<PurchasOrdersLines> attachedPurchasOrdersLinesListNew = new
ArrayList<PurchasOrdersLines>();
        for (PurchasOrdersLines purchasOrdersLinesListNewPurchasOrdersLinesToAttach :
purchasOrdersLinesListNew) {

```

```

        purchasOrdersLinesListNewPurchasOrdersLinesToAttach =
em.getReference(purchasOrdersLinesListNewPurchasOrdersLinesToAttach.getClass(),
purchasOrdersLinesListNewPurchasOrdersLinesToAttach.getId());

attachedPurchasOrdersLinesListNew.add(purchasOrdersLinesListNewPurchasOrdersLinesToAttach);
    }
    purchasOrdersLinesListNew = attachedPurchasOrdersLinesListNew;
    products.setPurchasOrdersLinesList(purchasOrdersLinesListNew);
    List<ProductSuppliers> attachedProductSuppliersListNew = new ArrayList<ProductSuppliers>();
    for (ProductSuppliers productSuppliersListNewProductSuppliersToAttach :
productSuppliersListNew) {
        productSuppliersListNewProductSuppliersToAttach =
em.getReference(productSuppliersListNewProductSuppliersToAttach.getClass(),
productSuppliersListNewProductSuppliersToAttach.getId());
        attachedProductSuppliersListNew.add(productSuppliersListNewProductSuppliersToAttach);
    }
    productSuppliersListNew = attachedProductSuppliersListNew;
    products.setProductSuppliersList(productSuppliersListNew);
    List<ProductBinCard> attachedProductBinCardListNew = new ArrayList<ProductBinCard>();
    for (ProductBinCard productBinCardListNewProductBinCardToAttach : productBinCardListNew) {
        productBinCardListNewProductBinCardToAttach =
em.getReference(productBinCardListNewProductBinCardToAttach.getClass(),
productBinCardListNewProductBinCardToAttach.getId());
        attachedProductBinCardListNew.add(productBinCardListNewProductBinCardToAttach);
    }
    productBinCardListNew = attachedProductBinCardListNew;
    products.setProductBinCardList(productBinCardListNew);
    List<ProductIngredients> attachedProductIngredientsListNew = new
ArrayList<ProductIngredients>();
    for (ProductIngredients productIngredientsListNewProductIngredientsToAttach :
productIngredientsListNew) {
        productIngredientsListNewProductIngredientsToAttach =
em.getReference(productIngredientsListNewProductIngredientsToAttach.getClass(),
productIngredientsListNewProductIngredientsToAttach.getId());
        attachedProductIngredientsListNew.add(productIngredientsListNewProductIngredientsToAttach);
    }
    productIngredientsListNew = attachedProductIngredientsListNew;
    products.setProductIngredientsList(productIngredientsListNew);
    List<ProductIngredients> attachedProductIngredientsList1New = new
ArrayList<ProductIngredients>();
    for (ProductIngredients productIngredientsList1NewProductIngredientsToAttach :
productIngredientsList1New) {
        productIngredientsList1NewProductIngredientsToAttach =
em.getReference(productIngredientsList1NewProductIngredientsToAttach.getClass(),
productIngredientsList1NewProductIngredientsToAttach.getId());
        attachedProductIngredientsList1New.add(productIngredientsList1NewProductIngredientsToAttach);
    }
    productIngredientsList1New = attachedProductIngredientsList1New;
    products.setProductIngredientsList1(productIngredientsList1New);
    List<SalesOrderLines> attachedSalesOrderLinesListNew = new ArrayList<SalesOrderLines>();
    for (SalesOrderLines salesOrderLinesListNewSalesOrderLinesToAttach : salesOrderLinesListNew)
    {
        salesOrderLinesListNewSalesOrderLinesToAttach =
em.getReference(salesOrderLinesListNewSalesOrderLinesToAttach.getClass(),
salesOrderLinesListNewSalesOrderLinesToAttach.getId());

```

```

        attachedSalesOrderLinesListNew.add(salesOrderLinesListNewSalesOrderLinesToAttach);
    }
    salesOrderLinesListNew = attachedSalesOrderLinesListNew;
    products.setSalesOrderLinesList(salesOrderLinesListNew);
    products = em.merge(products);
    for (GRNLine GRNLineListNewGRNLine : GRNLineListNew) {
        if (!GRNLineListOld.contains(GRNLineListNewGRNLine)) {
            Products oldProductsOfGRNLineListNewGRNLine =
GRNLineListNewGRNLine.getProducts();
            GRNLineListNewGRNLine.setProducts(products);
            GRNLineListNewGRNLine = em.merge(GRNLineListNewGRNLine);
            if (oldProductsOfGRNLineListNewGRNLine != null &&
!oldProductsOfGRNLineListNewGRNLine.equals(products)) {

oldProductsOfGRNLineListNewGRNLine.getGRNLineList().remove(GRNLineListNewGRNLine);
                oldProductsOfGRNLineListNewGRNLine =
em.merge(oldProductsOfGRNLineListNewGRNLine);
            }
        }
    }
    for (PurchasOrdersLines purchasOrdersLinesListNewPurchasOrdersLines :
purchasOrdersLinesListNew) {
        if (!purchasOrdersLinesListOld.contains(purchasOrdersLinesListNewPurchasOrdersLines)) {
            Products oldProductsOfPurchasOrdersLinesListNewPurchasOrdersLines =
purchasOrdersLinesListNewPurchasOrdersLines.getProducts();
            purchasOrdersLinesListNewPurchasOrdersLines.setProducts(products);
            purchasOrdersLinesListNewPurchasOrdersLines =
em.merge(purchasOrdersLinesListNewPurchasOrdersLines);
            if (oldProductsOfPurchasOrdersLinesListNewPurchasOrdersLines != null &&
!oldProductsOfPurchasOrdersLinesListNewPurchasOrdersLines.equals(products)) {

oldProductsOfPurchasOrdersLinesListNewPurchasOrdersLines.getPurchasOrdersLinesList().remove(purc
hasOrdersLinesListNewPurchasOrdersLines);
                oldProductsOfPurchasOrdersLinesListNewPurchasOrdersLines =
em.merge(oldProductsOfPurchasOrdersLinesListNewPurchasOrdersLines);
            }
        }
    }
    for (ProductSuppliers productSuppliersListOldProductSuppliers : productSuppliersListOld) {
        if (!productSuppliersListNew.contains(productSuppliersListOldProductSuppliers)) {
            productSuppliersListOldProductSuppliers.setProducts(null);
            productSuppliersListOldProductSuppliers =
em.merge(productSuppliersListOldProductSuppliers);
        }
    }
    for (ProductSuppliers productSuppliersListNewProductSuppliers : productSuppliersListNew) {
        if (!productSuppliersListOld.contains(productSuppliersListNewProductSuppliers)) {
            Products oldProductsOfProductSuppliersListNewProductSuppliers =
productSuppliersListNewProductSuppliers.getProducts();
            productSuppliersListNewProductSuppliers.setProducts(products);
            productSuppliersListNewProductSuppliers =
em.merge(productSuppliersListNewProductSuppliers);
            if (oldProductsOfProductSuppliersListNewProductSuppliers != null &&
!oldProductsOfProductSuppliersListNewProductSuppliers.equals(products)) {

```

```

oldProductsOfProductSuppliersListNewProductSuppliers.getProductSuppliersList().remove(productSuppliersListNewProductSuppliers);
    oldProductsOfProductSuppliersListNewProductSuppliers =
em.merge(oldProductsOfProductSuppliersListNewProductSuppliers);
    }
    }
    }
    for (ProductBinCard productBinCardListOldProductBinCard : productBinCardListOld) {
        if (!productBinCardListNew.contains(productBinCardListOldProductBinCard)) {
            productBinCardListOldProductBinCard.setProducts(null);
            productBinCardListOldProductBinCard = em.merge(productBinCardListOldProductBinCard);
        }
    }
    for (ProductBinCard productBinCardListNewProductBinCard : productBinCardListNew) {
        if (!productBinCardListOld.contains(productBinCardListNewProductBinCard)) {
            Products oldProductsOfProductBinCardListNewProductBinCard =
productBinCardListNewProductBinCard.getProducts();
            productBinCardListNewProductBinCard.setProducts(products);
            productBinCardListNewProductBinCard =
em.merge(productBinCardListNewProductBinCard);
            if (oldProductsOfProductBinCardListNewProductBinCard != null &&
!oldProductsOfProductBinCardListNewProductBinCard.equals(products)) {

oldProductsOfProductBinCardListNewProductBinCard.getProductBinCardList().remove(productBinCardListNewProductBinCard);
            oldProductsOfProductBinCardListNewProductBinCard =
em.merge(oldProductsOfProductBinCardListNewProductBinCard);
            }
        }
    }
    for (ProductIngredients productIngredientsListOldProductIngredients : productIngredientsListOld)
    {
        if (!productIngredientsListNew.contains(productIngredientsListOldProductIngredients)) {
            productIngredientsListOldProductIngredients.setProducts(null);
            productIngredientsListOldProductIngredients =
em.merge(productIngredientsListOldProductIngredients);
        }
    }
    for (ProductIngredients productIngredientsListNewProductIngredients : productIngredientsListNew)
    {
        if (!productIngredientsListOld.contains(productIngredientsListNewProductIngredients)) {
            Products oldProductsOfProductIngredientsListNewProductIngredients =
productIngredientsListNewProductIngredients.getProducts();
            productIngredientsListNewProductIngredients.setProducts(products);
            productIngredientsListNewProductIngredients =
em.merge(productIngredientsListNewProductIngredients);
            if (oldProductsOfProductIngredientsListNewProductIngredients != null &&
!oldProductsOfProductIngredientsListNewProductIngredients.equals(products)) {

oldProductsOfProductIngredientsListNewProductIngredients.getProductIngredientsList().remove(productIngredientsListNewProductIngredients);
            oldProductsOfProductIngredientsListNewProductIngredients =
em.merge(oldProductsOfProductIngredientsListNewProductIngredients);
            }
        }
    }

```

```

    }
    for (ProductIngredients productIngredientsList1OldProductIngredients :
productIngredientsList1Old) {
        if (!productIngredientsList1New.contains(productIngredientsList1OldProductIngredients)) {
            productIngredientsList1OldProductIngredients.setProducts1(null);
            productIngredientsList1OldProductIngredients =
em.merge(productIngredientsList1OldProductIngredients);
        }
    }
    for (ProductIngredients productIngredientsList1NewProductIngredients :
productIngredientsList1New) {
        if (!productIngredientsList1Old.contains(productIngredientsList1NewProductIngredients)) {
            Products oldProducts1OfProductIngredientsList1NewProductIngredients =
productIngredientsList1NewProductIngredients.getProducts1();
            productIngredientsList1NewProductIngredients.setProducts1(products);
            productIngredientsList1NewProductIngredients =
em.merge(productIngredientsList1NewProductIngredients);
            if (oldProducts1OfProductIngredientsList1NewProductIngredients != null &&
!oldProducts1OfProductIngredientsList1NewProductIngredients.equals(products)) {

oldProducts1OfProductIngredientsList1NewProductIngredients.getProductIngredientsList1().remove(produ
ctIngredientsList1NewProductIngredients);
                oldProducts1OfProductIngredientsList1NewProductIngredients =
em.merge(oldProducts1OfProductIngredientsList1NewProductIngredients);
            }
        }
    }
    for (SalesOrderLines salesOrderLinesListNewSalesOrderLines : salesOrderLinesListNew) {
        if (!salesOrderLinesListOld.contains(salesOrderLinesListNewSalesOrderLines)) {
            Products oldProductsOfSalesOrderLinesListNewSalesOrderLines =
salesOrderLinesListNewSalesOrderLines.getProducts();
            salesOrderLinesListNewSalesOrderLines.setProducts(products);
            salesOrderLinesListNewSalesOrderLines =
em.merge(salesOrderLinesListNewSalesOrderLines);
            if (oldProductsOfSalesOrderLinesListNewSalesOrderLines != null &&
!oldProductsOfSalesOrderLinesListNewSalesOrderLines.equals(products)) {

oldProductsOfSalesOrderLinesListNewSalesOrderLines.getSalesOrderLinesList().remove(salesOrderLines
ListNewSalesOrderLines);
                oldProductsOfSalesOrderLinesListNewSalesOrderLines =
em.merge(oldProductsOfSalesOrderLinesListNewSalesOrderLines);
            }
        }
    }
    em.getTransaction().commit();
} catch (Exception ex) {
    String msg = ex.getMessage();
    if (msg == null || msg.length() == 0) {
        Integer id = products.getProductID();
        if (findProducts(id) == null) {
            throw new NonexistentEntityException("The products with id " + id + " no longer exists.");
        }
    }
    throw ex;
} finally {
    if (em != null) {

```

```

        em.close();
    }
}
}

public void destroy(Integer id) throws IllegalOrphanException, NonexistentEntityException {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        Products products;
        try {
            products = em.getReference(Products.class, id);
            products.getProductID();
        } catch (EntityNotFoundException enfe) {
            throw new NonexistentEntityException("The products with id " + id + " no longer exists.", enfe);
        }
        List<String> illegalOrphanMessages = null;
        List<GRNLine> GRNLineListOrphanCheck = products.getGRNLineList();
        for (GRNLine GRNLineListOrphanCheckGRNLine : GRNLineListOrphanCheck) {
            if (illegalOrphanMessages == null) {
                illegalOrphanMessages = new ArrayList<String>();
            }
            illegalOrphanMessages.add("This Products (" + products + ") cannot be destroyed since the GRNLine " + GRNLineListOrphanCheckGRNLine + " in its GRNLineList field has a non-nullable products field.");
        }
        List<PurchasOrdersLines> purchasOrdersLinesListOrphanCheck = products.getPurchasOrdersLinesList();
        for (PurchasOrdersLines purchasOrdersLinesListOrphanCheckPurchasOrdersLines : purchasOrdersLinesListOrphanCheck) {
            if (illegalOrphanMessages == null) {
                illegalOrphanMessages = new ArrayList<String>();
            }
            illegalOrphanMessages.add("This Products (" + products + ") cannot be destroyed since the PurchasOrdersLines " + purchasOrdersLinesListOrphanCheckPurchasOrdersLines + " in its purchasOrdersLinesList field has a non-nullable products field.");
        }
        List<SalesOrderLines> salesOrderLinesListOrphanCheck = products.getSalesOrderLinesList();
        for (SalesOrderLines salesOrderLinesListOrphanCheckSalesOrderLines : salesOrderLinesListOrphanCheck) {
            if (illegalOrphanMessages == null) {
                illegalOrphanMessages = new ArrayList<String>();
            }
            illegalOrphanMessages.add("This Products (" + products + ") cannot be destroyed since the SalesOrderLines " + salesOrderLinesListOrphanCheckSalesOrderLines + " in its salesOrderLinesList field has a non-nullable products field.");
        }
        if (illegalOrphanMessages != null) {
            throw new IllegalOrphanException(illegalOrphanMessages);
        }
        List<ProductSuppliers> productSuppliersList = products.getProductSuppliersList();
        for (ProductSuppliers productSuppliersListProductSuppliers : productSuppliersList) {
            productSuppliersListProductSuppliers.setProducts(null);
            productSuppliersListProductSuppliers = em.merge(productSuppliersListProductSuppliers);
        }
    }
}

```

```

        List<ProductBinCard> productBinCardList = products.getProductBinCardList();
        for (ProductBinCard productBinCardListProductBinCard : productBinCardList) {
            productBinCardListProductBinCard.setProducts(null);
            productBinCardListProductBinCard = em.merge(productBinCardListProductBinCard);
        }
        List<ProductIngredients> productIngredientsList = products.getProductIngredientsList();
        for (ProductIngredients productIngredientsListProductIngredients : productIngredientsList) {
            productIngredientsListProductIngredients.setProducts(null);
            productIngredientsListProductIngredients =
em.merge(productIngredientsListProductIngredients);
        }
        List<ProductIngredients> productIngredientsList1 = products.getProductIngredientsList1();
        for (ProductIngredients productIngredientsList1ProductIngredients : productIngredientsList1) {
            productIngredientsList1ProductIngredients.setProducts1(null);
            productIngredientsList1ProductIngredients =
em.merge(productIngredientsList1ProductIngredients);
        }
        em.remove(products);
        em.getTransaction().commit();
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public List<Products> findProductsEntities() {
    return findProductsEntities(true, -1, -1);
}

public List<Products> findProductsEntities(int maxResults, int firstResult) {
    return findProductsEntities(false, maxResults, firstResult);
}

private List<Products> findProductsEntities(boolean all, int maxResults, int firstResult) {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(Products.class));
        Query q = em.createQuery(cq);
        if (!all) {
            q.setMaxResults(maxResults);
            q.setFirstResult(firstResult);
        }
        return q.getResultList();
    } finally {
        em.close();
    }
}

public Products findProducts(Integer id) {
    EntityManager em = getEntityManager();
    try {
        return em.find(Products.class, id);
    } finally {
        em.close();
    }
}

```

```

    }
}

public int getProductsCount() {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        Root<Products> rt = cq.from(Products.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    } finally {
        em.close();
    }
}
}
}

```

C.3. DATABASE CREATION SCRIPT

```

SET client_encoding = 'UTF8';
SET standard_conforming_strings = off;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET escape_string_warning = off;

CREATE PROCEDURAL LANGUAGE plpgsql;
ALTER PROCEDURAL LANGUAGE plpgsql OWNER TO postgres;
SET search_path = public, pg_catalog;
SET default_tablespace = '';
SET default_with_oids = false;
CREATE TABLE "tblContacts" (
    "ContactID" integer NOT NULL,
    "Adress" character(500),
    "Mobile" character(10),
    "Telephone" character(10),
    "City" character(20),
    "Contry" character(20),
    "Email" character(200)
);

ALTER TABLE public."tblContacts" OWNER TO postgres;
CREATE TABLE "tblCustomer" (
    "CustomerID" integer NOT NULL,
    "CustomerName" character(200),
    "ContactID" integer
);

ALTER TABLE public."tblCustomer" OWNER TO postgres;

CREATE TABLE "tblGRN" (
    "GRNID" integer NOT NULL,
    "SupplierID" integer,
    "GRNDate" date,
    "CreatedDatetime" timestamp without time zone DEFAULT now(),
    "CreatedBy" integer,
    "GRNTotal" numeric(10,2) NOT NULL
);

```



```

ALTER TABLE public."tblGRN" OWNER TO postgres;

CREATE TABLE "tblGRNLine" (
    "ID" integer NOT NULL,
    "GRNID" integer NOT NULL,
    "ProductID" integer NOT NULL,
    "ExpectedQTY" integer,
    "RQTY" integer,
    "CreatedDatetime" timestamp without time zone DEFAULT now(),
    "EachPrice" numeric(10,2),
    "TotalPrice" numeric(10,2)
);

ALTER TABLE public."tblGRNLine" OWNER TO postgres;

CREATE TABLE "tblPayrollSettings" (
    "ID" integer NOT NULL,
    "ETF" numeric(10,2),
    "BTF" numeric(10,2),
    "TransportPayments" numeric(10,2),
    "CreatedBy" integer
);

ALTER TABLE public."tblPayrollSettings" OWNER TO postgres;

CREATE TABLE "tblPerson" (
    "FirstName" character(100) NOT NULL,
    "LastName" character(100) NOT NULL,
    "Age" integer NOT NULL,
    "Gender" smallint NOT NULL,
    "BasicSalary" numeric(10,2) NOT NULL,
    "Bday" date NOT NULL,
    "ETF" numeric(10,2),
    "BTF" numeric(10,2),
    "ExtraPayments" numeric(10,2),
    "UpdatedBy" integer,
    "UpdatedDateTime" timestamp without time zone DEFAULT now(),
    "PersonID" integer NOT NULL,
    "ContactID" integer,
    "NoOfLeaves" integer
);

ALTER TABLE public."tblPerson" OWNER TO postgres;

CREATE TABLE "tblPersonAttendance" (
    "ID" integer NOT NULL,
    "PersonID" integer,
    "Came" smallint,
    "DateTime" timestamp without time zone DEFAULT now(),
    "CreatedBy" integer
);

ALTER TABLE public."tblPersonAttendance" OWNER TO postgres;

CREATE TABLE "tblPersonPayments" (
    "Payment" numeric(10,2) NOT NULL,
    "DatePayed" date DEFAULT now(),
    "CreatedBy" integer NOT NULL,
    "CreatedDateTime" timestamp without time zone DEFAULT now(),
    "ID" integer NOT NULL,
    "PersonID" integer
);

```

```

ALTER TABLE public."tblPersonPayments" OWNER TO postgres;

CREATE TABLE "tblProductBinCard" (
    "ID" integer NOT NULL,
    "ProductID" integer,
    "INN" integer,
    "Outt" integer,
    "DateTimeDone" timestamp without time zone DEFAULT now(),
    "DoneBy" integer,
    "Reason" character(200)
);

ALTER TABLE public."tblProductBinCard" OWNER TO postgres;

CREATE TABLE "tblProductIngredients" (
    "ID" integer NOT NULL,
    "ProductID" integer,
    "IngredientsID" integer,
    "NumberOfItems" integer
);

ALTER TABLE public."tblProductIngredients" OWNER TO postgres;

CREATE TABLE "tblProductSuppliers" (
    "ID" integer NOT NULL,
    "ProductID" integer,
    "SupplierID" integer
);

ALTER TABLE public."tblProductSuppliers" OWNER TO postgres;

CREATE TABLE "tblProducts" (
    "ProductID" integer NOT NULL,
    "Code" character(100),
    "Name" character(200),
    "Price" numeric(10,2),
    "IsPurchas" smallint DEFAULT 0,
    "IsProduction" smallint DEFAULT 0,
    "HardwareDesign" bytea,
    "SoftwareDesign" bytea,
    "StockLevel" integer
);

ALTER TABLE public."tblProducts" OWNER TO postgres;

CREATE TABLE "tblPurchasOrders" (
    "purchasOrderID" integer NOT NULL,
    "SupplierID" integer,
    "OrderDate" date,
    "CreatedDateTime" timestamp without time zone DEFAULT now(),
    "purchasTotal" numeric(10,2),
    "CreatedBy" integer,
    "Description" character(200),
    completed smallint DEFAULT 0
);

ALTER TABLE public."tblPurchasOrders" OWNER TO postgres;

CREATE TABLE "tblPurchasOrdersLines" (
    "ID" integer NOT NULL,
    "purchasOrderID" integer NOT NULL,
    "ProductID" integer NOT NULL,

```

```

        "QTY" integer NOT NULL,
        "EachPrice" numeric(10,2),
        "TotalPrice" numeric(10,2) NOT NULL,
        "CreatedDateTime" timestamp without time zone DEFAULT now()
    );

ALTER TABLE public."tblPurchasOrdersLines" OWNER TO postgres;

CREATE TABLE "tblRoles" (
    "RoleName" character(100),
    "RoleID" integer NOT NULL
);

ALTER TABLE public."tblRoles" OWNER TO postgres;

CREATE TABLE "tblSalesOrderLines" (
    "ID" integer NOT NULL,
    "SalesOrderID" integer NOT NULL,
    "ProductID" integer NOT NULL,
    "QTY" integer NOT NULL,
    "EachPrice" numeric(10,2) NOT NULL,
    "TotalPrice" numeric(10,2) NOT NULL,
    "CreatedDateTime" timestamp without time zone DEFAULT now()
);

ALTER TABLE public."tblSalesOrderLines" OWNER TO postgres;

CREATE TABLE "tblSalesOrders" (
    "OrderID" integer NOT NULL,
    "OrderDate" date,
    "CreatedDate" timestamp without time zone DEFAULT now(),
    "CustomertID" integer,
    "SalesTotal" numeric(10,2),
    "CreatedBy" integer,
    "Description" character(100),
    completed smallint DEFAULT 0
);

ALTER TABLE public."tblSalesOrders" OWNER TO postgres;

CREATE TABLE "tblSuppliers" (
    "SupplierName" character(100) NOT NULL,
    "CreatedBy" integer NOT NULL,
    "SupplierID" integer NOT NULL,
    "ContactID" integer
);

ALTER TABLE public."tblSuppliers" OWNER TO postgres;

CREATE TABLE "tblUsers" (
    "UserName" character(20) NOT NULL,
    "Password" character(20) NOT NULL,
    "PersonID" integer,
    "RoleID" smallint NOT NULL,
    "UserID" integer NOT NULL
);

ALTER TABLE public."tblUsers" OWNER TO postgres;

CREATE SEQUENCE "tblContacts_AddressID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE

```

```

        NO MINVALUE
        CACHE 1;

ALTER TABLE public."tblContacts_AddressID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblContacts_AddressID_seq" OWNED BY "tblContacts"."ContactID";

CREATE SEQUENCE "tblCustomer_CustomerID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblCustomer_CustomerID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblCustomer_CustomerID_seq" OWNED BY "tblCustomer"."CustomerID";

CREATE SEQUENCE "tblGRNLine_ID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblGRNLine_ID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblGRNLine_ID_seq" OWNED BY "tblGRNLine"."ID";

CREATE SEQUENCE "tblGRN_GRNID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblGRN_GRNID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblGRN_GRNID_seq" OWNED BY "tblGRN"."GRNID";

CREATE SEQUENCE "tblPayrollSettings_ID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblPayrollSettings_ID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblPayrollSettings_ID_seq" OWNED BY "tblPayrollSettings"."ID";

CREATE SEQUENCE "tblPersonAttendance_ID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblPersonAttendance_ID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblPersonAttendance_ID_seq" OWNED BY "tblPersonAttendance"."ID";

CREATE SEQUENCE "tblPersonPayments_ID_seq"

```

```

        START WITH 1
        INCREMENT BY 1
        NO MAXVALUE
        NO MINVALUE
        CACHE 1;

ALTER TABLE public."tblPersonPayments_ID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblPersonPayments_ID_seq" OWNED BY "tblPersonPayments"."ID";

CREATE SEQUENCE "tblPerson_PersonID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblPerson_PersonID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblPerson_PersonID_seq" OWNED BY "tblPerson"."PersonID";

CREATE SEQUENCE "tblProductBinCard_ID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblProductBinCard_ID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblProductBinCard_ID_seq" OWNED BY "tblProductBinCard"."ID";

CREATE SEQUENCE "tblProductIngredients_ID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblProductIngredients_ID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblProductIngredients_ID_seq" OWNED BY
"tblProductIngredients"."ID";

CREATE SEQUENCE "tblProductSuppliers_ID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblProductSuppliers_ID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblProductSuppliers_ID_seq" OWNED BY "tblProductSuppliers"."ID";

CREATE SEQUENCE "tblProducts_ProductID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

```

```

ALTER TABLE public."tblProducts_ProductID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblProducts_ProductID_seq" OWNED BY "tblProducts"."ProductID";

CREATE SEQUENCE "tblPurchasOrdersLines_ID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblPurchasOrdersLines_ID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblPurchasOrdersLines_ID_seq" OWNED BY
"tblPurchasOrdersLines"."ID";

CREATE SEQUENCE "tblPurchasOrders_purchasOrderID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblPurchasOrders_purchasOrderID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblPurchasOrders_purchasOrderID_seq" OWNED BY
"tblPurchasOrders"."purchasOrderID";

CREATE SEQUENCE "tblRoles_RoleID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblRoles_RoleID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblRoles_RoleID_seq" OWNED BY "tblRoles"."RoleID";

CREATE SEQUENCE "tblSalesOrderLines_ID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblSalesOrderLines_ID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblSalesOrderLines_ID_seq" OWNED BY "tblSalesOrderLines"."ID";

CREATE SEQUENCE "tblSalesOrders_OrderID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblSalesOrders_OrderID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblSalesOrders_OrderID_seq" OWNED BY "tblSalesOrders"."OrderID";

CREATE SEQUENCE "tblSuppliers_SupplierID_seq"
    START WITH 1

```

```

        INCREMENT BY 1
        NO MAXVALUE
        NO MINVALUE
        CACHE 1;

ALTER TABLE public."tblSuppliers_SupplierID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblSuppliers_SupplierID_seq" OWNED BY
"tblSuppliers"."SupplierID";

CREATE SEQUENCE "tblUsers_UserID_seq"
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE public."tblUsers_UserID_seq" OWNER TO postgres;

ALTER SEQUENCE "tblUsers_UserID_seq" OWNED BY "tblUsers"."UserID";

ALTER TABLE "tblContacts" ALTER COLUMN "ContactID" SET DEFAULT
nextval('"tblContacts_AddressID_seq"'::regclass);

ALTER TABLE "tblCustomer" ALTER COLUMN "CustomerID" SET DEFAULT
nextval('"tblCustomer_CustomerID_seq"'::regclass);

ALTER TABLE "tblGRN" ALTER COLUMN "GRNID" SET DEFAULT
nextval('"tblGRN_GRNID_seq"'::regclass);

ALTER TABLE "tblGRNLine" ALTER COLUMN "ID" SET DEFAULT
nextval('"tblGRNLine_ID_seq"'::regclass);

ALTER TABLE "tblPayrollSettings" ALTER COLUMN "ID" SET DEFAULT
nextval('"tblPayrollSettings_ID_seq"'::regclass);

ALTER TABLE "tblPerson" ALTER COLUMN "PersonID" SET DEFAULT
nextval('"tblPerson_PersonID_seq"'::regclass);

ALTER TABLE "tblPersonAttendance" ALTER COLUMN "ID" SET DEFAULT
nextval('"tblPersonAttendance_ID_seq"'::regclass);

ALTER TABLE "tblPersonPayments" ALTER COLUMN "ID" SET DEFAULT
nextval('"tblPersonPayments_ID_seq"'::regclass);

ALTER TABLE "tblProductBinCard" ALTER COLUMN "ID" SET DEFAULT
nextval('"tblProductBinCard_ID_seq"'::regclass);

ALTER TABLE "tblProductIngredients" ALTER COLUMN "ID" SET DEFAULT
nextval('"tblProductIngredients_ID_seq"'::regclass);

ALTER TABLE "tblProductSuppliers" ALTER COLUMN "ID" SET DEFAULT
nextval('"tblProductSuppliers_ID_seq"'::regclass);

ALTER TABLE "tblProducts" ALTER COLUMN "ProductID" SET DEFAULT
nextval('"tblProducts_ProductID_seq"'::regclass);

ALTER TABLE "tblPurchasOrders" ALTER COLUMN "purchasOrderID" SET DEFAULT
nextval('"tblPurchasOrders_purchasOrderID_seq"'::regclass);

ALTER TABLE "tblPurchasOrdersLines" ALTER COLUMN "ID" SET DEFAULT
nextval('"tblPurchasOrdersLines_ID_seq"'::regclass);

```

```

ALTER TABLE "tblRoles" ALTER COLUMN "RoleID" SET DEFAULT
nextval('tblRoles_RoleID_seq'::regclass);

ALTER TABLE "tblSalesOrderLines" ALTER COLUMN "ID" SET DEFAULT
nextval('tblSalesOrderLines_ID_seq'::regclass);

ALTER TABLE "tblSalesOrders" ALTER COLUMN "OrderID" SET DEFAULT
nextval('tblSalesOrders_OrderID_seq'::regclass);

ALTER TABLE "tblSuppliers" ALTER COLUMN "SupplierID" SET DEFAULT
nextval('tblSuppliers_SupplierID_seq'::regclass);

ALTER TABLE "tblUsers" ALTER COLUMN "UserID" SET DEFAULT
nextval('tblUsers_UserID_seq'::regclass);

ALTER TABLE ONLY "tblContacts"
    ADD CONSTRAINT "tblContacts_pkey" PRIMARY KEY ("ContactID");

ALTER TABLE ONLY "tblCustomer"
    ADD CONSTRAINT "tblCustomer_pkey" PRIMARY KEY ("CustomerID");

ALTER TABLE ONLY "tblGRNLine"
    ADD CONSTRAINT "tblGRNLine_pkey" PRIMARY KEY ("ID");

ALTER TABLE ONLY "tblGRN"
    ADD CONSTRAINT "tblGRN_pkey" PRIMARY KEY ("GRNID");

ALTER TABLE ONLY "tblPayrollSettings"
    ADD CONSTRAINT "tblPayrollSettings_pkey" PRIMARY KEY ("ID");

ALTER TABLE ONLY "tblPersonAttendance"
    ADD CONSTRAINT "tblPersonAttendance_pkey" PRIMARY KEY ("ID");

ALTER TABLE ONLY "tblPersonPayments"
    ADD CONSTRAINT "tblPersonPayments_pkey" PRIMARY KEY ("ID");

ALTER TABLE ONLY "tblPerson"
    ADD CONSTRAINT "tblPerson_pkey" PRIMARY KEY ("PersonID");

ALTER TABLE ONLY "tblProductBinCard"
    ADD CONSTRAINT "tblProductBinCard_pkey" PRIMARY KEY ("ID");

ALTER TABLE ONLY "tblProductIngredients"
    ADD CONSTRAINT "tblProductIngredients_pkey" PRIMARY KEY ("ID");

ALTER TABLE ONLY "tblProductSuppliers"
    ADD CONSTRAINT "tblProductSuppliers_pkey" PRIMARY KEY ("ID");

ALTER TABLE ONLY "tblProducts"
    ADD CONSTRAINT "tblProducts_pkey" PRIMARY KEY ("ProductID");

ALTER TABLE ONLY "tblPurchasOrdersLines"
    ADD CONSTRAINT "tblPurchasOrdersLines_pkey" PRIMARY KEY ("ID");

ALTER TABLE ONLY "tblPurchasOrders"
    ADD CONSTRAINT "tblPurchasOrders_pkey" PRIMARY KEY ("purchasOrderID");

ALTER TABLE ONLY "tblRoles"
    ADD CONSTRAINT "tblRoles_pkey" PRIMARY KEY ("RoleID");

ALTER TABLE ONLY "tblSalesOrderLines"
    ADD CONSTRAINT "tblSalesOrderLines_pkey" PRIMARY KEY ("ID");

```



```

ALTER TABLE ONLY "tblSalesOrders"
    ADD CONSTRAINT "tblSalesOrders_pkey" PRIMARY KEY ("OrderID");

ALTER TABLE ONLY "tblSuppliers"
    ADD CONSTRAINT "tblSuppliers_pkey" PRIMARY KEY ("SupplierID");

ALTER TABLE ONLY "tblUsers"
    ADD CONSTRAINT "tblUsers_pkey" PRIMARY KEY ("UserID");

CREATE INDEX fki_ ON "tblUsers" USING btree ("PersonID");

CREATE INDEX "fki_Client" ON "tblSalesOrders" USING btree ("CustomertID");

CREATE INDEX "fki_Contact" ON "tblSuppliers" USING btree ("ContactID");

CREATE INDEX "fki_Contacts" ON "tblCustomer" USING btree ("ContactID");

CREATE INDEX "fki_GR" ON "tblGRNLine" USING btree ("GRNID");

CREATE INDEX "fki_Ingredients" ON "tblProductIngredients" USING btree
("IngredientsID");

CREATE INDEX "fki_PER" ON "tblPersonAttendance" USING btree ("PersonID");

CREATE INDEX "fki_PROD" ON "tblPurchasOrdersLines" USING btree ("ProductID");

CREATE INDEX "fki_PRR" ON "tblProductSuppliers" USING btree ("ProductID");

CREATE INDEX "fki_PRS" ON "tblGRNLine" USING btree ("ProductID");

CREATE INDEX "fki_PUR" ON "tblPurchasOrdersLines" USING btree
("purchasOrderID");

CREATE INDEX "fki_Person" ON "tblPersonPayments" USING btree ("PersonID");

CREATE INDEX "fki_Product" ON "tblProductBinCard" USING btree ("ProductID");

CREATE INDEX "fki_ProductS" ON "tblSalesOrderLines" USING btree ("ProductID");

CREATE INDEX "fki_Products" ON "tblProductIngredients" USING btree
("ProductID");

CREATE INDEX "fki_Role" ON "tblUsers" USING btree ("RoleID");

CREATE INDEX "fki_SUP" ON "tblPurchasOrders" USING btree ("SupplierID");

CREATE INDEX "fki_SUP1" ON "tblGRN" USING btree ("SupplierID");

CREATE INDEX "fki_SUPP" ON "tblProductSuppliers" USING btree ("SupplierID");

CREATE INDEX "fki_Sales" ON "tblSalesOrderLines" USING btree ("SalesOrderID");

CREATE INDEX "fki_U" ON "tblPerson" USING btree ("UpdatedBy");

CREATE INDEX "fki_US" ON "tblPurchasOrders" USING btree ("CreatedBy");

CREATE INDEX "fki_USER" ON "tblPersonAttendance" USING btree ("CreatedBy");

CREATE INDEX "fki_USR" ON "tblGRN" USING btree ("CreatedBy");

CREATE INDEX "fki_USer" ON "tblPersonPayments" USING btree ("CreatedBy");

```

```

CREATE INDEX "fki_USers" ON "tblPayrollSettings" USING btree ("CreatedBy");

CREATE INDEX "fki_User" ON "tblSuppliers" USING btree ("CreatedBy");

CREATE INDEX "fki_UserD" ON "tblProductBinCard" USING btree ("DoneBy");

ALTER TABLE ONLY "tblCustomer"
    ADD CONSTRAINT "tblCustomer_ContactID_fkey" FOREIGN KEY ("ContactID")
    REFERENCES "tblContacts"("ContactID");

ALTER TABLE ONLY "tblGRNLine"
    ADD CONSTRAINT "tblGRNLine_GRNID_fkey" FOREIGN KEY ("GRNID") REFERENCES
    "tblGRN"("GRNID");

ALTER TABLE ONLY "tblGRNLine"
    ADD CONSTRAINT "tblGRNLine_ProductID_fkey" FOREIGN KEY ("ProductID")
    REFERENCES "tblProducts"("ProductID");

ALTER TABLE ONLY "tblGRN"
    ADD CONSTRAINT "tblGRN_CreatedBy_fkey" FOREIGN KEY ("CreatedBy") REFERENCES
    "tblUsers"("UserID");

ALTER TABLE ONLY "tblGRN"
    ADD CONSTRAINT "tblGRN_SupplierID_fkey" FOREIGN KEY ("SupplierID")
    REFERENCES "tblSuppliers"("SupplierID");

ALTER TABLE ONLY "tblPayrollSettings"
    ADD CONSTRAINT "tblPayrollSettings_CreatedBy_fkey" FOREIGN KEY ("CreatedBy")
    REFERENCES "tblUsers"("UserID");

ALTER TABLE ONLY "tblPersonAttendance"
    ADD CONSTRAINT "tblPersonAttendance_CreatedBy_fkey" FOREIGN KEY
    ("CreatedBy") REFERENCES "tblUsers"("UserID");

ALTER TABLE ONLY "tblPersonAttendance"
    ADD CONSTRAINT "tblPersonAttendance_PersonID_fkey" FOREIGN KEY ("PersonID")
    REFERENCES "tblPerson"("PersonID");

ALTER TABLE ONLY "tblPersonPayments"
    ADD CONSTRAINT "tblPersonPayments_CreatedBy_fkey" FOREIGN KEY ("CreatedBy")
    REFERENCES "tblUsers"("UserID");

ALTER TABLE ONLY "tblPersonPayments"
    ADD CONSTRAINT "tblPersonPayments_PersonID_fkey" FOREIGN KEY ("PersonID")
    REFERENCES "tblPerson"("PersonID");

ALTER TABLE ONLY "tblPerson"
    ADD CONSTRAINT "tblPerson_PersonID_fkey" FOREIGN KEY ("PersonID") REFERENCES
    "tblContacts"("ContactID");

ALTER TABLE ONLY "tblPerson"
    ADD CONSTRAINT "tblPerson_UpdatedBy_fkey" FOREIGN KEY ("UpdatedBy")
    REFERENCES "tblUsers"("UserID");

ALTER TABLE ONLY "tblProductBinCard"
    ADD CONSTRAINT "tblProductBinCard_DoneBy_fkey" FOREIGN KEY ("DoneBy")
    REFERENCES "tblUsers"("UserID");

ALTER TABLE ONLY "tblProductBinCard"
    ADD CONSTRAINT "tblProductBinCard_ProductID_fkey" FOREIGN KEY ("ProductID")
    REFERENCES "tblProducts"("ProductID");

ALTER TABLE ONLY "tblProductIngredients"

```

```

        ADD CONSTRAINT "tblProductIngredients_IngredientsID_fkey" FOREIGN KEY
("IngredientsID") REFERENCES "tblProducts"("ProductID");

ALTER TABLE ONLY "tblProductIngredients"
    ADD CONSTRAINT "tblProductIngredients_ProductID_fkey" FOREIGN KEY
("ProductID") REFERENCES "tblProducts"("ProductID");

ALTER TABLE ONLY "tblProductSuppliers"
    ADD CONSTRAINT "tblProductSuppliers_ProductID_fkey" FOREIGN KEY
("ProductID") REFERENCES "tblProducts"("ProductID");

ALTER TABLE ONLY "tblProductSuppliers"
    ADD CONSTRAINT "tblProductSuppliers_SupplierID_fkey" FOREIGN KEY
("SupplierID") REFERENCES "tblSuppliers"("SupplierID");

ALTER TABLE ONLY "tblPurchasOrdersLines"
    ADD CONSTRAINT "tblPurchasOrdersLines_ProductID_fkey" FOREIGN KEY
("ProductID") REFERENCES "tblProducts"("ProductID");

ALTER TABLE ONLY "tblPurchasOrdersLines"
    ADD CONSTRAINT "tblPurchasOrdersLines_purchasOrderID_fkey" FOREIGN KEY
("purchasOrderID") REFERENCES "tblPurchasOrders"("purchasOrderID");

ALTER TABLE ONLY "tblPurchasOrders"
    ADD CONSTRAINT "tblPurchasOrders_CreatedBy_fkey" FOREIGN KEY ("CreatedBy")
REFERENCES "tblUsers"("UserID");

ALTER TABLE ONLY "tblPurchasOrders"
    ADD CONSTRAINT "tblPurchasOrders_SupplierID_fkey" FOREIGN KEY ("SupplierID")
REFERENCES "tblSuppliers"("SupplierID");

ALTER TABLE ONLY "tblSalesOrderLines"
    ADD CONSTRAINT "tblSalesOrderLines_ProductID_fkey" FOREIGN KEY ("ProductID")
REFERENCES "tblProducts"("ProductID");

ALTER TABLE ONLY "tblSalesOrderLines"
    ADD CONSTRAINT "tblSalesOrderLines_SalesOrderID_fkey" FOREIGN KEY
("SalesOrderID") REFERENCES "tblSalesOrders"("OrderID");

ALTER TABLE ONLY "tblSalesOrders"
    ADD CONSTRAINT "tblSalesOrders_ClientID_fkey" FOREIGN KEY ("CustomertID")
REFERENCES "tblCustomer"("CustomerID");

ALTER TABLE ONLY "tblSuppliers"
    ADD CONSTRAINT "tblSuppliers_ContactID_fkey" FOREIGN KEY ("ContactID")
REFERENCES "tblContacts"("ContactID");

ALTER TABLE ONLY "tblSuppliers"
    ADD CONSTRAINT "tblSuppliers_CreatedBy_fkey" FOREIGN KEY ("CreatedBy")
REFERENCES "tblUsers"("UserID");

ALTER TABLE ONLY "tblUsers"
    ADD CONSTRAINT "tblUsers_RoleID_fkey" FOREIGN KEY ("RoleID") REFERENCES
"tblRoles"("RoleID");

REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;

```

APPENDIX D

CLIENT CERTIFICATE



Sanergy (pvt) Ltd,
11/18A, School Avenue,
Mahindarama Rd,
Ethul – Kotte.

Director,
Bachelor of Information Degree Program,
UCSC,
Colombo – 07.
05 – 08 – 2010

Dear Sir,

Confirmation of software development project undertaken by Mr. Chamarawarna Samaraweera

Mr. Chamarawarna Samaraweera was assigned the task of developing an enterprise resource planning system for Synergy (pvt) Ltd. This was undertaken by the above mentioned in fulfillment of the requirements requested under the BIT program.

He has developed a program which meets the needs of the institution and could be used to support the organization as its ERP system adequately fulfilling the planned requirements.

Thank you.

Yours Faithfully,

H.V Ajith

Director Engineering

05-08-2010

SANERGY PRIVATE LTD
11/18A, School Avenue,
Mahindarama Road,
Ethul-Kotte.