## TABLE OF CONTENTS

### CONTENTS

# TITLE OF PROJECT

# BIG BAZAR AUTOMATION SYSTEM

## INTRODUCTION OF THE PROJECT

### BACKGROUND

This software has been developed to fulfill the requirements of **Big Bazar,** a Departmental Store. It takes the franchise of the **Big Bazar.** The most extensively suggested title for the project/software is "**BIG BAZAR AUTOMATION SYSTEM**".

In SUPER MARKET Automation System the valuable information of different products are being stored and managed properly, as per the need of Finance and program implementation individual information of categories wise, such as purchase , sale, purchase return, sale return, report generation. It handles the economic status of each category in monthly and yearly basis. It also maintains the growth progressive by of individual activity such as purchase & sale, employee salary etc.

**This project is a translation of my methodology to achieve following goals:-**

- **Better communication between the various divisions of the Sale System.**
- **Centralized management reporting & decision support.**
- **Timely report of accounting activities.**

- **Effective & controlled handling of the databases related to the transaction.**
- **Made for the organization.**
- **Cost-effective system.**
- **Less Paper Work.**
- **Monthly & Yearly Report generation facility.**

To develop this project we gather the information from the management of local Super Market and its Employees. We collect the information how the work is completed before automated the system. The old system is time consuming. It has no facility to generate the report properly. This project has also facility to see which product is most demand.

# OBJECTIVE OF THE PROJECT

This project has been prepared to fulfill the requirements of super market for "**BIG BAZAR**", keeping in mind the standard rules and regulations as well as the standard data of Super market system that may be similar to another Super Market system. .As far as our project covers all-important details regarding its shop and their respective information. Therefore, in future it will help together such information about Sale & Purchase.

## PURPOSE AND SCOPE:

### 1. PURPOSE:-

#### 1.1 GREATER ACCURACY AND CONSISTENCY:

Carrying out computing steps including arithmetic, correctly and consistency because of Java fully support math co-processor.

#### 1.2 BETTER SECURITY:

Safeguarding sensitive and important by making it accessible only to authorized persons. There are three level securities, one is data level, second is user level and third is administrator level.

### 2. SCOPES:-

This project has been prepared to fulfill the requirements of Super market, keeping in mind the standard rules and regulations as well as the standard data of SUPER MARKET management system that may be similar to another Super market management system. It means having changed slightly, this project can also be used in other Garment Store. As far as our project covers all-important details regarding its shop and their respective information. Therefore, in future it will help together such information about purchase & sale.

- Storing large amount of data for future point of view.

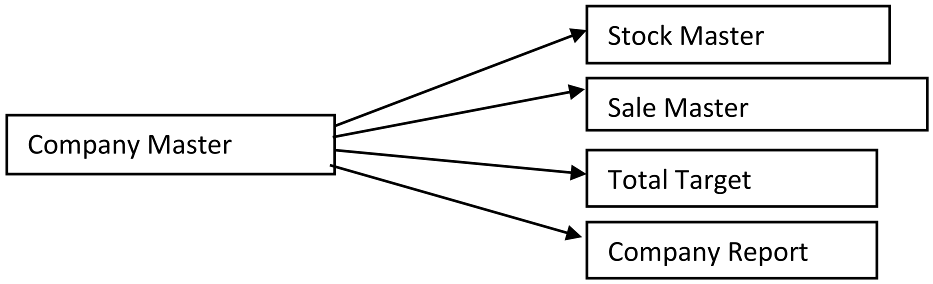
- Reducing manual efforts for maintaining the records.
- Reducing the lead-time.
- It gives correct information about purchase & sale for Super market that has to be further interpreted.

## REQUIREMENTS AND ANALYSIS

### CONCEPTUAL MODELS:

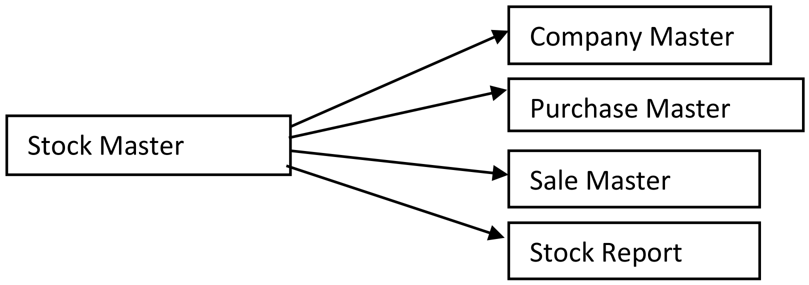
### 1 COMPANY MASTER: -

The module Company Master keeps the information of different company in order to compile the company wise detail information. In our project company information is also store and maintain. We also store the company name, address, transaction etc. The relationship of Company Master is



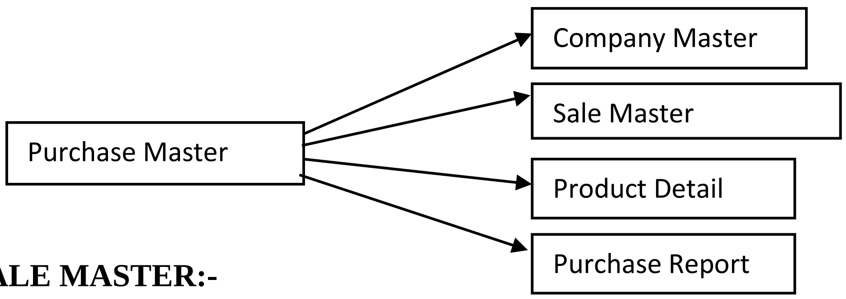
### 2. STOCK MASTER:-

This module Stock Master keeps the information of different garments in order to compile the product wise detail information. In our project mainly maintain the stock information. If we sale the product stock is decrease & purchase the product then product is increase in the Stock. The Relationship of Stock is



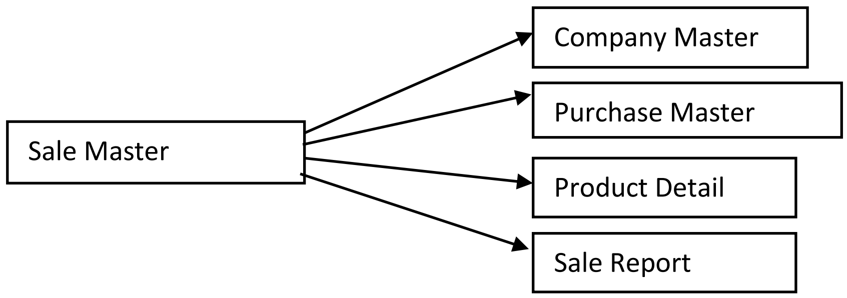
**3. PURCHASE MASTER:-**

The input module Purchase Master keeps the information of purchase garment product in order to compile the product wise detail information. If employee inputs the purchase information product stock is increase.



**4. SALE MASTER:-**

The input module Sale Master keeps the information of sale garment product in order to compile the product wise detail information. If employee inputs the sale information product stock is decrease.



**5. REPORT:-**

This software provides the facility to generate the report easily. There are mainly two types of reports----Monthly Report & Yearly Report. It also provides purchase report, sale report, employee report, purchase return report, sale return report, stock status report etc.

## PROJECT CATEGORY

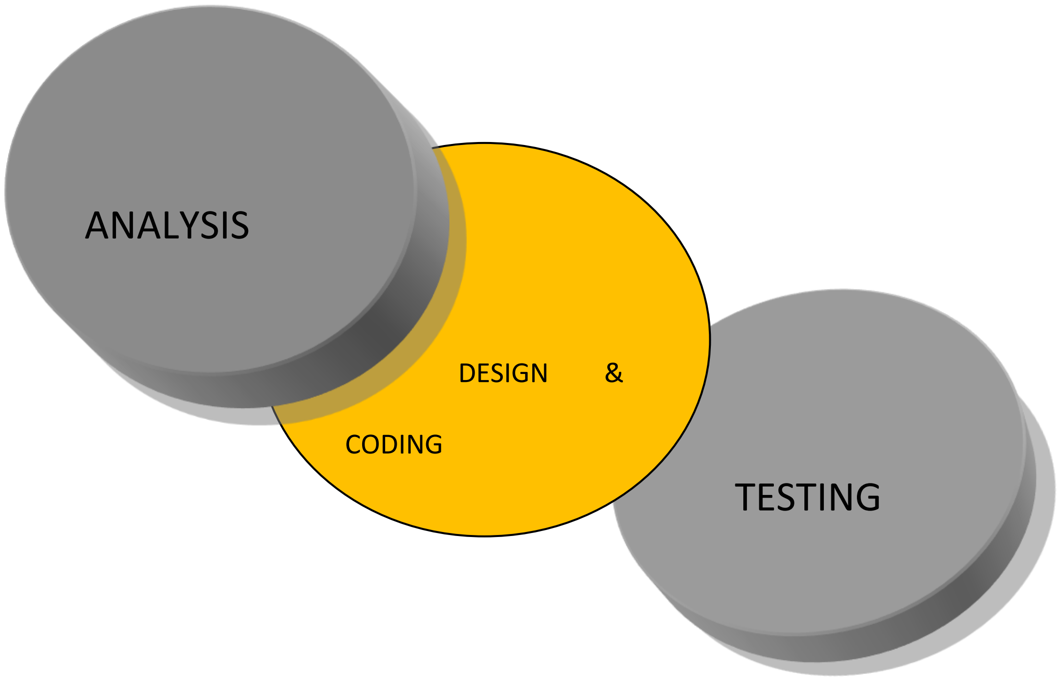
This project belongs to the category of Object Oriented Relational Database Management System, which is based on OORDBMS and is implemented using Oracle9i and Java. In this project I am using Java at the Front End for designing the Graphical User Interface (GUI) which will help the user to work on the software easily. I have chosen Oracle9i at the Back End of my project, in which I will store the information that is being used.

We have selected OORDBMS platform because my project contains various types of information in the form of database and involves several types of reports.

The advantage of using OORDBMS i.e. Oracle extracts in our project.

- Better Security password.
- Minimum Redundancy.
- Removal of Inconsistency.
- Minimizes database anomalies.
- Easy Maintenance of Records.
- Effective Data Structure.

## SOFTWARE DEVELOPMENT LIFE CYCLE

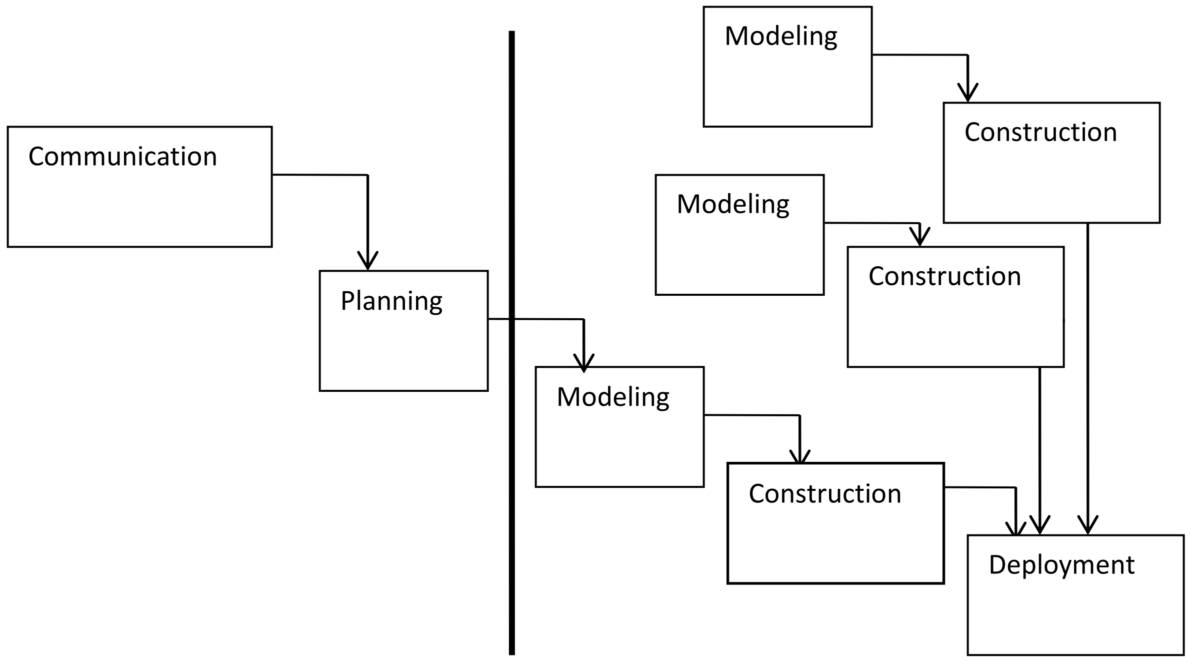


**Analysis as a Bridge between System Engineering and Software Design**

# RAPID APPLICATION DEVELOPMENT MODEL

We choose the RAD (Rapid Application Development) model to develop the project.

It is an incremental software process model that emphasizes a short development cycle. The RAD model is "high-speed" adaptation of Water Fall Model, in which rapid development is achieved by using a component based construction approach. If requirements are well understood and project scope is constrained, the RAD process enables a development team to create "fully functional system "within a very short time period.



**Communication:-** Communication works to understand the business problem and the information characteristics that the software must accommodate.

**Planning:-** Planning is essential because multiple software teams work in parallel on different system functions.

**Modeling:-**Modeling encompasses three major phases- business modeling, data modeling and process modeling- and established design representations that serve as the basis for RAD's construction activity.

**Construction:-**Construction emphasizes the use of preexisting software components and the application of automatic code generation.

**Deployment:-**Deployment establishes a basis for subsequent iterations.

## SYSTEM ENVIRONMENT:-

**Software Tools:-**

| | | |
|---|---|---|
| Operating system | : | Windows 10 |
| Programming Tools | : | Jdk 1.6(JAVA) |
| Database | : | Oracle 10g, SQL |
| Documentation | : | Microsoft Office 2010 |

**OPERATING SYSTEM:** **Windows 10**

**Hardware Requirements:**

- INTEL Core i3
- Colour Monitor (15'')
- Key Board, Mouse(Logitek)
- DVD Writer

**INTERNAL MEMORY**

| | |
|---|---|
| RAM | : 8 GB |
| Flash Memory | : 32MB |

**EXTERNAL MEMORY**

| | |
|---|---|
| HDD | 500 GB |
| Pen Drive | 16 GB |

## COST AND BENEFIT ANALYSIS

Developing an IT application is an investment. Since after developing an application it provides the organization with profits. Profits can be monetary or in the form of an improved working environment. However, it carries risks, because in some cases an estimate can be wrong. And the project might not actually turn out to be beneficial.

Cost benefit analysis helps to give automation a picture of the cost, benefits and risks. It usually involves comparing alternate investments.

Cost benefit determines the benefits and savings that are expected from the system and compares them with the expected costs.

In performing cost and benefit analysis it is important to identify cost and benefits factors. Cost and benefits can be categorized into the following categories:

1. **Development Costs –** Development costs is the costs that are incurred during the development of the system. It is one time investment.
2. **Operating Costs –** Operating Costs are the expenses required for the day to day running of the system. Examples of Operating Costs are Wages, Supplies and Overheads.
3. **Hardware/Software Costs –** It includes the cost of purchasing or leasing of computers and it's peripherals. Software cost involves required S/W costs.
4. **Personnel Costs –** It is the money spent on the people involved in the development of the system.
5. **Facility Costs –** Expenses that are incurred during the preparation of the physical site where the system will be operational. These can be wiring, flooring, acoustics, lightning, and air-conditioning.
6. **Supply Costs –** These are variable costs that are very proportionately with the amount of use of paper, ribbons, disks, and the like.

**BENEFITS**

We can define benefits as

**Profit or Benefit = Income – Costs**

Benefits can be accrued by:

- Increasing income, or
- Decreasing costs, or
- Both

**Cost and Benefit Analysis of SUPER MARKET Automation System**

**Costs:**

| Cost | Cost per unit | Quantity | Cost |
|---|---|---|---|
| **Software** | | | |
| **Java** | 30,000 | 1 | 3,000 |
| **Windows XP** | 30,000 | 1 | 30,000 |
| **Hardware** | 15,000 | 1 | 15,000 |
| | 4,000 | 2 | 8,000 |
| **Central Computer** | 100,000 | 1 | 1,00,000 |
| Client Machine | 50,000 | 4 | 2,00,000 |
| Development | | | |
| Analyst | 50,000 | 1 | 50,000 |
| Developer | 20,000 | 2 | 40,000 |
| Training | 20,000 | 1 | 20,000 |
| Data Entry | 5,0000 | 1 | 5,000 |
| Warranty (1 month) | | | |
| Professional | 20,000 | 1 | 20,000 |
| **TOTAL COST** | **4,91,000** | | |

According to the SUPER MARKET automation System, Rs.250 is to be charged from the Customer for a day of a single bed and Rs.450 is charged for the doctor's consultation fee and the fees charged in O.P.D. is Rs.30.

Expected increase in the number of products: 40 per month and number of products in O.P.D. is: 150 per day.
Let the amount collected from operations in a month: 250,000 for a month.

Amount collected from the discharged Customer this year
=12*(40 * (250 + 450) + 150 * 30 * 30 + 250000)
=Rs. 49,56,000

For three years = 3 * 4956,000
= Rs.1,48,68,000

Now using Net Present Value Method for cost benefit analysis we have,
Net Present Value (origin) = Benefits – Costs
=14868000-491000
=Rs. 14377000
gain % = Net Present Value / Investment
=14377000/491000
=29.28%
Overall gain = 2928% in five year
<u>For each year</u>
<u>1st year:</u>
Investment = 491,000
Benefit = 49,56,000
Net Present Value for first year = 4956000-491000
=4965000
gain%=4965000/491000
=909.36% in first year

2nd year:

Investment = 491,000

Benefit = 10412,000

Net Present Value for first year = 10412000-491000

=9921000

gain%=9921000/491000

=2020.57% at the end of second year

3rd year:

Investment = 491,000

Benefit = 15859000

Net Present Value for first year = 15859000-491000

=15368000

gain%=15368000/491000

=3129.93% at the end of third year

From cost and benefit analysis we have found that the project is economically feasible since it is showing great gains (approx. above 3000%).

After economic feasibility, technical feasibility is done. In this, major issue is to see if the system is developed what is the likelihood that it'll be implemented and put to operation? Will there be any resistance from its user?

It is clear that the new automated system will work more efficiently and faster. So the users will certainly accept it. Also they are being actively involved in the development of the new system. So our system is operationally feasible.

After the feasibility study has been done and it is found to be feasible, the management has approved this project.

# FACT FINDING TECHNIQUES

The functioning of the system is to be understood by the system analyst to design the proposed system. Various methods are used for this and these are known as fact-finding techniques. The analyst needs to fully understand the current system.

The analyst needs data about the requirements and demands of the project undertaken and the techniques employed to gather this data are known as fact-finding techniques. Various kinds of techniques and the most popular among them are interviews, questionnaires, record views, case tools and also the personal observations made by the analyst himself.

- **Interviews**

Interview is a very important data gathering technique as in this the analyst directly contacts system and the potential user of the proposed system.

One very essential aspect of conducting the interview is that the interviewer should first establish a rapport with the interviewee. It should also be taken into account that the interviewee may or may not be a technician and the analyst should prefer to use day to day language instead of jargon and technical terms.

The advantage of the interview is that the analyst has a free hand and the he can extract almost all the information from the concerned people but then as it is a very time consuming method, he should also employ other means such as questionnaires, record reviews, etc. This may also help the analyst to verify and validate the information gained. Interviewing should be approached, as logically and from a general point of view the following guides can be very beneficial for a successful interview:

1. Set the stage for the interview.
2. Establish rapport; put the interview at ease.
3. Phrase questions clearly and succinctly.
4. Be a good listener; a void arguments.
5. Evaluate the outcome of the interview.

The interviews are of the two types namely **structured** and **unstructured**.

### I . Structured Interview

Structured interviews are those where the interviewee is asked a standard set of questions in a particular order. All interviews are asked the same set of questions. The questions are further divided into two kinds of formats for conducting this type if interview.

### II. Unstructured Interview

The unstructured interviews are undertaken in a question-and-answer format. This is of a much more flexible nature than the structured and can be very rightly used to gather general information about the system.

- **Questionnaires:**

Questionnaires are another way of information gathering where the potential users of the system are given questionnaires to be filled up and returned to the analyst.

Questionnaires are useful when the analyst need to gather information from a large number of people. It is not possible to interview each individual. Also if the time is very short, in that case also questionnaires are useful. If the analyst guarantees the anonymity of the respondent then the respondent answers the questionnaires very honestly and critically.

The analyst should sensibly design and frame questionnaires with clarity of it's objective so as to do just to the cost incurred on their development and distribution.

- **Record Reviews**

Records and reports are the collection of information and data accumulated over the time by the users about the system and it's operations. This can also put light on the requirements of the system and the modifications it has undergone. Records and reports may have a limitation if they are not up-to-date or if some essential links are missing. All the changes, which the system suffers, may not be recorded. The analyst may scrutinize the records either at the beginning of his study which may give him a fair introduction about the system and will make him familiar with it or in the end which will provide the analyst with a comparison between what exactly is/was desired from the system and it's current working.

## ❑ On-Site Observation

On-site observations are one of the most effectively tools with the analyst where the analyst personally goes to the site and discovers the functioning of the system. As a observer, the analyst can gain first hand knowledge of the activities, operations, processes of the system on-site, hence here the role of an analyst is of an information seeker. This information is very meaningful as it is unbiased and has been directly taken by the analyst. This exposure also sheds some light on the actual happenings of the system as compared to what has already been documented, thus the analyst gets closer to system. This technique is also time-consuming and the analyst should not jump to conclusions or draw inferences from small samples of observation rather the analyst should be more Customer in gathering the information. This method is however less effective for learning about people's perceptions, feelings and motivations.

## ANALYST'S INTERVIEW WITH SUPER MARKET ADMINISTRATOR

**Analyst:** Hi, I have come to talk to you regarding the functioning of your SUPER MARKET project.

**Administrator:** Hello, do come in. I was expecting you.

**Analyst:** I'll come straight to the point. Don't hesitate, you can be as much open you want. There are no restrictions.

**Administrator:** I'll give you my whole contribution.

**Analyst:** Tell me are you excited about the idea of having an automated system for your SUPER MARKET?

**Administrator:** Yes, I do. Very much. After all it's going to reduce our loads of work.

**Analyst:** Will you elaborate on it?

**Administrator:** Major problem is managing the sales and purchase information of the product. There are a number of products of different companies. So, maintain the stock information is tedious task.

**Analyst:** What do you think be ideal solution to this?

**Administrator:** All the information of products should be put into computer. It'll be easy for us to check how many products are not available and which company product is available in stock.

**Analyst:** Could you explain how?

**Administrator:** Look whenever a new product is entry, product information in stock is automatically updated and product quantity is increase. But, if sales the product the quantity of product is automatically decrease.

**Analyst:** Do you have different product categories?

**Administrator:** Yes, I categories the product on the basics of size and company.

**Analyst:** Do you have any other expectations or suggestion for the new system?

**Administrator:** It should be able to produce reports faster.

**Analyst:** Reports? I completely forgot about them. What reports you people produce presently?

**Administrator:** Well first is for product status, another for employee information and salary information.

**Analyst:** Do you have some format for them?

**Administrator:** Yes we do have and we want that the same format be used by the new system.

**Analyst:** Yes we’ll take care of that. Any other suggestions?

**Administrator:** No. You have already covered all the fields.

**Analyst:** Thanks for your co-operation. It was nice talking to you.

**Administrator:** My pleasure. Bye.

## QUESTIONNAIRES FOR SUPER MARKET STAFF

Instructions: Answer as specified by the format. Put NA for non-application situation.

1. What are your expectations out of the new system (computerized)? Rate the following on a scale of 1-4 giving allow value for low priority.
   (a) better cataloguing
   (b) better managing of users
   (c) better account and products management
   (d) computer awareness
   (e) any other________________
2. How many users are you expecting?
   ____________________________
3. How many products are there in the SUPER MARKET?
   ____________________________
4. How you want the product to be categorized for searching (like by company name, product id, product name)?
   ____________________________
5. Is there any difference in the roles (privileges) of two or more products?
   Yes/No Please specify if Yes
   _____________________________________________________________________
6. Do you want facility of reserving a product from phone in advance?
   Yes/No
7. Do you have data of products entered into some kind of database?
   Yes/No
8. How do you want users to be categorized?
   _______________________or_______________________
9. Would you like online registration for users rather than the printed form?
   Yes/No
10. Do you already have some existing categorization of products on the basis as specified in question 4 above?
    Yes/No
11. Any other specific suggestion/expectation out of the proposed system.
    ____________________________________________________________________________

## SYSTEM OVERVIEW

The limited time and resources have restricted us to incorporate, in this project, only a main activities that are performed in a SUPER MARKET AUTOMATION System, but utmost care has been taken to make the system efficient and user friendly. "SUPER MARKET AUTOMATION System" has been designed to computerized the following functions that are performed by the system:

1. On Line purchasing the Products

a) Admission of New Customer

2. Free Advice For the Products
3. Bill generation facility
4. Training Courses Provided by the SUPER MARKET
5. Statement of Customer Details
6. Total number of Products purchases in the SUPER MARKET
7. Products available in the SUPER MARKET
8. Administrator Links

a. Login Form

b. To add new products in the site

## IMPORTANCE OF SUPER MARKET AUTOMATION SYSTEM

There are several attributes in which the computer based information works. Broadly the working of computer system is divided into two main groups:

- ♦ Transaction System
- ♦ Decision Support System

**Transaction System**:

A transaction is a record of some well-defined single and usually small occurrence in a system. Transactions are input into the computer to update the database files. It checks the entering data for its accuracy. This means that numeric data appears in numeric field and character data in character field. Once all the checks are made, transaction is used to update the database. Transaction can be inputted in on-line mode or batch mode. In on-line mode, transactions are entered and updated into the database almost instantaneously. In batch mode, transactions are collected into batches, which may be held for a while and inputted later.

**Decision Support System**:

It assists the user to make analytical decision. It shows the various data in organized way called analysis. This analysis can be made to study preferences and help in making decisions.

Computer system works out best with record maintenance. It will tell you which customer would get how much pending/reports statements. It will also help to search the information about a particular person by simply entering his telephone number.

User can store information as per requirement, which can be used for comparison with other reports.

## SYSTEM DESIGN

The design document that we will develop during this phase is the blueprint of the software. It describes how the solution to the customer problem is to be built. Since solution to complex problems isn't usually found in the first try, iterations are most likely required. This is true for software design as well. For this reason, any design strategy, design method, or design language must be flexible and must easily accommodate changes due to iterations in the design. Any technique or design needs to support and guide the partitioning process in such a way that the resulting sub-problems are as independent as possible from each other and can be combined easily for the solution to the overall problem. Sub-problem independence and easy combination of their solutions reduces the complexity of the problem. This is the objective of the partitioning process. Partitioning or decomposition during design involves three types of decisions: -

Define the boundaries along which to break;

Determine into how money pieces to break; and

Identify the proper level of detail when design should stop and implementation should start.

Basic design principles that enable the software engineer to navigate the design process suggest a set of principles for software design, which have been adapted and extended in the following list:

Free from the suffer from "tunnel vision." A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job.

The design should be traceable to the analysis model. Because a single element of the design model often traces to multiple requirements, it is necessary to have a means for tracking how requirements have been satisfied by the design model.

The design should not repeat the same thing. Systems are constructed using a set of design patterns, many of which have likely been encountered before. These patterns should always be chosen as an alternative to reinvention. Time is short and resources are limited! Design time should be invested in representing truly new ideas and integrating those patterns that already exist.

The design should "minimize the intellectual distance" between the software and the problem as it exists in the real world. That is, the structure of the software design should (whenever possible) mimic the structure of the problem domain.

The design should exhibit uniformity and integration. A design is uniform if it appears that one person developed the entire thing. Rules of style and format should be defined for a design team before design work begins. A design is integrated if care is taken in defining interfaces between design components.

The design activity begins when the requirements document for the software to be developed is available. This may be the SRS for the complete system, as is the case if the waterfall model is being followed or the requirements for the next "iteration" if the iterative enhancement is being followed or the requirements for the prototype if the prototyping is being followed. While the requirements specification activity is entirely in the problem domain, design is the first step in moving from the problem domain toward the solution domain. Design is essentially the bridge between requirements specification and the final solution for satisfying the requirements.

The design of a system is essentially a blueprint or a plan for a solution for the system. We consider a system to be a set of components with clearly defined behavior that interacts with each other in a fixed defined manner to produce some behavior or services for its environment. A component of a system can be considered a system, with its own components. In a software system, a component is a software module.

The design process for software systems, often, has two levels. At the first level, the focus is on deciding which modules are needed for the system, the specifications of these modules, and how the modules should be interconnected. This is what is called the system design or top-level design. In the second level, the internal design of the modules, or how the specifications of the module can be satisfied, is decided. This design level is often called detailed design or logic design. Detailed design essentially expands the system design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding.

Because the detailed design is an extension of system design, the system design controls the major structural characteristics of the system. The system design has a major impact on the testability and modifiability of a system, and it impacts its efficiency. Much of the design effort for designing software is spent creating the system design.

The input to the design phase is the specifications for the system to be designed. Hence, reasonable entry criteria can be that the specifications are stable and have been approved, hoping that the approval mechanism will ensure that the specifications are complete, consistent, unambiguous, etc. The output of the top-level design phase is the architectural design or the system design for the software system to be built. This can be produced with or without using a design methodology. A reasonable exit criteria for the phase could be that the design has been verified against the input specifications and has been evaluated and approved for quality.

A design can be object-oriented or function-oriented. In function-oriented design, the design consists of module definitions, with each module supporting a functional abstraction. In object-oriented design, the modules in the design represent data abstraction (these abstractions are discussed in more detail later). In the function-oriented methods for design and describe one particular methodology the structured design methodology in some detail. In a function- oriented design approach, a system is viewed as a transformation function, transforming the inputs to the desired outputs. The purpose of the design phase is to specify the components for this transformation function, so that each component is also a transformation function. Hence, the basic output of the system design phase, when a function oriented design approach is being followed, is the definition of all the major data structures in the system, all the major modules of the system, and how the modules interact with each other.

Once the designer is satisfied with the design he has produced, the design is to be precisely specified in the form of a document. To specify the design, specification languages are used. Producing the design specification is the ultimate objective of the design phase. The purpose of this design document is quite different from that of the design notation. Whereas a design represented using the design notation is largely to be used by the designer, a design specification has to be so precise and complete that it can be used as a basis of further development by other programmers. Generally, design specification uses textual structures, with design notation helping in understanding.

## SCHEDULING

Scheduling of a software project does not differ greatly from scheduling of any multi- task engineering effort. Therefore, generalized project scheduling tools and techniques can be applied with little modification to software projects.
Program evaluation and review technique (PERT) and critical path method (CPM) are two project scheduling methods that can be applied to software development. Both techniques are driven by information already developed in earlier project planning activities.

## Estimates of Effort

- A decomposition of the product function.
- The selection of the appropriate process model and task set.
- Decomposition of tasks.

Interdependencies among tasks may be defined using a task network. Tasks, sometimes called the project Work Breakdown Structure (WBS) are defined for the product as a whole or for individual functions.

Both PERT and CPM provide quantitative tools that allow the software planner to

(1) determine the critical path-the chain of tasks that determines the duration of the project;

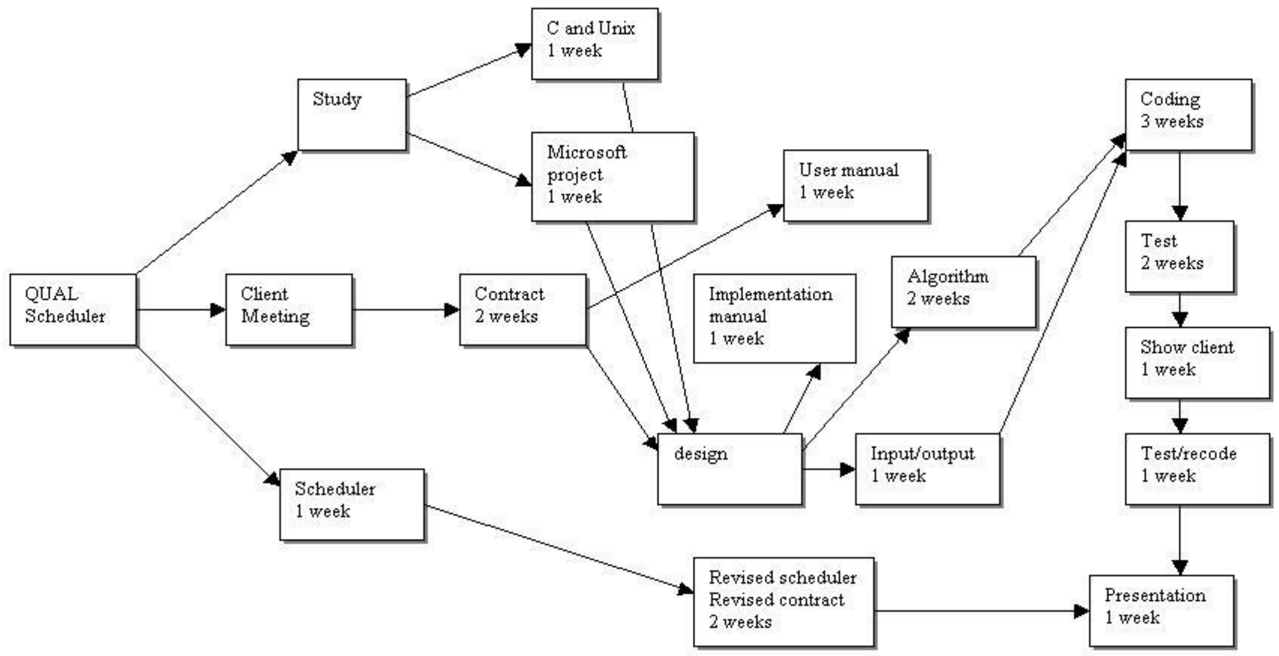
(2) establish "most likely" time estimates for individual tasks by applying statistical models; and

(3) calculate "boundary times" that define a time window" for a particular task.

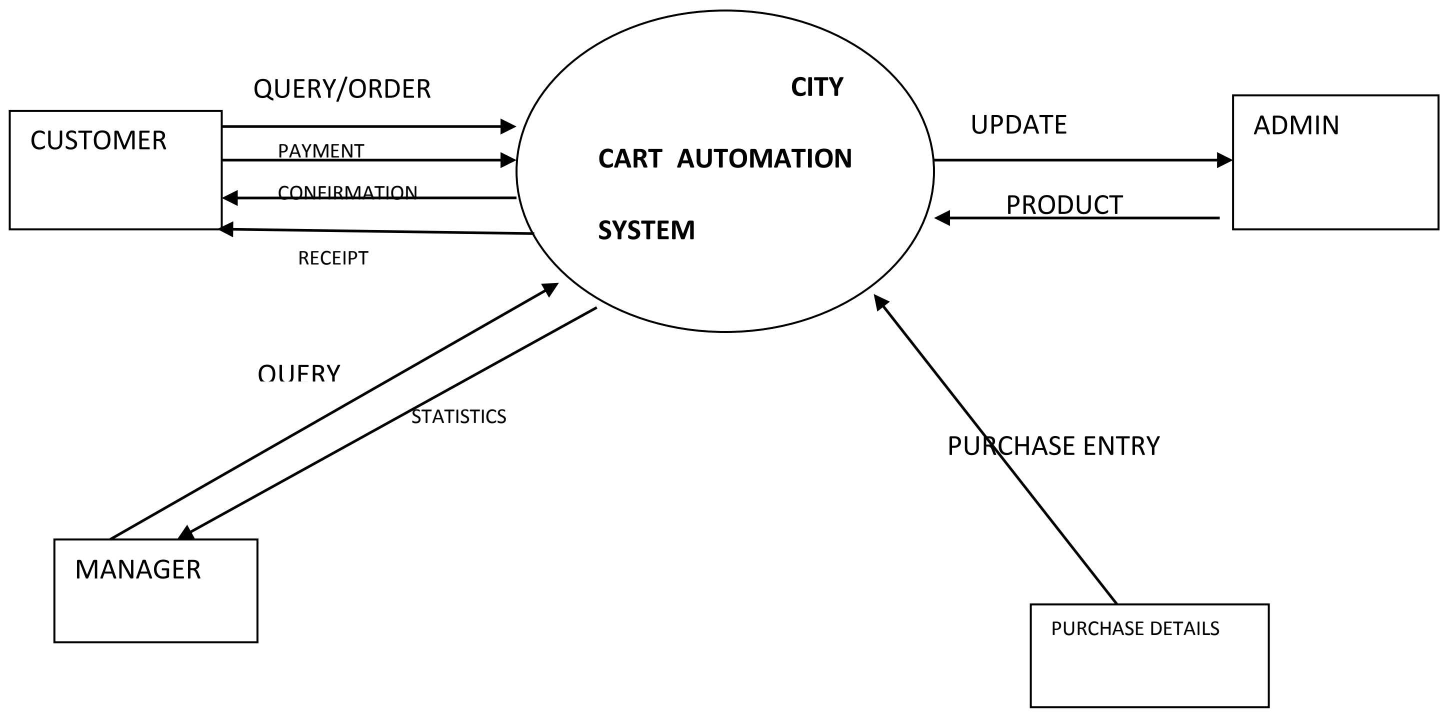
Boundary time calculations can be very useful in software project scheduling. Slippage in the design of one function, for example, can retard further development of other functions. It describes important boundary times that may be discerned from a PERT or CPM network: (I) the earliest time that a task can begin when preceding tasks are completed in the shortest possible time, (2) the latest time for task initiation before the minimum project completion time is delayed, (3) the earliest finish-the sum of the earliest start and the task duration, (4) the latest finish- the latest start time added to task duration, and (5) the total float-the amount of surplus time or leeway allowed in scheduling tasks so that the network critical path maintained on schedule. Boundary time calculations lead to a determination of critical path and provide the manager with a quantitative method for evaluating progress as tasks are completed.

Both PERT and CPM have been implemented in a wide variety of automated tools that are available for the personal computer. Such tools are easy to use and take the scheduling methods described previously available to every software project manager.

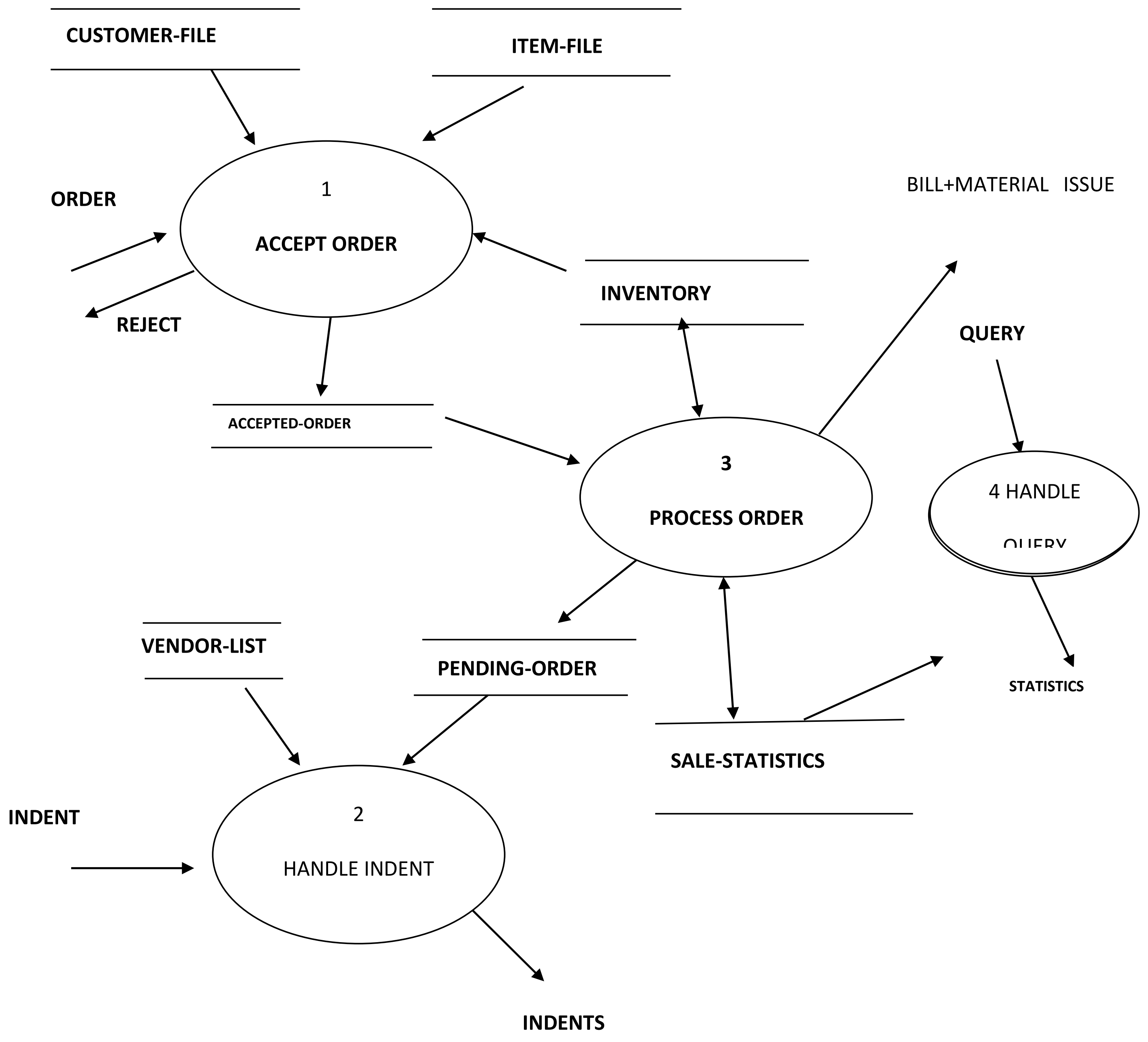
## PERT CHART


QUAL Scheduler
Study
C and Unix 1 week
Microsoft project 1 week
Client Meeting
Contract 2 weeks
User manual 1 week
Implementation manual 1 week
Algorithm 2 weeks
design
Input/output 1 week
Scheduler 1 week
Revised scheduler Revised contract 2 weeks
Coding 3 weeks
Test 2 weeks
Show client 1 week
Test/recode 1 week
Presentation 1 week

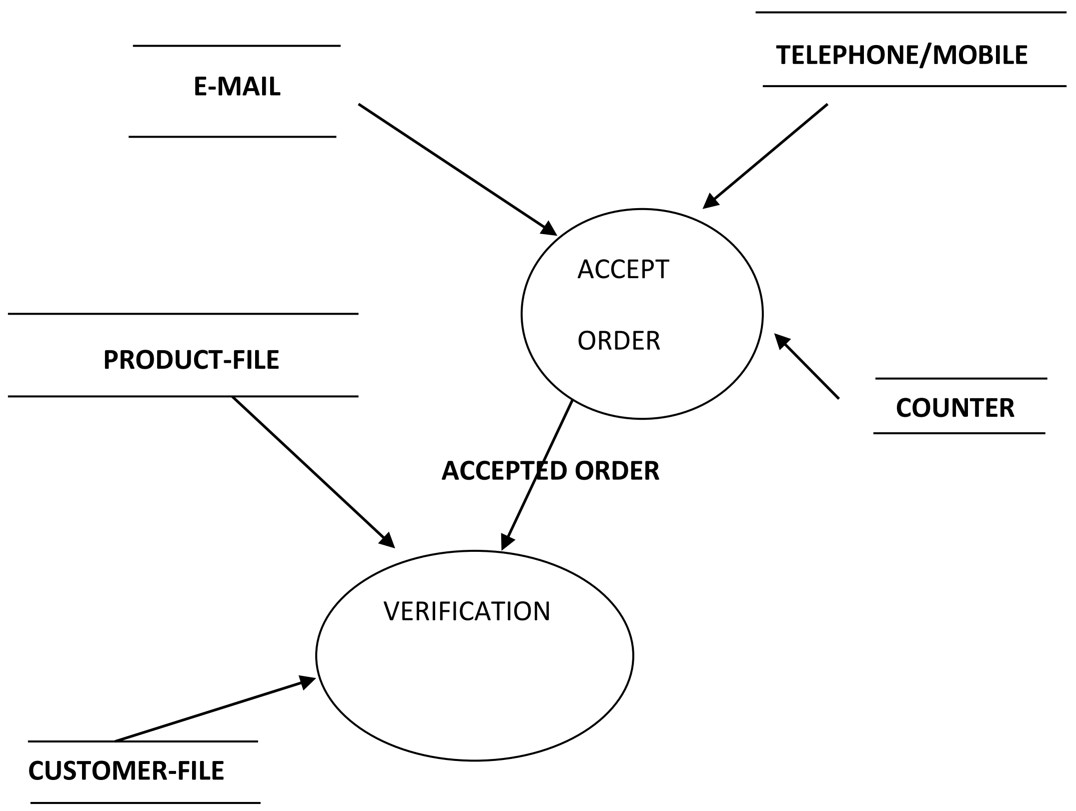
## DATA FLOW DIAGRAM



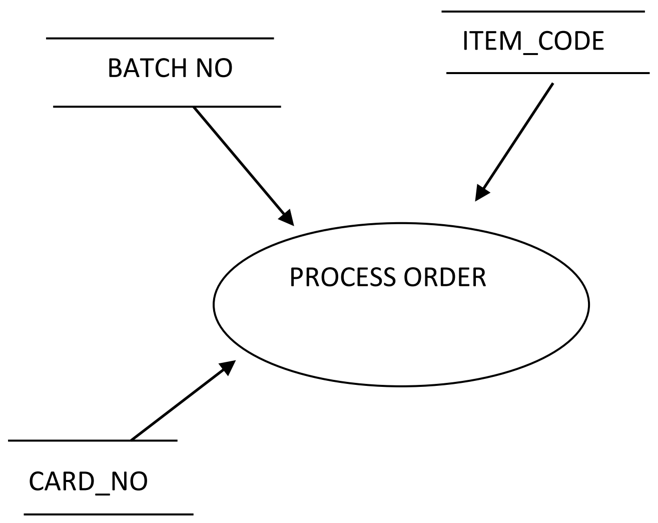
**CONTEXT LEVEL DATA FLOW DIAGRAM**



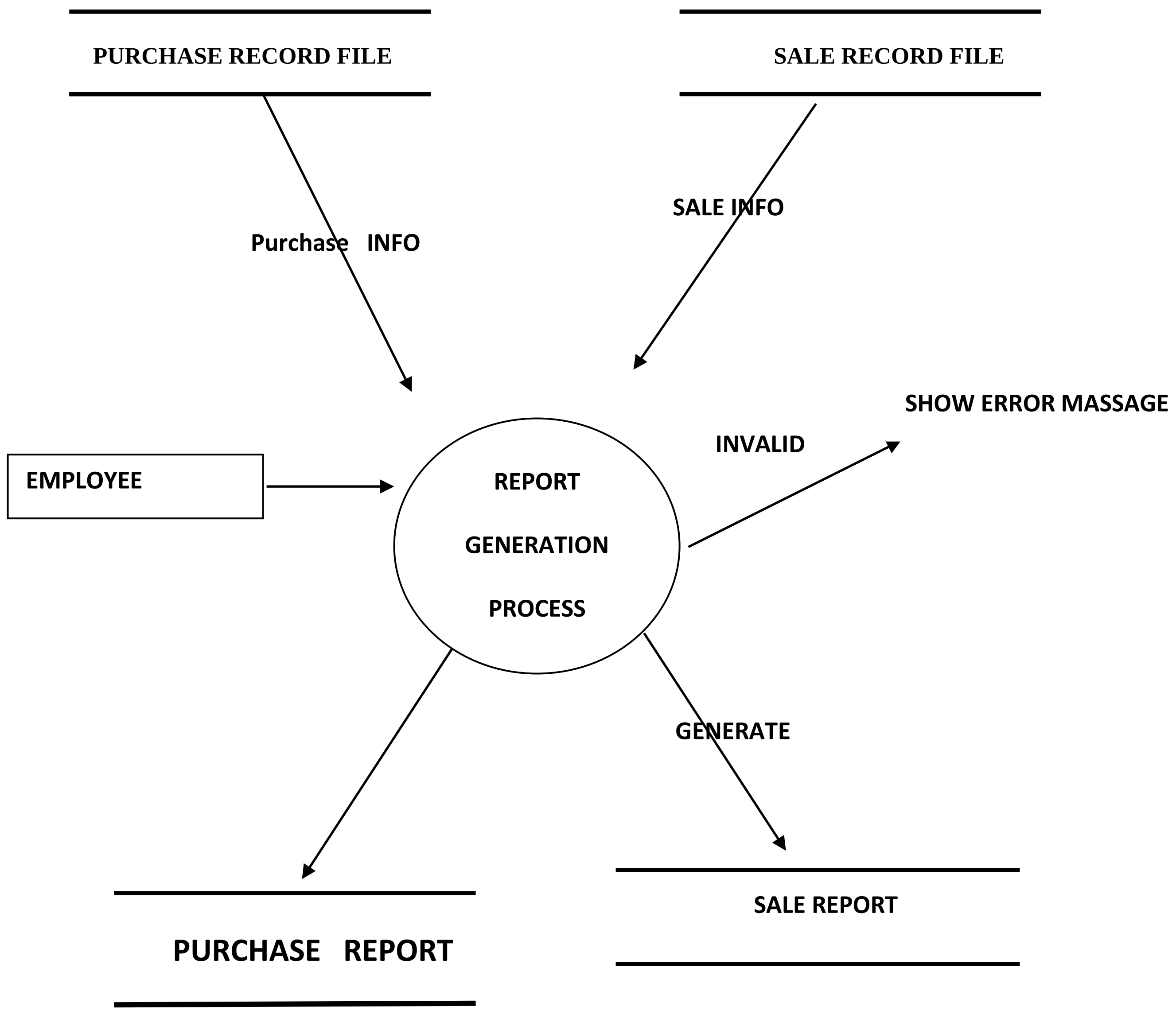
**1 LEVEL DATA FLOW DIGRAM**



**2ND LEVEL DFD FOR ORDER ACCEPT**

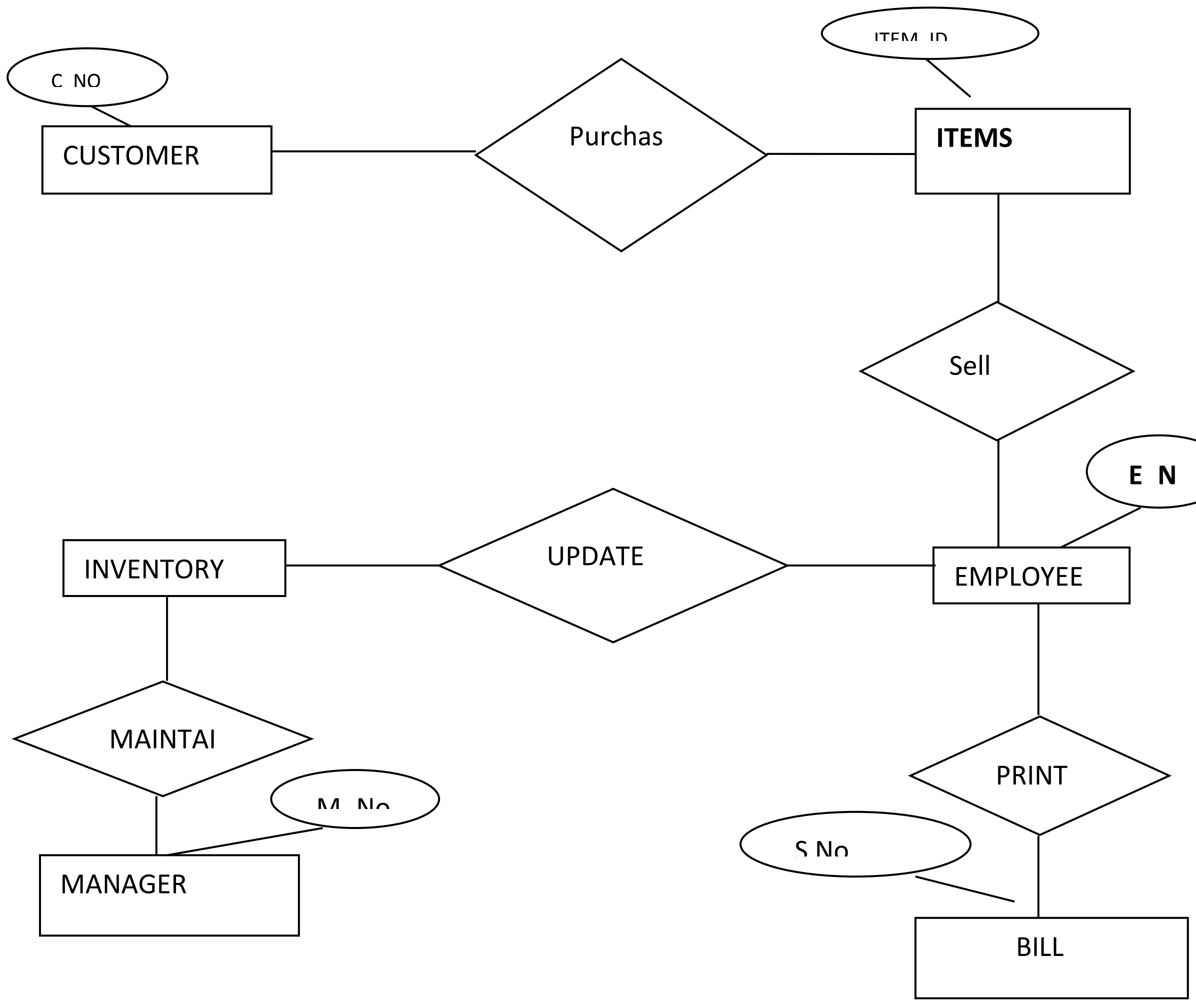


**2ND LEVEL DFD FOR PROCESS ORDER**

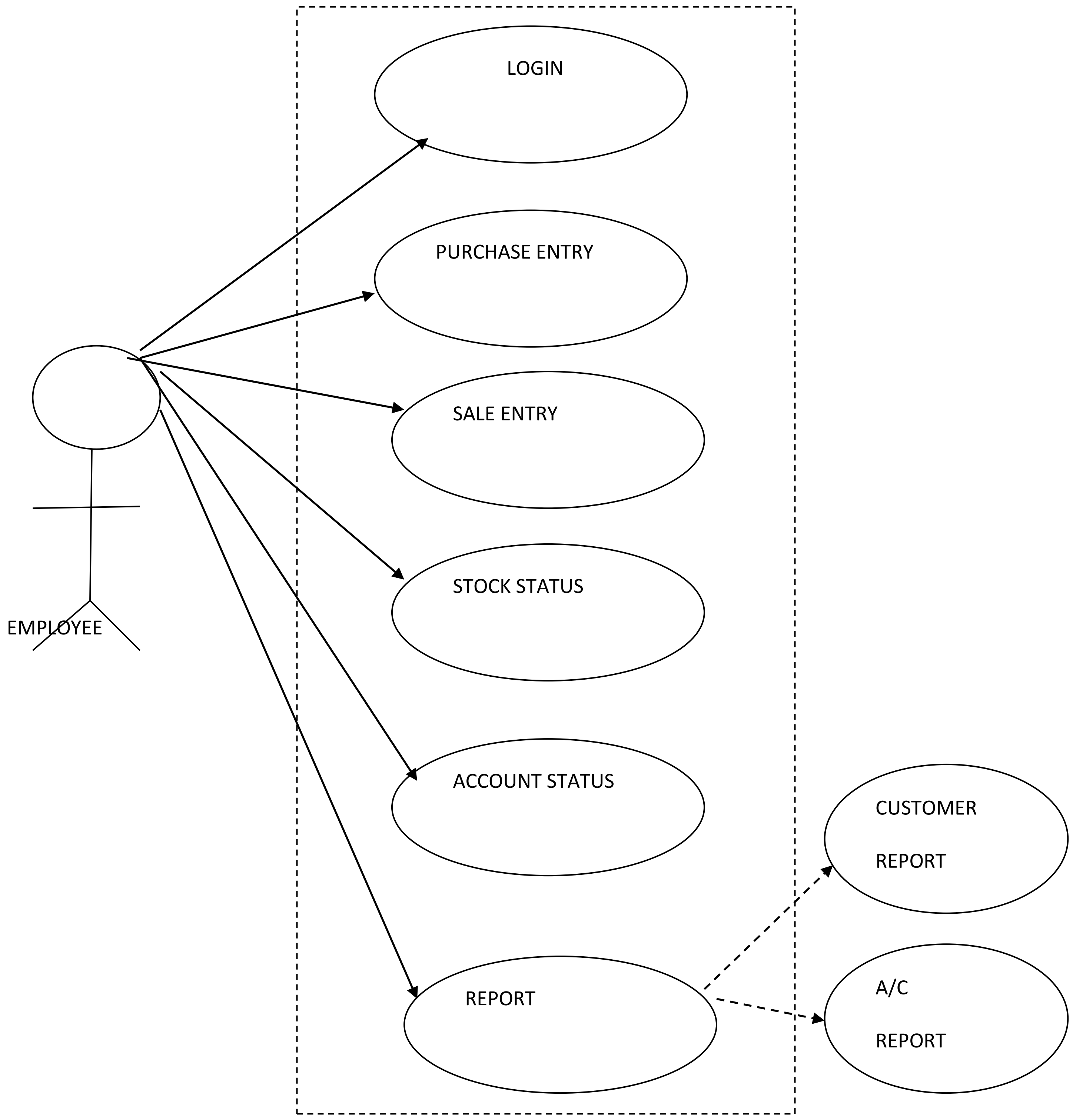


**2nd Level Data Flow Diagram of Report Generation Process**

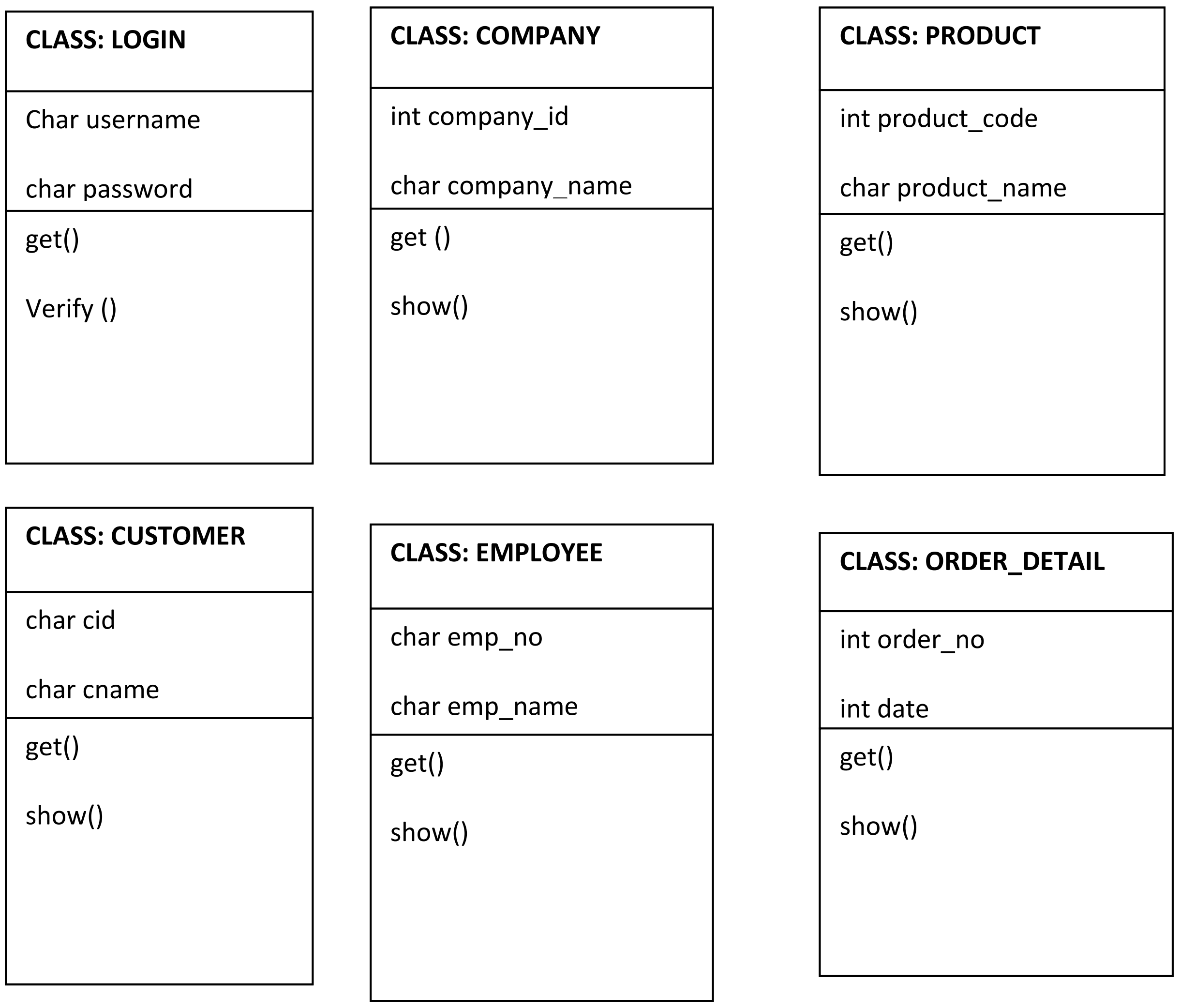
## E-R DIAGRAM


ITEM_ID
C_NO
CUSTOMER
Purchas
ITEMS
Sell
E_N
INVENTORY
UPDATE
EMPLOYEE
MAINTAI
M_No
PRINT
S No
MANAGER
BILL

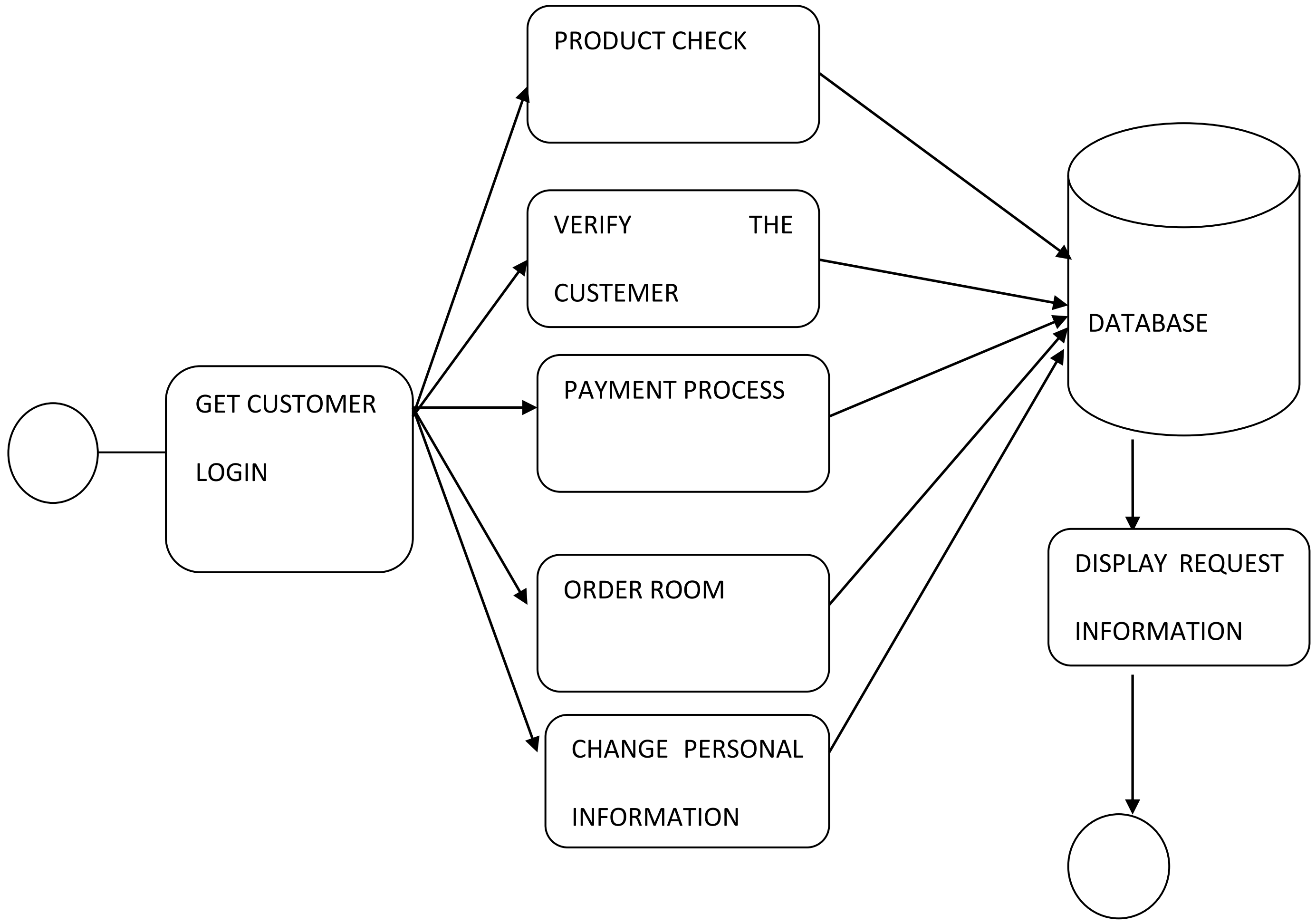
## USE-CASE DIAGRAM FOR EMPLOYEE



## CLASS DIAGRAM



## PROCESS LOGIC DIAGRAM



## TABLE DESCRIPTION

The database is designed such a way that, it is free from redundancy, potential inconsistency, insertion anomalies & deletion anomalies. Proper care has been taken while designing the tables.

- Connect system/manager
- Create user citycart identified by citycart.
- Grant dba to citycart
- Connect  citycart/citycart

**TABLE: LOGIN**

**DESCRIPTION:** This table stores the username and password description.

**UNIQUE:** **USERID**

**QUERY:**
```
CREATE TABLE LOGIN
(USERNAME VARCHAR (10),
PASSWORD VARCHAR (20),
CONSTRAINT PKO1 PRIMARY KEY (USERNAME));
```

| ATTRIBUTE NAME | DATA TYPE | SIZE | CONSTRAINT |
|---|---|---|---|
| USER_NAME | VARCHAR2 | 4 | PRIMARY KEY(C) |
| PASSWORD | VARCHAR2 | 20 | |

**MASTER**

**TABLE: COMPANY**

**DESCRIPTION:** This table stores the information about the company.

**UNIQUE:** **COMPANY_ID**

**QUERY:**
```
CREATE TABLE COMPANY
(COMPANY_ID NUMBER (10),
COMPANY_NAME VARCHAR (25),
ADDRESS VARCHAR (40),
CITY VARCHAR (40),
STATE VARCHAR (40),
MOBILE NUMBER (10),
EMAIL VARCHAR (40),
CONSTRAINT PKO2 PRIMARY KEY (COMPANY_ID));
```

| **ATTRIBUTE NAME** | **DATATYPE** | **SIZE** | **CONSTRAINT** |
|---|---|---|---|
| COMPANY_ID | NUMBER | 10 | PRIMARY KEY |
| COMPANY_NAME | VARCHAR2 | 25 | NOT NULL |
| ADDRESS | VARCHAR2 | 40 | NOT NULL |
| CITY | VARCHAR2 | 40 | |
| STATE | VARCHAR2 | 30 | |
| MOBILE | NUMBER | 10 | |
| EMAIL | VARCHAR2 | 40 | |

**TABLE: PRODUCT**

**DESCRIPTION:** This table stores the information about the product.

**UNIQUE:** **PRODUCT_ID**

**QUERY:**
```
CREATE TABLE PRODUCT
(PRODUCT_ID NUMBER (10),
PRODUCT_NAME VARCHAR (25),
PRICE NUMBER (7, 2),
```

LENGHT NUMBER (5),

BATCHNO NUMBER (5),

COMPANY_ID NUMBER (10),

CONSTRAINT PKO3 PRIMARY KEY (PRODUCT_ID));

| ATTRIBUTE NAME | DATATYPE | SIZE | CONSTRAINT |
|---|---|---|---|
| PRODUCT_ID | NUMBER | 10 | PRIMARY KEY |
| PRODUCT_NAME | VARCHAR2 | 25 | NOT NULL |
| PRICE | NUMBER | 7,2 | |
| LENGTH | NUMBER | 5 | |
| BATCH_NO | NUMBER | 5 | FOREIGN KEY |
| COMPANY_ID | NUMBER | 5 | FOREIGN KEY |

**TABLE: CUSTOMER**

**DESCRIPTION:** This table stores the information about the customer.

**UNIQUE:** **CUSTOMER_NO**

**QUERY:** CREATE TABLE CUSTOMER

(CUSTOMER_NO NUMBER (4),

CUSTOMER_NAME VARCHAR (25),

SEX VARCHAR (7),

DOB DATE,

ADDRESS VARCHAR (30),

MOBILE NUMBER (10),

F_NAME VARCHAR (15),

F_OCCUPATION VARCHAR (15);

RELIGION VARCHAR (10),

NATIONALITY VARCHAR (10),

CONSTRAINT PKO4 PRIMARY KEY (CUSTOMER_NO));

| **ATTRIBUTE NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
|---|---|---|---|
| CUSTOMER_NO | VARCHAR2 | 4 | PRIMARY KEY(C) |
| CUSTOMER_NAME | VARCHAR2 | 20 | |
| SEX | VARCHAR2 | 7 | |
| DOB | DATE | - | |
| ADDRESS | VARCHAR2 | 30 | |
| PHONE_CODE | VARCHAR2 | 12 | |
| F_NAME | VARCHAR2 | 15 | |
| F_OCCUPATION | VARCHAR2 | 15 | |
| RELIGION | VARCHAR2 | 10 | |
| NATIONALITY | VARCHAR2 | 10 | |

**TABLE: EMPLOYEE**

**DESCRIPTION:** This table stores the information about the employee.

**UNIQUE:** **CUSTOMER_NO**

**QUERY:**

```
CREATE TABLE EMPLOYEE
(EMP_CODE VARCHAR (4),
EMP_NAME VARCHAR (20),
ADDRESS VARCHAR (30),
DOJ DATE,
SEX VARCHAR (7),
TYPE VARCHAR (20),
PHONE_NO NUMBER (10),
SALARY NUMBER (7),
CONSTRAINT PKO5 PRIMARY KEY (EMP_CODE));
```

| **ATTRIBUTE NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
|---|---|---|---|
| EMP_CODE | VARCHAR2 | 4 | PRIMARY KEY |
| EMP_NAME | VARCHAR2 | 20 | |
| ADDRESS | VARCHAR2 | 30 | |
| DOJ | DATE | - | |
| SEX | CHAR | 7 | |
| TYPE | VARCHAR2 | 20 | |
| PHONE_NO | NUMBER | 12 | |
| SALARY | NUMBER | 7 | |

**TRANSACTION**

**TABLE: ORDER_DETAIL**

**DESCRIPTION:** This table stores the information about the order detail. This is the transaction table.

**UNIQUE:** **ORDER_NO**

**QUERY:**
```
CREATE TABLE ORDER_DETAIL
(ORDER_NO NUMBER (10),
ORDER_DATE DATE,
PRODUCT_ID NUMBER (10),
QTY NUMBER (5),
ORDER_STATUS VARCHAR (10),
CONSTRAINT PKO6 PRIMARY KEY (ORDER_NO));
```

| **ATTRIBUTE NAME** | **DATATYPE** | **SIZE** | **CONSTRAINT** |
|---|---|---|---|
| ORDER_NO | NUMBER | 10 | PRIMARY KEY |

| | | | |
|---|---|---|---|
| ORDER_DATE | DATE | - | NOT NULL |
| PRODUCT_ID | NUMBER | 10 | FOREIGN KEY |
| QTY | NUMBER | 5 | |
| ORDER_STATUS | VARCHAR2 | 10 | |

**TABLE: BILL_DETAIL**

**DESCRIPTION:** This table stores the information about the bill detail. This is also the transaction table.

**UNIQUE:** **BILL_NO**

**QUERY:**

```
CREATE TABLE BILL_DETAIL
(BILL_NO NUMBER (10),
BILL_DATE DATE,
ORDER_NO NUMBER (10),
SALE_QTY NUMBER (5),
PRODUCT_ID NUMBER 10),
CUSTOMER_NO NUMBER (10),
DISCOUNT NUMBER (5, 2),
AMOUNT NUMBER (8, 2),
CONSTRAINT PKO7 PRIMARY KEY (BILL_NO));
```

| ATTRIBUTE NAME | DATATYPE | SIZE | CONSTRAINT |
|---|---|---|---|
| BILL_NO | NUMBER | 10 | PRIMARY KEY |
| BILL_DATE | DATE | | |
| ORDER_NO | NUMBER | 10 | FOREIGN KEY |
| SALE_QTY | NUMBER | 5 | |
| PRODUCT_ID | NUMBER | 10 | FOREIGN KEY |
| CUSTOMER_NO | NUMBER | 10 | FOREIGN KEY |

| | | | |
|---|---|---|---|
| DISCOUNT | NUMBER | 5,2 | |
| AMOUNT | NUMBER | 8,2 | |

**TABLE: PAYMENT**

**DESCRIPTION:** This table stores the information about the payment detail. This is also the transaction table.

**UNIQUE:** **PAYMENT_NO**

**QUERY:**

```
CREATE TABLE PAYMENT
(PAYMENT_NO NUMBER (10),
P_DATE DATE,
P_AMOUNT NUMBER (4),
CUSTOMER_NO VARCHAR (4),
CUSTOMER_TYPE VARCHAR (4),
YEAR VARCHAR (4),
CONSTRAINT PKO8 PRIMARY KEY (PAYMENT_NO));
```

| ATTRIBUTE NAME | DATA TYPE | SIZE | CONSTRAINT |
|---|---|---|---|
| PAYMENT_NO | VARCHAR2 | 4 | PRIMARY KEY |
| P_DATE | DATE | | |
| P_AMOUNT | NUMBER | 4 | |
| CUSTOMER_NO | VARCHAR2 | 4 | |
| CUSTOMER_TYPE | VARCHAR2 | 4 | |
| YEAR | VARCHAR2 | 4 | |

**INPUT PROCESS OUTPUT**

| **FORM NAME: PRODUCT_DETAIL** | | |
|---|---|---|
| **INPUT** | **PROCESS** | **OUTPUT** |
| PRODUCT_ID | | SUCCESSFULLY |
| PRODUCT_NAME | INSERT OR UPDATE | ERROR: NOT SUCCESSFULLY |
| PRICE | | DATA IS FOUND |
| QUANTITY | SEARCH | OR, DATA IS NOT FOUND |
| BATCH_NO | | DELETE THE RECORD |
| COMPANY_ID | DELETE | OR, DATA NOT FOUND |
| | | |

| **FORM NAME: EMPLOYEE DETAIL** | | |
|---|---|---|
| **INPUT** | **PROCESS** | **OUTPUT** |
| EMP_CODE | | SUCCESSFULLY |
| EMP_NAME | INSERT OR UPDATE | ERROR: NOT SUCCESSFULLY |
| ADDRESS | | DATA IS FOUND |
| DOJ | SEARCH | OR, DATA IS NOT FOUND |
| SEX | | DELETE THE RECORD |
| PHONE_NO | DELETE | OR, DATA NOT FOUND |

**FORM NAME: CUSTOMER DETAIL**

| **INPUT** | **PROCESS** | **OUTPUT** |
|---|---|---|
| CUSTOMER_NO | | SUCCESSFULLY |
| CUSTOMER_NAME | INSERT OR UPDATE | ERROR: NOT SUCCESSFULLY |
| ADDRESS | | DATA IS FOUND |
| DOJ | SEARCH | OR, DATA IS NOT FOUND |
| SEX | | DELETE THE RECORD |
| MOBILE | DELETE | OR, DATA NOT FOUND |
| RELIGION | | |

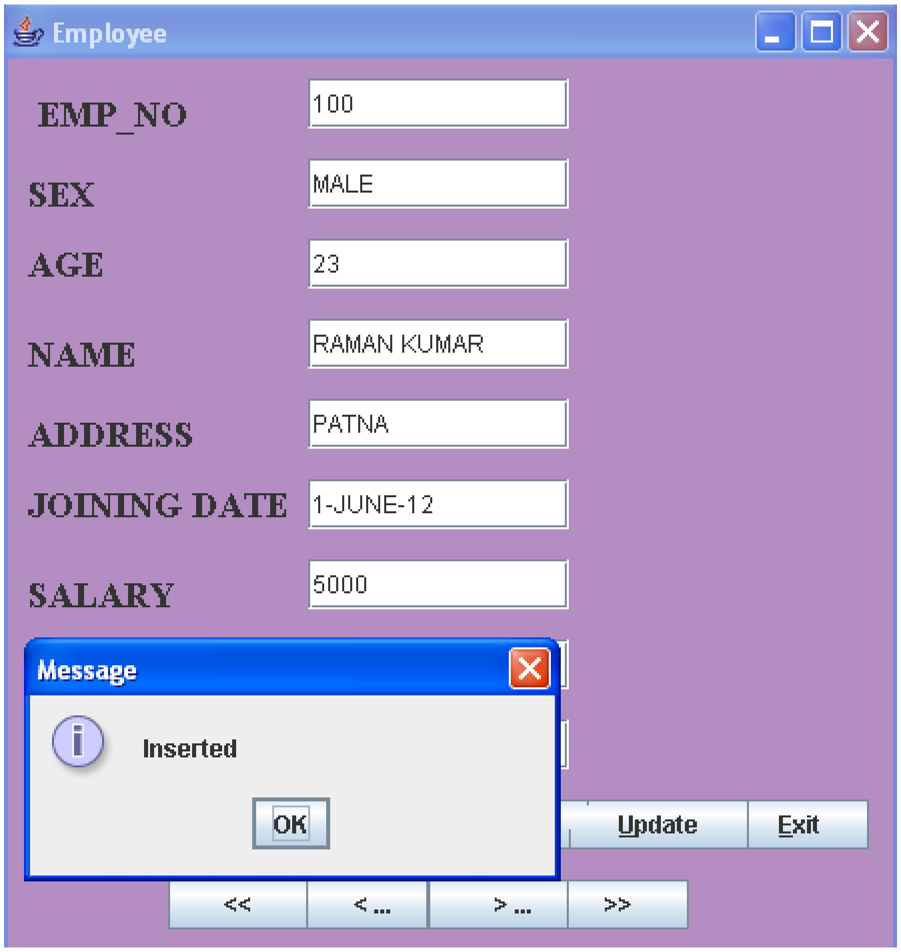
**FORM NAME: BILL DETAIL**

| **INPUT** | **PROCESS** | **OUTPUT** |
|---|---|---|
| BILL_NO | | SUCCESSFULLY |
| BILL_DATE | INSERT OR UPDATE | ERROR: NOT SUCCESSFULLY |
| ORDER_NO | | DATA IS FOUND |
| SALE_QTY | SEARCH | OR, DATA IS NOT FOUND |
| PRODUCT_ID | | DELETE THE RECORD |
| AMOUNT | DELETE | OR, DATA NOT FOUND |

| **FORM NAME: ORDER DETAIL** | | |
|---|---|---|
| **INPUT** | **PROCESS** | **OUTPUT** |
| ORDER_NO<br>ORDER_DATE<br>PRODUCT_ID<br>QTY<br>ORDER_STATUS | INSERT OR UPDATE | SUCCESSFULLY |
| | | ERROR: NOT SUCCESSFULLY |
| | SEARCH | DATA IS FOUND |
| | | OR, DATA IS NOT FOUND |
| | DELETE | DELETE THE RECORD |
| | | OR, DATA NOT FOUND |

| **FORM NAME: PAYMENT DETAIL** | | |
|---|---|---|
| **INPUT** | **PROCESS** | **OUTPUT** |
| PAYMENT_NO<br>P_DATE<br>P_AMOUNT<br>CUSTOMER_NO<br>CUSTOMER_TYPE<br>YEAR | INSERT OR UPDATE | SUCCESSFULLY |
| | | ERROR: NOT SUCCESSFULLY |
| | SEARCH | DATA IS FOUND |
| | | OR, DATA IS NOT FOUND |
| | DELETE | DELETE THE RECORD |
| | | OR, DATA NOT FOUND |

# SCREEN LAYOUT

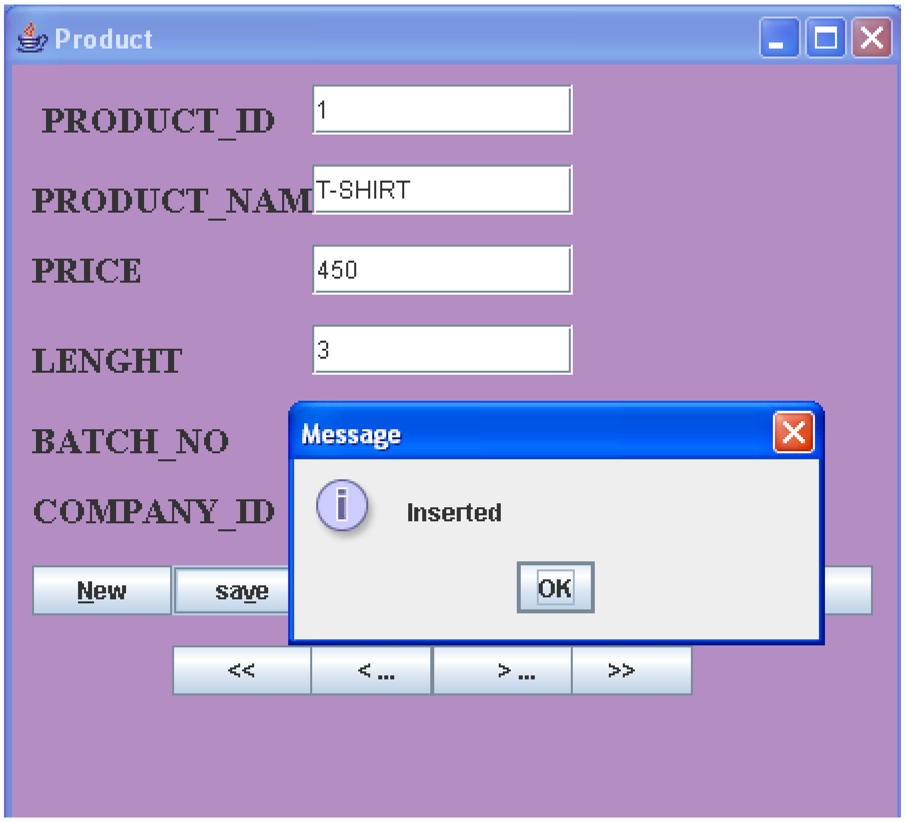


Employee

| | |
|---|---|
| EMP_NO | 100 |
| SEX | MALE |
| AGE | 23 |
| NAME | RAMAN KUMAR |
| ADDRESS | PATNA |
| JOINING DATE | 1-JUNE-12 |
| SALARY | 5000 |

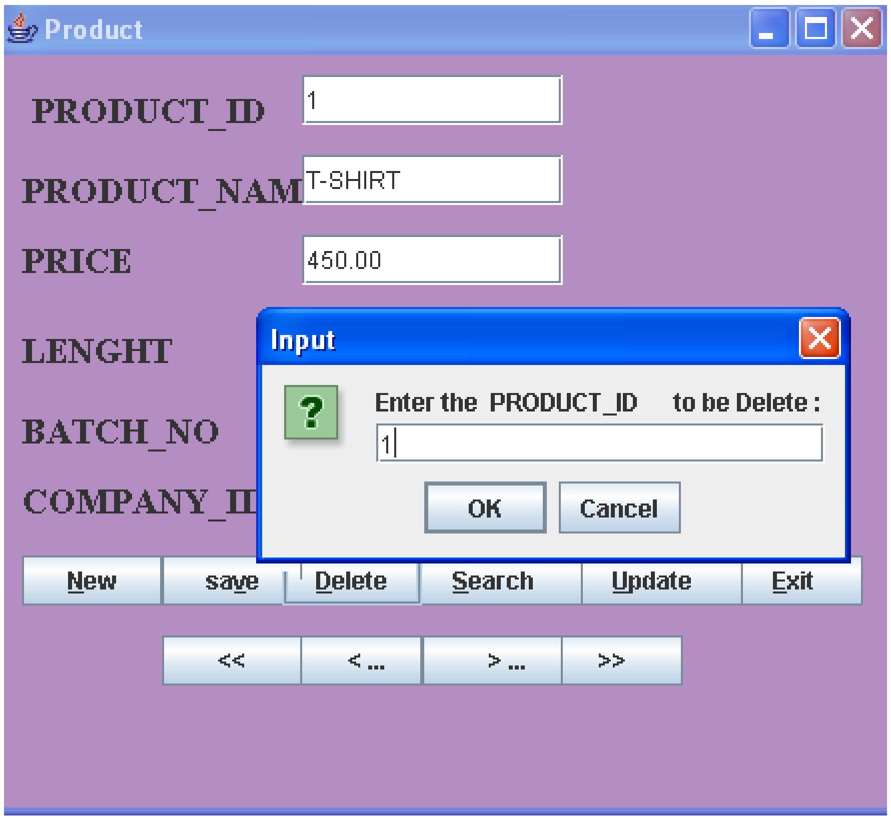
Message

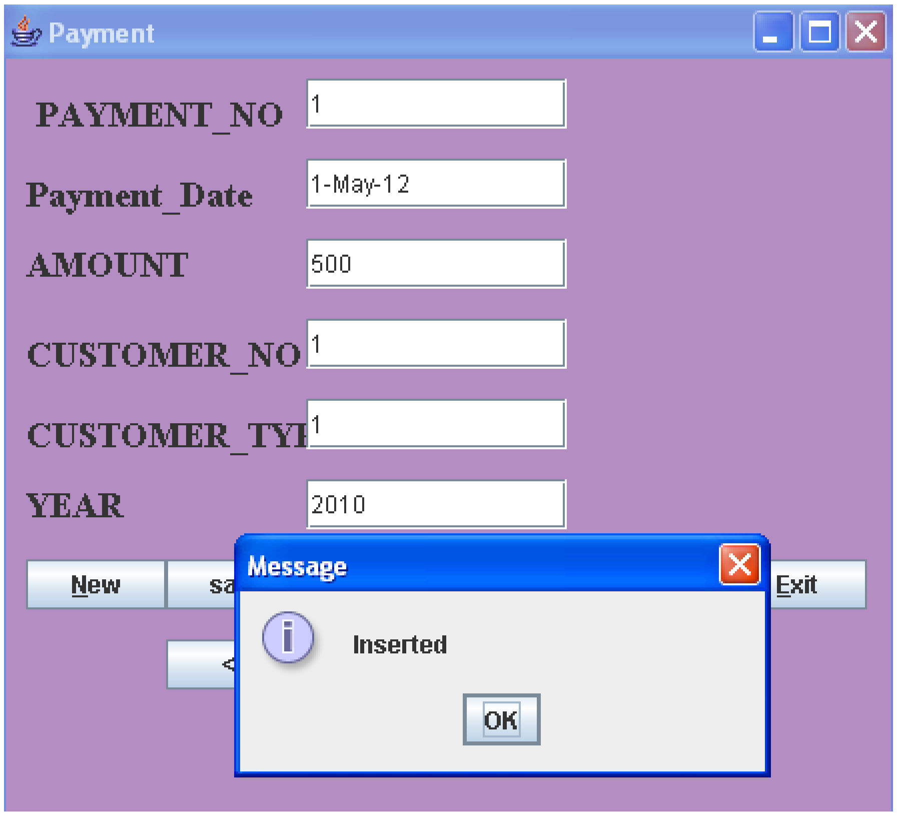
Inserted



| | |
|---|---|
| PRODUCT_ID | 1 |
| PRODUCT_NAM | T-SHIRT |
| PRICE | 450 |
| LENGHT | 3 |

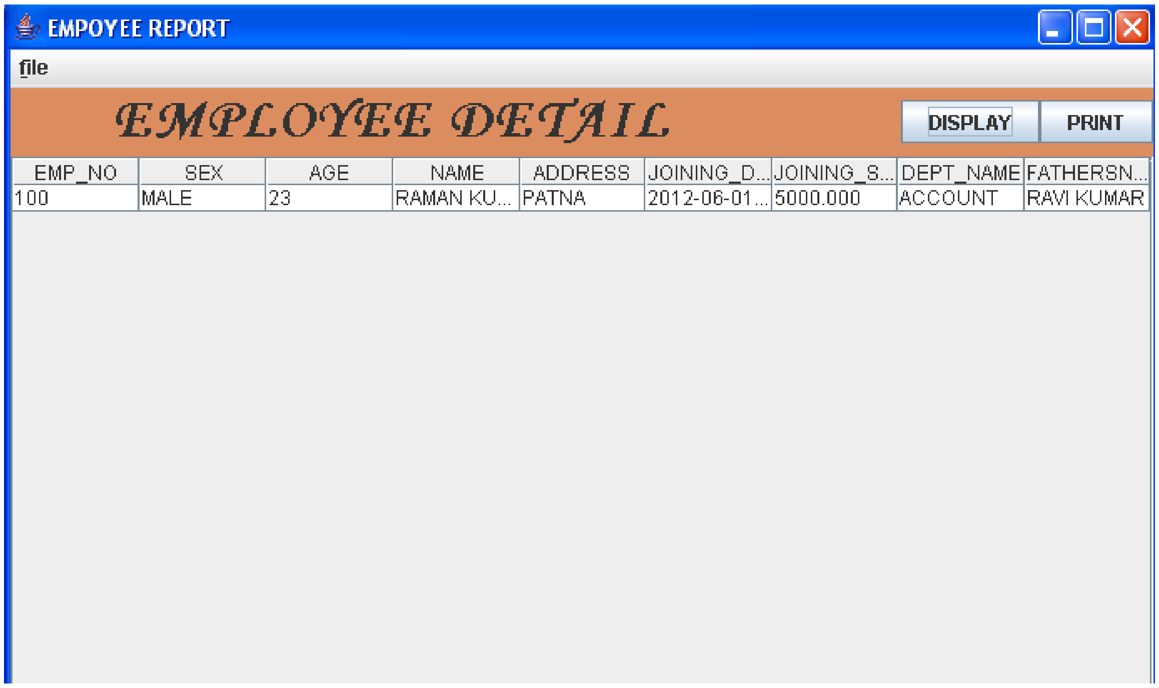
Inserted


Product
PRODUCT_ID
1
PRODUCT_NAM
T-SHIRT
PRICE
450.00
LENGHT
BATCH_NO
COMPANY_II
Input
Enter the PRODUCT_ID to be Delete :
1
OK
Cancel
New
save
Delete
Search
Update
Exit
<<
< ...
> ...
>>

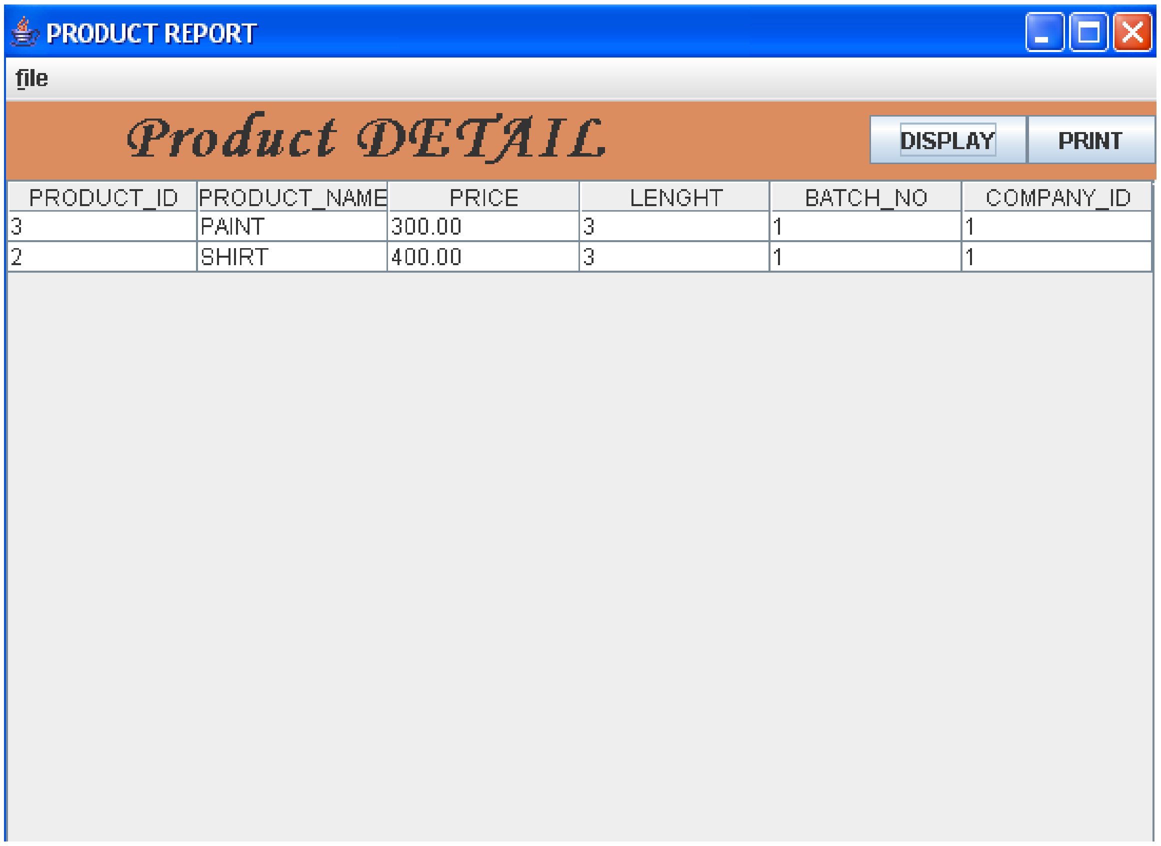


| | |
|---|---|
| PAYMENT_NO | 1 |
| Payment_Date | 1-May-12 |
| AMOUNT | 500 |
| CUSTOMER_NO | 1 |
| CUSTOMER_TYI | 1 |
| YEAR | 2010 |

Inserted



| EMP_NO | SEX | AGE | NAME | ADDRESS | JOINING_D... | JOINING_S... | DEPT_NAME | FATHERSN... |
|---|---|---|---|---|---|---|---|---|
| 100 | MALE | 23 | RAMAN KU... | PATNA | 2012-06-01... | 5000.000 | ACCOUNT | RAVI KUMAR |



| PRODUCT_ID | PRODUCT_NAME | PRICE | LENGHT | BATCH_NO | COMPANY_ID |
|---|---|---|---|---|---|
| 3 | PAINT | 300.00 | 3 | 1 | 1 |
| 2 | SHIRT | 400.00 | 3 | 1 | 1 |

# CODING

**LOGIN**

```
import java.lang.*;

import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

import java.lang.String;

import java.net.*;

import java.sql.*;


 public class LoginPassword extends JFrame

  {

   private JLabel l1,l2;

   private JTextField t1;

   private JButton change,ok,cancle;

   private JPasswordField password;


            String str,str1,str2,user,pass;

            ResultSet rs;

            Connection connection;

            Statement statement;


   public LoginPassword()

    {


      super("SUPER MARKET AUTOMATION SYSTEM");

      setLayout(null);
      l1=new JLabel("User Name");
      getContentPane().setBackground(new Color(180,141,195));
      l1.setFont(new Font("Serif",Font.BOLD,22));
      l2=new JLabel("Password");
      l2.setFont(new Font("Serif",Font.BOLD,22));
      t1=new JTextField(10);
      password=new JPasswordField(10);
     change=new JButton("Change");
       ok=new JButton("ok");
      ok.addActionListener(new okListener());
      cancle=new JButton("cancel");
    cancle.addActionListener(new cancleListener());
    connect();
     l1.setBounds(10,10,130,25);t1.setBounds(120,10,130,25);
      l2.setBounds(10,50,130,25);password.setBounds(120,50,130,25);
      change.setBounds(10,90,80,25); ok.setBounds(105,90,50,25);
      cancle.setBounds(170,90,80,25);
       add(l1);add(t1);add(l2);
       add(password);add(change);add(cancle);add(ok);
       pack();
      setSize(300,200);
      setVisible(true);
    }//End of constructor
public void connect()
            {
                        try
```

```
                 {
                 try
                 {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
connection = DriverManager.getConnection("jdbc:odbc:dsncitycart", "citycart1", "citycart");
statement = connection.createStatement();
                 }
         catch(SQLException e)
       {
    JOptionPane.showMessageDialog(null,"EXCEPTION"+e);
         }
    }
       catch(Exception e)
       {
                 JOptionPane.showMessageDialog(null,"NOT CONNECTED");
       }
}
private class okListener  implements ActionListener
{
public void actionPerformed(ActionEvent e)
         {
                 try
                 {
                 rs=statement.executeQuery("select *from  login");
                 rs.next();
                 str=rs.getString(1);
```

```
                    str1=rs.getString(2);
                    user=t1.getText();
                    pass=String.format("%s", new String(password.getPassword()));
                    if(user.compareTo(str)==0 && pass.compareTo(str1)==0)
                    {
                     MasterMenu ms=new MasterMenu();
                     dispose();
                    }
                    else
                    JOptionPane.showMessageDialog(null,"wrong password:Try Again");
                    }
                    catch(SQLException sqle)
                    {
                    }
          }
}
private class cancleListener implements ActionListener
{
          public void actionPerformed(ActionEvent e)
          {
                    try
                    {
                              dispose();
                    }
                    catch(Exception eg)
                    {}
```

```
        }
}
public static void main(String arg[])
{
        LoginPassword lp=new LoginPassword();
}
}
```

**Employee**

```
import java.lang.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.String;
import java.net.*;
import java.sql.*;

public class Employee extends JFrame
  {
   private JLabel l1,l2,l3,l4,l5,l6,l7,l8,l9;
   private JTextField t1,t2,t3,t4,t5,t6,t7,t8,t9;
   private JButton add1,del,ser,ext,upd,save,first,last,pre,next;
   int c,k,f;
   String st,str,str1,str2,str3,str4,str5,str6,str7,str8,str9;
   ResultSet rs;
   Connection connection;
   Statement statement;
```

```
public Employee()
 {


   super("Employee ");
   setLayout(null);
   getContentPane().setBackground(new Color(180,141,195));
  l1=new JLabel(" EMP_NO");
  l1.setFont(new Font("Serif",Font.BOLD,18));
  l2=new JLabel("SEX");
  l2.setFont(new Font("Serif",Font.BOLD,18));
  l3=new JLabel("AGE");
  l3.setFont(new Font("Serif",Font.BOLD,18));
  l4=new JLabel("NAME");
  l4.setFont(new Font("Serif",Font.BOLD,18));
  l5=new JLabel("ADDRESS");
  l5.setFont(new Font("Serif",Font.BOLD,18));
  l6=new JLabel("JOINING DATE");
  l6.setFont(new Font("Serif",Font.BOLD,18));
  l7=new JLabel("SALARY");
  l7.setFont(new Font("Serif",Font.BOLD,18));
  l8=new JLabel("Dept_Name");
  l8.setFont(new Font("Serif",Font.BOLD,18));
  l9=new JLabel("FATHER's NAME ");
  l9.setFont(new Font("Serif",Font.BOLD,18));
 t1=new JTextField(10);
  t2=new JTextField(10);
```

```
t3=new JTextField(10);
t4=new JTextField(10);
t5=new JTextField(10);
t6=new JTextField(10);
t7=new JTextField(10);
t8=new JTextField(10);
t9=new JTextField(10);
 add1=new JButton("New");
  add1.setToolTipText("CREATE A NEW DOCUMENT : ");
add1.addActionListener(new addListener());
                    save=new JButton("save");
                    save.setToolTipText("SAVE THE DOCUMENT : ");
                    save.addActionListener(new saveListener());
   del=new JButton("Delete");
  del.setToolTipText("DELETE THE DATA : ");
del.addActionListener(new delListener());
   upd=new JButton("Update");
  upd.setToolTipText("UPDATE THE DATA : ");
 upd.addActionListener(new updListener());
 ext=new JButton("Exit");
  ext.setToolTipText("EXIT FROM THE FORM : ");
ext.addActionListener(new extListener());
                    ser=new JButton("Search");
                    ser.setToolTipText("SEARCH THE DATA : ");
                    ser.addActionListener(new serListener());
```

```
            pre=new JButton("     <     ");
        pre.addActionListener(new preListener());


            next=new JButton("     >     ");
            next.addActionListener(new nextListener());


            first=new JButton("   <<   ");
            first.addActionListener(new firstListener());
            last=new JButton("   >>   ");
            last.addActionListener(new lastListener());
            add1.setMnemonic('N');
            del.setMnemonic('D');
            ser.setMnemonic('S');
            upd.setMnemonic('U');
            save.setMnemonic('v');
            ext.setMnemonic('E');
 connect();
l1.setBounds(10,10,250,35);t1.setBounds(150,10,130,25);
l2.setBounds(10,50,230,35);t2.setBounds(150,50,130,25);
l3.setBounds(10,90,230,25);t3.setBounds(150,90,130,25);
l4.setBounds(10,130,250,35);t4.setBounds(150,130,130,25);
l5.setBounds(10,170,230,35);t5.setBounds(150,170,130,25);
l6.setBounds(10,210,230,25);t6.setBounds(150,210,130,25);
l7.setBounds(10,250,250,35);t7.setBounds(150,250,130,25);
l8.setBounds(10,290,230,35);t8.setBounds(150,290,130,25);
l9.setBounds(10,330,230,25);t9.setBounds(150,330,130,25);
```

```
    add1.setBounds(10,370,70,25);save.setBounds(80,370,70,25);
     del.setBounds(140,370,70,25);ser.setBounds(200,370,90,25);
     upd.setBounds(280,370,90,25);ext.setBounds(360,370,70,25);
     first.setBounds(80,410,70,25);pre.setBounds(140,410,70,25);
     next.setBounds(210,410,70,25);last.setBounds(270,410,70,25);
      add(l1);add(t1);
          add(l2);add(t2);
          add(l3);add(t3);
          add(l4);add(t4);
          add(l5);add(t5);
          add(l6);add(t6);
          add(l7);add(t7);
          add(l8);add(t8);
          add(l9);add(t9);
          add(add1);add(save);
          add(del);add(ser);
                  add(upd);add(ext);
                  add(first);add(pre);
                  add(next);add(last);
    pack();
     setSize(450,410);
     setVisible(true);
   }//End of constructor

/* DATABASE CONNECTIVITY*/
public void connect()
```

```
            {


                        try
                        {
                        try
                        {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
connection = DriverManager.getConnection("jdbc:odbc:dsncitycart", "citycart1", "citycart");
statement = connection.createStatement();
                        }
catch(SQLException e)
          {
      JOptionPane.showMessageDialog(null,"EXCEPTION"+e);
            }
      }
          catch(Exception e)
          {
                        JOptionPane.showMessageDialog(null,"NOT CONNECTED");
          }
}
/* INSERT DATA IN DATABASE*/
private class saveListener implements ActionListener
            {
                        public void actionPerformed(ActionEvent e)
                        {
                                    try
                                    {
                                    str1=t1.getText();
                                    str2=t2.getText();
                                    str3=t3.getText();
                                    str4=t4.getText();
                                    str5=t5.getText();
                                    str6=t6.getText();
                                    str7=t7.getText();
                                    str8=t8.getText();
                                    str9=t9.getText();
statement.executeUpdate("insert          into                              Employee
values('"+str1+"','"+str2+"',"+str3+",'"+str4+"','"+str5+"','"+str6+"',"+str7+",'"+str8+"','"+str9+"')");
                                    statement.executeUpdate("commit");
                                    statement.executeUpdate("commit");
                                    JOptionPane.showMessageDialog(null,"Inserted");
                                    }
                                    catch(SQLException sqle)
                                    {
                                        JOptionPane.showMessageDialog(null,"could not Inserted"+sqle);
                                    }
                        }
            }
private class addListener implements ActionListener
            {
                        public void actionPerformed(ActionEvent e)
                        {
                                    try
                                    {
                                    rs=statement.executeQuery("select * from  Employee ");
                                    rs.next();
                                    t1.setText(rs.getString(1));
                                    t1.setText("");
                                    t2.setText("");
                                    t3.setText("");
                                    t4.setText("");
                                    t5.setText("");
                                    t6.setText("");
                                    t7.setText("");
                                    t8.setText("");
                                    t9.setText("");
                                    t1.requestFocus();
                                    }
                                    catch(SQLException sq)
                                    {
                                    JOptionPane.showMessageDialog(null,"could not found primary"+sq);
                                    }
                        }
            }
            private class delListener implements ActionListener
            {
                        String s;
                        public void actionPerformed(ActionEvent e)
```

```
                {
                        try
                        {
                        s=JOptionPane.showInputDialog("Enter the  EMP_NO to be Delete :");
                        statement.executeUpdate("delete from   Employee  where  EMP_NO ='"+s+" '");
                        statement.executeUpdate("commit");
                        JOptionPane.showMessageDialog(null,"delete");
                        }
                        catch(SQLException sqle)
                        {
                                JOptionPane.showMessageDialog(null,"could not Deleted"+sqle);
                        }
                }
        }
        private class updListener implements ActionListener
        {
                public void actionPerformed(ActionEvent e)
                {
                        try
                        {
                        str1=t1.getText();
                        str2=t2.getText();
                        str3=t3.getText();
                        str4=t4.getText();
                        str5=t5.getText();
                        str6=t6.getText();
```

```
                    str7=t7.getText();
                    str8=t8.getText();
                    str9=t9.getText();
                      statement.executeUpdate("update Employee set ( SEX='"+str2+"',AGE="+str3+",NAME='"+str4+"',ADDRESS='"+str5+"',JOINING_DATE='"+str6+"', JOINING_SALARY="+str7+",DEPT_NAME ='"+str8+"',FATHERSNAME='"+str9+"' where EMP_NO="+str1+")");
                         statement.executeUpdate("commit");
                         JOptionPane.showMessageDialog(null,"update");
                    }
                    catch(SQLException sqle)
                    {
                    JOptionPane.showMessageDialog(null,"could not Update"+sqle);
                    }
               }
          }
          private class extListener implements ActionListener
          {
               public void actionPerformed(ActionEvent e)
               {
                    dispose();
               }
          }
          private class serListener implements ActionListener
          {
               String t;
```

```
            public void actionPerformed(ActionEvent e)

            {

            try

                        {

              t=JOptionPane.showInputDialog("Enter the  Emp_code  to be search  :");

            rs = statement.executeQuery("select * from   Employee  where    Emp_code    ='"+t+"'");

            rs.next();


                        t1.setText(rs.getString(1));

                        t2.setText(rs.getString(2));

                        t3.setText(rs.getString(3));

                        t4.setText(rs.getString(4));

                        t5.setText(rs.getString(5));

                        t6.setText(rs.getString(6));

                        t7.setText(rs.getString(7));

                        t8.setText(rs.getString(8));

                        t9.setText(rs.getString(9));


                        JOptionPane.showMessageDialog(null,"found");

                        }

                        catch(SQLException sqle)

                        {

                                    JOptionPane.showMessageDialog(null,"not found");

                        }

            }

}
```

```
private class firstListener implements ActionListener
{
        public void actionPerformed(ActionEvent e)
        {
                try
                {
                        rs=statement.executeQuery("select *from Employee");
                        rs.next();
                        t1.setText(rs.getString(1));
                        t2.setText(rs.getString(2));
                        t3.setText(rs.getString(3));
                        t4.setText(rs.getString(4));
                        t5.setText(rs.getString(5));
                        t6.setText(rs.getString(6));
                        t7.setText(rs.getString(7));
                        t8.setText(rs.getString(8));
                        t9.setText(rs.getString(9));

                }
                catch(SQLException sqle)
                {
                        JOptionPane.showMessageDialog(null,"This is the first record");
                }
        }
}
```

```
private class lastListener implements ActionListener
{


        public void actionPerformed(ActionEvent e)
        {
                try
                {
                c=0;
                rs=statement.executeQuery("select *from  Employee ");
                while(rs.next())
                c=c+1;

                rs=statement.executeQuery("select *from Employee");
                while(c!=0)
                {
                        rs.next();
                        c=c-1;
                }

                t1.setText(rs.getString(1));
                t2.setText(rs.getString(2));
                t3.setText(rs.getString(3));
                t4.setText(rs.getString(4));
                t5.setText(rs.getString(5));
                t6.setText(rs.getString(6));
                t7.setText(rs.getString(7));
                t8.setText(rs.getString(8));
                t9.setText(rs.getString(9));


                }
                catch(SQLException sqle)
                {
                        JOptionPane.showMessageDialog(null,"could not found");
                }


        }
}


private class preListener implements ActionListener
{
        public void actionPerformed(ActionEvent e)
        {
                str=t1.getText();
                try
                {
                rs=statement.executeQuery("Select *from Employee ");
                c=0;
                while(rs.next())
                {
                        st=rs.getString(1);
                        if(str.compareTo(st)==0)
                                break;
                        c++;
                }
                rs=statement.executeQuery("Select *from Employee");
                k=0;
                while(rs.next())
                {
                        k++;
                        if(k==c)
                        break;
                }
                t1.setText(rs.getString(1));
                t2.setText(rs.getString(2));
                t3.setText(rs.getString(3));
                t4.setText(rs.getString(4));
                t5.setText(rs.getString(5));
                t6.setText(rs.getString(6));
                t7.setText(rs.getString(7));
                t8.setText(rs.getString(8));
                t9.setText(rs.getString(9));
                }
                catch(SQLException sqle)
                {
                JOptionPane.showMessageDialog(null,"This is First Record");
                }
```

```
		}
	}


	private class nextListener implements ActionListener
	{


		public void actionPerformed(ActionEvent e)
		{
			try
			{
			str=t1.getText();
			if(t1.getText().equals(""))
			JOptionPane.showMessageDialog(null,"Click on previous Button");
			else
			{
			rs=statement.executeQuery("select *from  Employee ");


			while(rs.next())
			{
				st=rs.getString(1);


				if(str.compareTo(st)==0)


					break;
			}
			rs.next();
```

```
                    t1.setText(rs.getString(1));
                    t2.setText(rs.getString(2));
                    t3.setText(rs.getString(3));
                    t4.setText(rs.getString(4));
                    t5.setText(rs.getString(5));
                    t6.setText(rs.getString(6));
                    t7.setText(rs.getString(7));
                    t8.setText(rs.getString(8));
                    t9.setText(rs.getString(9));


                    }
            }
                    catch(SQLException sqle)
                    {
                    JOptionPane.showMessageDialog(null,"This is Last Record");
                    }
            }
}
public static void main(String arg[])
{
        Employee lp=new Employee();
}



  }
```

**EMPLOYEE_REPORT**

```
import java.sql.*;

import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

import java.util.*;

import javax.swing.JTable;

import java.lang.String;

import java.io.*;

import java.lang.*;

import java.util.Vector;

import javax.swing.table.AbstractTableModel;


public class Employee_Report extends JFrame implements ActionListener

{





  String url="jdbc:odbc:dsncitycart";


  ResultsModel model =new ResultsModel();

  JTextArea status=new JTextArea(3,1);

  JScrollPane resultspane;

  JMenuBar menubar=new JMenuBar();

  JLabel lb=new JLabel("      EMPLOYEE DETAIL              ");


 JButton bt,print;
 ImageIcon log;

  JMenuItem ext=new JMenuItem("EXIT");

  Color cl=new Color(220,141,95);

  Connection connection;

  Statement statement;

  int f=0;

public Employee_Report()

 {

 super("EMPOYEE REPORT ");

 setDefaultCloseOperation(DISPOSE_ON_CLOSE);


    bt=new JButton("DISPLAY");

    print=new JButton("PRINT");

    Box box =Box.createHorizontalBox();

    box.add(lb);

    getContentPane().setBackground(cl);

    lb.setFont(new Font("Monotype Corsiva",Font.BOLD+Font.ITALIC,35));


    box.add(bt);

    box.add(print);

  bt.setToolTipText("Key SQL command and press Enter");




 add(box,BorderLayout.NORTH);

 status.setLineWrap(true);

 status.setWrapStyleWord(true);
 getContentPane().add(status,BorderLayout.SOUTH);



 JMenu file=new JMenu("file");
 file.setMnemonic('f');
 file.add(ext);
 menubar.add(file);
 setJMenuBar(menubar);
 bt.addActionListener(this);
 ext.addActionListener(this);

try

  {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   connection =DriverManager.getConnection(url,"citycart1","citycart");
   statement =connection.createStatement();

   JTable    table = new JTable(model);
  table.setAutoResizeMode(1);
   resultspane = new JScrollPane(table);
  getContentPane().add(resultspane,BorderLayout.CENTER);

          print.addActionListener(new printListener(table));
  }
   catch(ClassNotFoundException en)
  {
    System.err.println(en);
  }
  catch(SQLException sqle)
  {
  System.err.println(sqle);
 }
setBounds(0,0,400,300);
pack();
setVisible(true);

 }  // construcor closed



private class printListener implements ActionListener
{
        JTable table1;
        public printListener(JTable table)
        {
          table1=table;
        }
        public void actionPerformed(ActionEvent e)
        {
                try
                {
                        if(f==1)
```

```
                    table1.print();

                    else

                    JOptionPane.showMessageDialog(null,"At First Display Report");

                }

                catch(Exception p)

                {}


            }

        }


public void executesql()

 {

 String query=" select  * from employee  ";

 if(query==null || query.length()==0)

{

return;

}

  try

 {

 model.setResultSet(statement.executeQuery(query));

 status.setText("Resultset has " + model.getRowCount() + "rows.");

 }

 catch(SQLException sqle)

 {

 status.setText(sqle.getMessage());

 }
```

```
  }
public void actionPerformed(ActionEvent e)
{
  Object source=e.getSource();
  if(source==bt)
   {
          f=1;
   executesql();
  }
  else if(source==ext)
  {
  dispose();
  }
}
 class ResultsModel extends AbstractTableModel
 {
 private Vector<String[]> dataRows=new Vector<String[]>();
 private String[] columnNames =new String[0];
 public void setResultSet(ResultSet results)
 {
 try
 {
 ResultSetMetaData metadata=results.getMetaData();
 int columns=metadata.getColumnCount();
 columnNames=new String[columns];
   for(int i=0;i<columns;i++)
```

```
    {
     columnNames[i]=metadata.getColumnLabel(i+1);
    }
dataRows.clear();
String[] rowData;
while(results.next())
{
rowData =new String[columns];
for(int i=0;i<columns;i++)
{
rowData[i]=results.getString(i+1);
}
dataRows.addElement(rowData);
}
fireTableChanged(null);
}
catch(SQLException sqle)
{
System.err.println(sqle);
 }
  }
 public int getColumnCount()
 {
 return columnNames.length;
 }
 public int getRowCount()
 {
 return dataRows ==null ? 0 : dataRows.size();
 }
 public String getValueAt(int row,int column)
 {
 return dataRows.elementAt(row)[column];
 }
 public String getColumnName(int column)
  {
  return columnNames[column] == null ? "No Name" : columnNames[column];
  }
}
public static void main(String args[])
{
		Employee_Report r= new Employee_Report();
}
}
```

**<u>PRODUCT</u>**

```
import java.lang.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.String;
import java.net.*;
import java.sql.*;
```

```
public class Product extends JFrame
 {
  private JLabel l1,l2,l3,l4,l5,l6,l7,l8,l9;
  private JTextField t1,t2,t3,t4,t5,t6,t7,t8,t9;
  private JButton add1,del,ser,ext,upd,save,first,last,pre,next;
  int c,k,f;
  String st,str,str1,str2,str3,str4,str5,str6,str7,str8,str9;
  ResultSet rs;
  Connection connection;
  Statement statement;
  public Product()
   {
     super("Product ");
     setLayout(null);
     getContentPane().setBackground(new Color(180,141,195));
    l1=new JLabel(" PRODUCT_ID");
    l1.setFont(new Font("Serif",Font.BOLD,18));
    l2=new JLabel("PRODUCT_NAME");
    l2.setFont(new Font("Serif",Font.BOLD,18));
    l3=new JLabel("PRICE");
    l3.setFont(new Font("Serif",Font.BOLD,18));
    l4=new JLabel("LENGHT");
    l4.setFont(new Font("Serif",Font.BOLD,18));
    l5=new JLabel("BATCH_NO");
    l5.setFont(new Font("Serif",Font.BOLD,18));
    l6=new JLabel("COMPANY_ID");
```

```
l6.setFont(new Font("Serif",Font.BOLD,18));
t1=new JTextField(10);
t2=new JTextField(10);
t3=new JTextField(10);
t4=new JTextField(10);
t5=new JTextField(10);
t6=new JTextField(10);
    add1=new JButton("New");
 add1.setToolTipText("CREATE A NEW DOCUMENT : ");
            add1.addActionListener(new addListener());
            save=new JButton("save");
            save.setToolTipText("SAVE THE DOCUMENT : ");
            save.addActionListener(new saveListener());
    del=new JButton("Delete");
 del.setToolTipText("DELETE THE DATA : ");
            del.addActionListener(new delListener());
    upd=new JButton("Update");
 upd.setToolTipText("UPDATE THE DATA : ");
            upd.addActionListener(new updListener());
    ext=new JButton("Exit");
 ext.setToolTipText("EXIT FROM THE FORM : ");
            ext.addActionListener(new extListener());
            ser=new JButton("Search");
            ser.setToolTipText("SEARCH THE DATA : ");
            ser.addActionListener(new serListener());
            pre=new JButton("    <    ");
```

```
      pre.addActionListener(new preListener());
            next=new JButton("     >     ");
            next.addActionListener(new nextListener());
            first=new JButton("   <<   ");
            first.addActionListener(new firstListener());
            last=new JButton("   >>   ");
            last.addActionListener(new lastListener());
            add1.setMnemonic('N');
            del.setMnemonic('D');
            ser.setMnemonic('S');
            upd.setMnemonic('U');
            save.setMnemonic('v');
            ext.setMnemonic('E');
     connect();
l1.setBounds(10,10,250,35);t1.setBounds(150,10,130,25);
l2.setBounds(10,50,230,35);t2.setBounds(150,50,130,25);
l3.setBounds(10,90,230,25);t3.setBounds(150,90,130,25);
l4.setBounds(10,130,250,35);t4.setBounds(150,130,130,25);
l5.setBounds(10,170,230,35);t5.setBounds(150,170,130,25);
l6.setBounds(10,210,230,25);t6.setBounds(150,210,130,25);
add1.setBounds(10,250,70,25);save.setBounds(80,250,70,25);
del.setBounds(140,250,70,25);ser.setBounds(200,250,90,25);
upd.setBounds(280,250,90,25);ext.setBounds(360,250,70,25);
first.setBounds(80,290,70,25);pre.setBounds(140,290,70,25);
next.setBounds(210,290,70,25);last.setBounds(270,290,70,25);
    add(l1);add(t1);
```

```
        add(l2);add(t2);
        add(l3);add(t3);
        add(l4);add(t4);
        add(l5);add(t5);
        add(l6);add(t6);
                add(add1);add(save);
                add(del);add(ser);
                add(upd);add(ext);
                add(first);add(pre);
                add(next);add(last);
    pack();
    setSize(450,410);
    setVisible(true);
   }//End of constructor
* DATABASE CONNECTIVITY*/
public void connect()
        {
                try
                {
                try
                {
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                        connection = DriverManager.getConnection("jdbc:odbc:dsncitycart", "citycart1",
"citycart");
                statement = connection.createStatement();
                }
```

```
                    catch(SQLException e)
          {
        JOptionPane.showMessageDialog(null,"EXCEPTION"+e);
            }
        }
          catch(Exception e)
          {
                    JOptionPane.showMessageDialog(null,"NOT CONNECTED");
          }


}
/* INSERT DATA IN DATABASE*/
private class saveListener implements ActionListener
        {
                    public void actionPerformed(ActionEvent e)
                    {
                            try
                            {
                            str1=t1.getText();
                            str2=t2.getText();
                            str3=t3.getText();
                            str4=t4.getText();
                            str5=t5.getText();
                            str6=t6.getText();
                    statement.executeUpdate("insert          into                                    Product
values("+str1+",'"+str2+"',"+str3+","+str4+","+str5+","+str6+")");
```

```
                    statement.executeUpdate("commit");
                    statement.executeUpdate("commit");
                    JOptionPane.showMessageDialog(null,"Inserted");
                    }
                    catch(SQLException sqle)
                    {
                        JOptionPane.showMessageDialog(null,"could not Inserted"+sqle);
                    }
                }
            }
                private class addListener implements ActionListener
            {
                public void actionPerformed(ActionEvent e)
                {
                    try
                    {
                    rs=statement.executeQuery("select * from  Product ");
                    rs.next();
                    t1.setText(rs.getString(1));
                    t1.setText("");
                    t2.setText("");
                    t3.setText("");
                    t4.setText("");
                    t5.setText("");
                    t6.setText("");
                    t1.requestFocus();
                    }
                    catch(SQLException sq)
                    {
                    JOptionPane.showMessageDialog(null,"could not found primary"+sq);
                    }
            }
    }

    private class delListener implements ActionListener
    {
            String s;
            public void actionPerformed(ActionEvent e)
            {
                    try
                    {
                            s=JOptionPane.showInputDialog("Enter the  PRODUCT_ID      to be Delete
:");



                            statement.executeUpdate("delete  from     Product    where    PRODUCT_ID
='"+s+" '");
                            statement.executeUpdate("commit");
                            JOptionPane.showMessageDialog(null,"delete");
                    }
                    catch(SQLException sqle)
                    {
                        JOptionPane.showMessageDialog(null,"could not Deleted"+sqle);
                    }
                }
        }

        private class updListener implements ActionListener
        {
                public void actionPerformed(ActionEvent e)
                {
                    try
                    {
                    str1=t1.getText();
                    str2=t2.getText();
                str3=t3.getText();
                    str4=t4.getText();
                    str5=t5.getText();
                str6=t6.getText();
      statement.executeUpdate("update Product set ( PRODUCT_NAME ='"+str2+"',
   PRICE="+str3+",LENGHT="+str4+",BATCH_NO="+str5+",COMPANY_ID="+str6+" where
   PRODUCT_ID ="+str1+")");
                        statement.executeUpdate("commit");
                        JOptionPane.showMessageDialog(null,"update");
                    }
                    catch(SQLException sqle)
                    {
                        JOptionPane.showMessageDialog(null,"could not Update"+sqle);
                        }
                }
        }
        private class extListener implements ActionListener
        {
                public void actionPerformed(ActionEvent e)
                {
                        dispose();
                }

        }

        private class serListener implements ActionListener
        {
                String t;
                public void actionPerformed(ActionEvent e)
                {

                        try
                        {
                          t=JOptionPane.showInputDialog("Enter the  Emp_code  to be search  :");

                        rs = statement.executeQuery("select * from   Product   where    Emp_code
  ='"+t+"'");
                        rs.next();
                        t1.setText(rs.getString(1));
                        t2.setText(rs.getString(2));
                        t3.setText(rs.getString(3));
                        t4.setText(rs.getString(4));
                        t5.setText(rs.getString(5));
                        t6.setText(rs.getString(6));
                        JOptionPane.showMessageDialog(null,"found");
                        }
                        catch(SQLException sqle)
                        {
                                JOptionPane.showMessageDialog(null,"not found");
                        }
                }
        }
private class firstListener implements ActionListener
{
        public void actionPerformed(ActionEvent e)
        {
                try
                {
                        rs=statement.executeQuery("select *from Product");
                        rs.next();
                        t1.setText(rs.getString(1));
                        t2.setText(rs.getString(2));
                        t3.setText(rs.getString(3));
                        t4.setText(rs.getString(4));

                        t5.setText(rs.getString(5));

                        t6.setText(rs.getString(6));


                }

                catch(SQLException sqle)

                {

                        JOptionPane.showMessageDialog(null,"This is the first record");

                }

        }

}

private class lastListener implements ActionListener

{

        public void actionPerformed(ActionEvent e)

        {

                try

                {

                c=0;

                rs=statement.executeQuery("select *from  Product ");

                while(rs.next())

                c=c+1;

                rs=statement.executeQuery("select *from Product");

                while(c!=0)

                {

                        rs.next();

                        c=c-1;
```

```
                    }
                    t1.setText(rs.getString(1));
                    t2.setText(rs.getString(2));
                    t3.setText(rs.getString(3));
                    t4.setText(rs.getString(4));
                    t5.setText(rs.getString(5));
                    t6.setText(rs.getString(6));
                    }
                    catch(SQLException sqle)
                    {
                              JOptionPane.showMessageDialog(null,"could not found");
                    }
          }
}
private class preListener implements ActionListener
{
          public void actionPerformed(ActionEvent e)
          {
                    str=t1.getText();
                    try
                    {
                    rs=statement.executeQuery("Select *from Product ");
                    c=0;
                    while(rs.next())
                    {
                              st=rs.getString(1);
```

```
                        if(str.compareTo(st)==0)
                                break;
                        c++;
                }
                rs=statement.executeQuery("Select *from Product");
                k=0;
                while(rs.next())
                {
                        k++;
                        if(k==c)
                        break;
                }
                t1.setText(rs.getString(1));
                t2.setText(rs.getString(2));
                t3.setText(rs.getString(3));
                t4.setText(rs.getString(4));
                t5.setText(rs.getString(5));
                t6.setText(rs.getString(6));
                }
                catch(SQLException sqle)
                {
                JOptionPane.showMessageDialog(null,"This is First Record");
                }
        }
}
private class nextListener implements ActionListener
```

```
{
        public void actionPerformed(ActionEvent e)
        {
                try
                {
                str=t1.getText();
                if(t1.getText().equals(""))
                JOptionPane.showMessageDialog(null,"Click on previous Button");
                else
                {
                rs=statement.executeQuery("select *from  Product ");
                while(rs.next())
                {
                        st=rs.getString(1);
                        if(str.compareTo(st)==0)
                                break;
                }
                rs.next();
                t1.setText(rs.getString(1));
                t2.setText(rs.getString(2));
                t3.setText(rs.getString(3));
                t4.setText(rs.getString(4));
                t5.setText(rs.getString(5));
                t6.setText(rs.getString(6));
                }
        }
```

```
                    catch(SQLException sqle)
                    {
                    JOptionPane.showMessageDialog(null,"This is Last Record");
                    }
          }
}
public static void main(String arg[])
{
          Product lp=new Product();
}


  }
```

**PAYMENT**

```
import java.lang.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.String;
import java.net.*;
import java.sql.*;


public class Payment extends JFrame
  {
   private JLabel l1,l2,l3,l4,l5,l6,l7,l8,l9;
   private JTextField t1,t2,t3,t4,t5,t6,t7,t8,t9;
   private JButton add1,del,ser,ext,upd,save,first,last,pre,next;
```

```
int c,k,f;

String st,str,str1,str2,str3,str4,str5,str6,str7,str8,str9;

ResultSet rs;

Connection connection;

Statement statement;

public Payment()

 {


    super("Payment ");

    setLayout(null);

    getContentPane().setBackground(new Color(180,141,195));

   l1=new JLabel(" PAYMENT_NO");

   l1.setFont(new Font("Serif",Font.BOLD,18));

   l2=new JLabel("Payment_Date");

   l2.setFont(new Font("Serif",Font.BOLD,18));

   l3=new JLabel("AMOUNT");

   l3.setFont(new Font("Serif",Font.BOLD,18));

   l4=new JLabel("CUSTOMER_NO");

   l4.setFont(new Font("Serif",Font.BOLD,18));

   l5=new JLabel("CUSTOMER_TYPE");

   l5.setFont(new Font("Serif",Font.BOLD,18));

   l6=new JLabel("YEAR");

   l6.setFont(new Font("Serif",Font.BOLD,18));

  t1=new JTextField(10);

   t2=new JTextField(10);

   t3=new JTextField(10);
```

```
t4=new JTextField(10);

t5=new JTextField(10);

t6=new JTextField(10);


 add1=new JButton("New");

 add1.setToolTipText("CREATE A NEW DOCUMENT : ");

            add1.addActionListener(new addListener());

            save=new JButton("save");

            save.setToolTipText("SAVE THE DOCUMENT : ");

            save.addActionListener(new saveListener());


 del=new JButton("Delete");

 del.setToolTipText("DELETE THE DATA : ");

            del.addActionListener(new delListener());


 upd=new JButton("Update");

 upd.setToolTipText("UPDATE THE DATA : ");

            upd.addActionListener(new updListener());


 ext=new JButton("Exit");

 ext.setToolTipText("EXIT FROM THE FORM : ");

            ext.addActionListener(new extListener());


            ser=new JButton("Search");

            ser.setToolTipText("SEARCH THE DATA : ");

            ser.addActionListener(new serListener());
```

```
                pre=new JButton("     <     ");
            pre.addActionListener(new preListener());


                next=new JButton("     >     ");
                next.addActionListener(new nextListener());
                first=new JButton("   <<   ");
                first.addActionListener(new firstListener());
                last=new JButton("   >>   ");
                last.addActionListener(new lastListener());
                add1.setMnemonic('N');
                del.setMnemonic('D');
                ser.setMnemonic('S');
                upd.setMnemonic('U');
                save.setMnemonic('v');
                ext.setMnemonic('E');


           connect();
       l1.setBounds(10,10,250,35);t1.setBounds(150,10,130,25);
       l2.setBounds(10,50,230,35);t2.setBounds(150,50,130,25);
       l3.setBounds(10,90,230,25);t3.setBounds(150,90,130,25);
       l4.setBounds(10,130,250,35);t4.setBounds(150,130,130,25);
       l5.setBounds(10,170,230,35);t5.setBounds(150,170,130,25);
       l6.setBounds(10,210,230,25);t6.setBounds(150,210,130,25);
       add1.setBounds(10,250,70,25);save.setBounds(80,250,70,25);
       del.setBounds(140,250,70,25);ser.setBounds(200,250,90,25);
```

```
      upd.setBounds(280,250,90,25);ext.setBounds(360,250,70,25);
      first.setBounds(80,290,70,25);pre.setBounds(140,290,70,25);
      next.setBounds(210,290,70,25);last.setBounds(270,290,70,25);


          add(l1);add(t1);
          add(l2);add(t2);
          add(l3);add(t3);
          add(l4);add(t4);
          add(l5);add(t5);
          add(l6);add(t6);
                  add(add1);add(save);
                  add(del);add(ser);
                  add(upd);add(ext);
                  add(first);add(pre);
                  add(next);add(last);
      pack();
      setSize(450,410);
      setVisible(true);
     }//End of constructor
  /* DATABASE CONNECTIVITY*/
  public void connect()
          {
                  try
                  {
                  try
                  {
```

```
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            connection = DriverManager.getConnection("jdbc:odbc:dsncitycart", "citycart1", "citycart");
            statement = connection.createStatement();
                        }
catch(SQLException e)
          {
       JOptionPane.showMessageDialog(null,"EXCEPTION"+e);
            }
      }
         catch(Exception e)
         {
                        JOptionPane.showMessageDialog(null,"NOT CONNECTED");
         }
}
/* INSERT DATA IN DATABASE*/
private class saveListener implements ActionListener
            {
                        public void actionPerformed(ActionEvent e)
                        {
                                    try
                                    {
                                    str1=t1.getText();
                                    str2=t2.getText();
                                    str3=t3.getText();
                                    str4=t4.getText();
                                    str5=t5.getText();
```

```
                    str6=t6.getText();


                    statement.executeUpdate("insert             into                                    Payment
values("+str1+",'"+str2+"','"+str3+"',"+str4+","+str5+","+str6+")");
                    statement.executeUpdate("commit");
                    statement.executeUpdate("commit");
                    JOptionPane.showMessageDialog(null,"Inserted");
                    }
                    catch(SQLException sqle)
                    {
                        JOptionPane.showMessageDialog(null,"could not Inserted"+sqle);
                    }
                }
            }



            private class addListener implements ActionListener
            {
                public void actionPerformed(ActionEvent e)
                {
                    try
                    {
                    rs=statement.executeQuery("select * from  Payment ");
                    rs.next();
                    t1.setText(rs.getString(1));
                    t1.setText("");
                    t2.setText("");
                    t3.setText("");
                    t4.setText("");
                    t5.setText("");
                    t6.setText("");

                    t1.requestFocus();
                    }
                    catch(SQLException sq)
                    {
                    JOptionPane.showMessageDialog(null,"could not found primary"+sq);
                    }
                }
            }

            private class delListener implements ActionListener
            {
                String s;
                public void actionPerformed(ActionEvent e)
                {
                    try
                    {
            s=JOptionPane.showInputDialog("Enter the  Payment_ID      to be Delete :");
            statement.executeUpdate("delete from   Payment  where  Payment_ID      ='"+s+" '");
            statement.executeUpdate("commit");
            JOptionPane.showMessageDialog(null,"delete");
```

```
                    }
                    catch(SQLException sqle)
                    {
                        JOptionPane.showMessageDialog(null,"could not Deleted"+sqle);
                    }
                }
            }
private class updListener implements ActionListener
            {
                public void actionPerformed(ActionEvent e)
                {
                    try
                    {
                    str1=t1.getText();
                    str2=t2.getText();
                    str3=t3.getText();
                    str4=t4.getText();
                    str5=t5.getText();
                    str6=t6.getText();
   statement.executeUpdate("update            Payment        set            (        P_DATE
='"+str2+"',P_AMOUNT="+str3+",CUSTOMER_NO='"+str4+"',CUSTOMER_TYPE='"+str5+"',YEAR='"
+str6+"' where   Payment_ID    ="+str1+")");
                        statement.executeUpdate("commit");
                        JOptionPane.showMessageDialog(null,"update");
                    }
                    catch(SQLException sqle)
                    {
                    JOptionPane.showMessageDialog(null,"could not Update"+sqle);
                    }
            }
    }
    private class extListener implements ActionListener
    {
            public void actionPerformed(ActionEvent e)
            {
                    dispose();
            }
    }
    private class serListener implements ActionListener
    {
            String t;
            public void actionPerformed(ActionEvent e)
            {


                    try
                    {
            t=JOptionPane.showInputDialog("Enter the  Emp_code  to be search  :");
    rs = statement.executeQuery("select * from   Payment  where    Emp_code    ='"+t+"'");
            rs.next();
                    t1.setText(rs.getString(1));
                    t2.setText(rs.getString(2));
                    t3.setText(rs.getString(3));
```

```
                        t4.setText(rs.getString(4));
                        t5.setText(rs.getString(5));
                        t6.setText(rs.getString(6));


                        JOptionPane.showMessageDialog(null,"found");
                        }
                        catch(SQLException sqle)
                        {
                                JOptionPane.showMessageDialog(null,"not found");
                        }
                }
        }
private class firstListener implements ActionListener
{
        public void actionPerformed(ActionEvent e)
        {
                try
                {
                        rs=statement.executeQuery("select *from Payment");
                        rs.next();
                        t1.setText(rs.getString(1));
                        t2.setText(rs.getString(2));
                        t3.setText(rs.getString(3));
                        t4.setText(rs.getString(4));
                        t5.setText(rs.getString(5));
                        t6.setText(rs.getString(6));
                }
                catch(SQLException sqle)
                {
                        JOptionPane.showMessageDialog(null,"This is the first record");
                }
        }
}
private class lastListener implements ActionListener
{


        public void actionPerformed(ActionEvent e)
        {
                try
                {
                c=0;
                rs=statement.executeQuery("select *from  Payment ");
                while(rs.next())
                c=c+1;

                rs=statement.executeQuery("select *from Payment");
                while(c!=0)
                {
                        rs.next();
                        c=c-1;
```

```
				}
				t1.setText(rs.getString(1));
				t2.setText(rs.getString(2));
				t3.setText(rs.getString(3));
				t4.setText(rs.getString(4));
				t5.setText(rs.getString(5));
				t6.setText(rs.getString(6));
				}
				catch(SQLException sqle)
				{
					JOptionPane.showMessageDialog(null,"could not found");
				}

		}
}

private class preListener implements ActionListener
{
		public void actionPerformed(ActionEvent e)
		{
				str=t1.getText();
				try
				{
				rs=statement.executeQuery("Select *from Payment ");
				c=0;
				while(rs.next())
```

```
{
        st=rs.getString(1);
        if(str.compareTo(st)==0)
                break;
        c++;
}
rs=statement.executeQuery("Select *from Payment");
k=0;
while(rs.next())
{
        k++;
        if(k==c)
        break;
}
t1.setText(rs.getString(1));
t2.setText(rs.getString(2));
t3.setText(rs.getString(3));
t4.setText(rs.getString(4));
t5.setText(rs.getString(5));
t6.setText(rs.getString(6));


}
catch(SQLException sqle)
{
JOptionPane.showMessageDialog(null,"This is First Record");
}
```

```
        }
}


private class nextListener implements ActionListener
{


        public void actionPerformed(ActionEvent e)
        {
                try
                {
                str=t1.getText();
                if(t1.getText().equals(""))
                JOptionPane.showMessageDialog(null,"Click on previous Button");
                else
                {
                rs=statement.executeQuery("select *from  Payment ");

                while(rs.next())
                {
                        st=rs.getString(1);

                        if(str.compareTo(st)==0)

                                break;
                }
                rs.next();
```

```
                    t1.setText(rs.getString(1));
                    t2.setText(rs.getString(2));
                    t3.setText(rs.getString(3));
                    t4.setText(rs.getString(4));
                    t5.setText(rs.getString(5));
                    t6.setText(rs.getString(6));


                    }
            }
                    catch(SQLException sqle)
                    {
                    JOptionPane.showMessageDialog(null,"This is Last Record");
                    }
            }
    }
    public static void main(String arg[])
    {
            Payment lp=new Payment();
    }



     }
```

# OPTIMIZATION OF CODE

Optimization of code is also very important for the good software. Without optimization it is very difficult to debug the syntactical and logical errors present in the code. If the code of the software will be optimized then it becomes very easy to find out the errors in stipulated time.

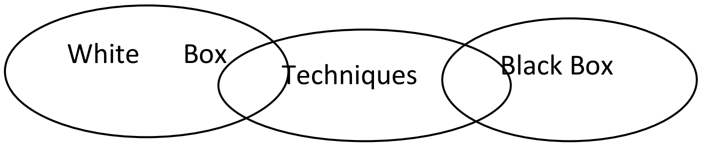
Every step has been taken to optimize the coding. The efforts has been made to modularize the whole working of the software by which it will be easy to locate the errors in time saving manner. Any future changing in the requirement of the company can also be implemented if the code of the software will be optimized.  By making the modules of the important activities as Yes, No checking etc. The same modules have been called in different places of need such as test report, add/update database etc.

# VALIDATION CHECKS

There are following validation checks are performed in my present software:

1) Any field in any database cannot be empty.
2) Name and address field of customer and employee will only accept data of its type and up to its length.
3) Enno, Enum will be unique. Duplicate values will not be accepted.
4) Amount field in product module will be checked by its numeric function.
5) Salary field of employee database will always take numeric values.
6) Before modify/delete/recall/pack any record it is searched in the available data files. If data is not available "Data not found" message is prompted on the user screen.
7) At the time of recalling/packing any record it is searched that the particular record has been deleted or not.
8) While adding any new record in product module all the records are searched from the beginning of the database and if the matching entry is found then the record is updated in lieu of adding new record in the database.
9) Password must be correct for any user to enter into the software.
10) Textbox will only hold entries of mentioned format and data type.
11) All the entries are case sensitive.
12) Price cannot be Negative.

# TESTING TECHNIQUES



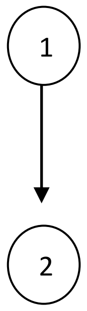
**Testing Method**

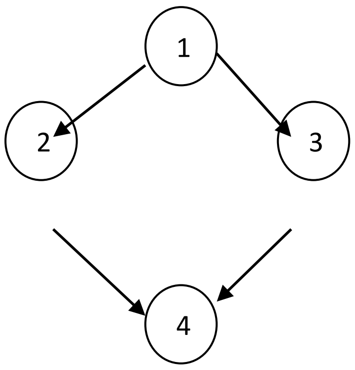
During testing phase I went through these testing techniques.--

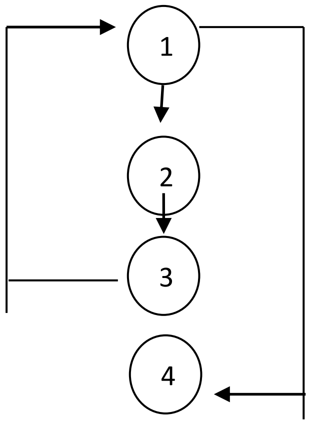
## 1.) White Box Testing

White-box testing, sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using white-box testing method, the software can be derived test cases that

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structure to ensure their validity.
- During, white box Testing, I perform these testing techniques:







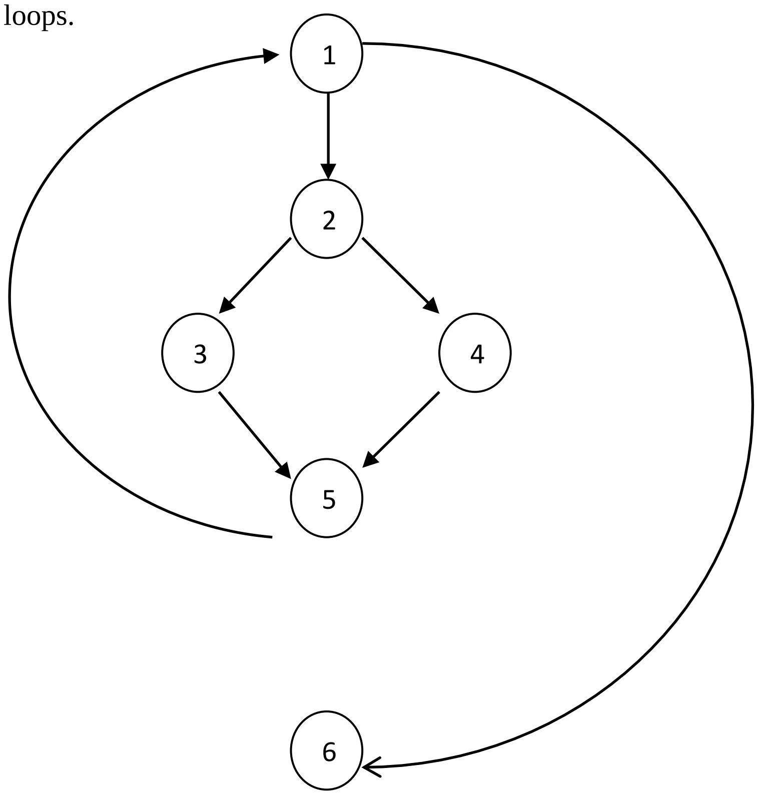
Sequence Selection

## Data Flow Testing

The data flow testing method selects test paths of a program according to the locations of definitions and uses of variables in the program. Data flow testing strategies are useful for selecting test paths of program containing nested if and loop statements.

## Loop Testing

It is also a white-box testing technique that focuses exclusively on the validity of loop constructs. Four different classes of loops can be defined: ---Simple loops, concatenated loops, nested loops, and unstructured loops.



## Control Flow Graph

## 2.) Black Box Testing

Black box testing looks into functional requirements of the software. It enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. It is not an alternative to white box techniques. Rather, it is a complementary approach that may uncover a different class of errors than white box methods.

Black box testing tries to find error in the following categories:

1. Incorrect or missing functions
2. Interface errors.
3. Errors in data structure or external data base access
4. Performance errors
5. Initialization and termination errors.

While white box testing is performed early in the testing process, black box testing is applied during later stage of testing. Because black box testing purposely disregards control structure, attention is focused on the information domain.

### GRAPH BASED TESTING METHODS:

The first step in black box testing is to understand the objects that are modeled in software and the relationship that connects these objects. Once this need has been accomplished, the next step is to define a series of test that verify, "all objects have the expected relationship to one another."

### EQUIVALENCE PARTITIONING:

Equivalence partitioning is a black-box technique that divides the input domain of a program into classes of data from which test cases can be derived. An ideal test cases single handedly uncovers a class of errors that might otherwise require many cases to execute before the general error is discovered. Equivalence partitioning strives to define a test case that uncovers a classes of error, thereby reducing the total number of test cases that must be developed. Test design for equivalence partitioning is based on an evaluation of equivalence classes for an input condition.

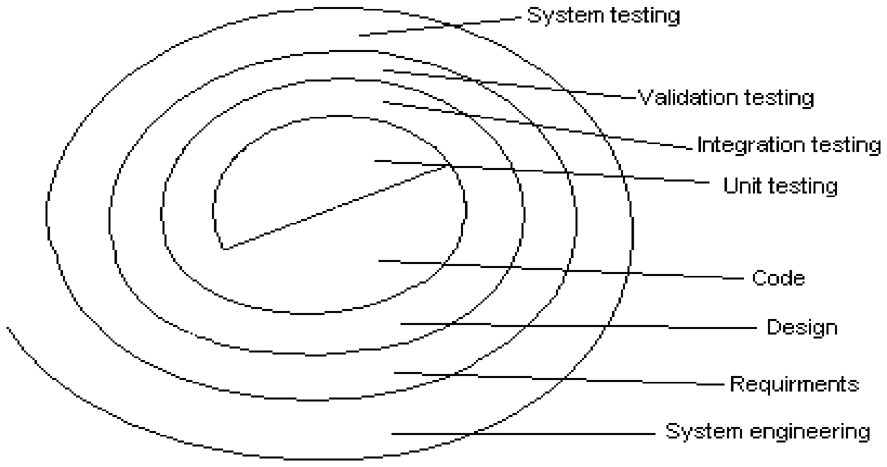
**BOUNDARY VALUE ANALYSIS:**

Boundary value analysis is a test case design technique that complements equivalence partitioning. Rather than selecting any element of equivalence classes, BVA leads to the selection of test cases at the “edge” of class.

By using these testing methods in the present software many defects and errors present has been removed and it looks a good software with no or minimum errors.

# TESTING STRATEGIES

At last I went through the testing strategy where I tested my whole system. The strategy to do this provides a road map that describes the steps to be conducted as part of testing, when these steps are planned and then undertaken, and how much effort, time, and resources will be required. A number of software testing strategies have been proposed. Because, for determining the quality of software it is necessary to follow these testing strategy before its implementation. It is one of the most crucial steps in the software development life cycle. Testing is required for determining whether the system is performing as per the requirement or not.



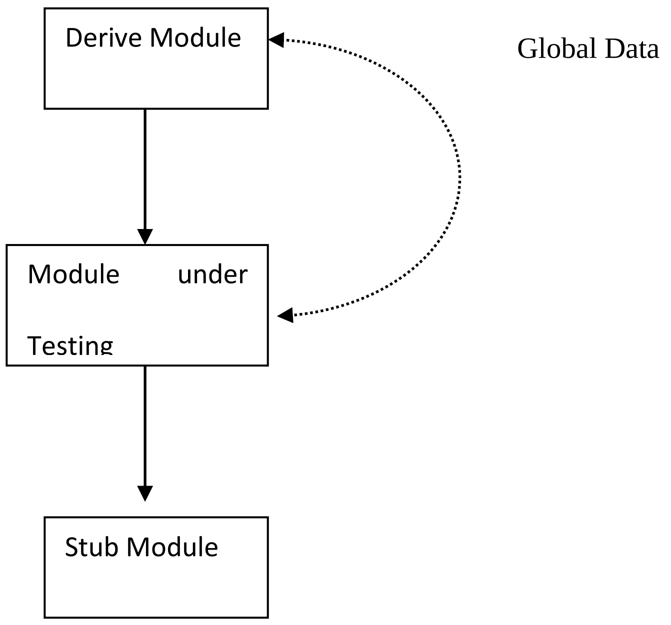
I follow these testing strategies to test my whole System.

- Unit Testing
- Integration Testing
- Validation Testing
- System Testing

## Unit Testing

- Unit testing focuses verification efforts on the smallest unit of software design-the software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The unit test is white-box oriented, and the step can be conducted in parallel for multiple components.

- This level of testing is done just after the coding of the module completed. Generally programmers do this testing itself during the development process.
- Here in my software different modules have been tested and it has helped me to rectify the errors.



## Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and built a program structure that has been dictated by design. This testing was performed after the integration of modules. At this stage whole system was tested as a single module and it was found correct.

## Validation Testing

Software validation is achieved through a series of black-box tests that demonstrate conformity with requirements. A test plan outlines the classes of tests to be conducted and a test procedure defines specific test cases that will be demonstrated conformity with requirements. Both the plan and procedure are designed to ensure that all functional requirements are satisfied, all behavioral characteristics are achieved, all performance requirements are attained and human-engineered and other requirements are met. At this level of testing whole system was tested, whether it is performing up to the mark or not. Here it was found that the performance of the system is optimum.

## System Testing

At this level of testing not only the software but also some other aspects were considered. Here whole system was tested on different hardware and software platforms.

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that system elements have been properly integrated and perform allocated functions. The purpose of system testing is to consider all the likely variations to which it will be subjected and then push the system to its limits. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully activated. Another reason for system testing is its utility as a user-oriented vehicle before implementation.

Using these testing strategies, which I have been used for the present software. As any software cannot be 100% error free, this software is also not an exception and it is possible to find out errors in the future (at the time of application). These errors will be freed whenever it comes.
So finally I would like to say that after the overall testing process, I found that my system is running well, I tried to make my system better.

| | |
|---|---|
| ⇨ | Critical Path |
| → | Activity |
| ⇢ | Dummy Activity |
| ○ | Node |

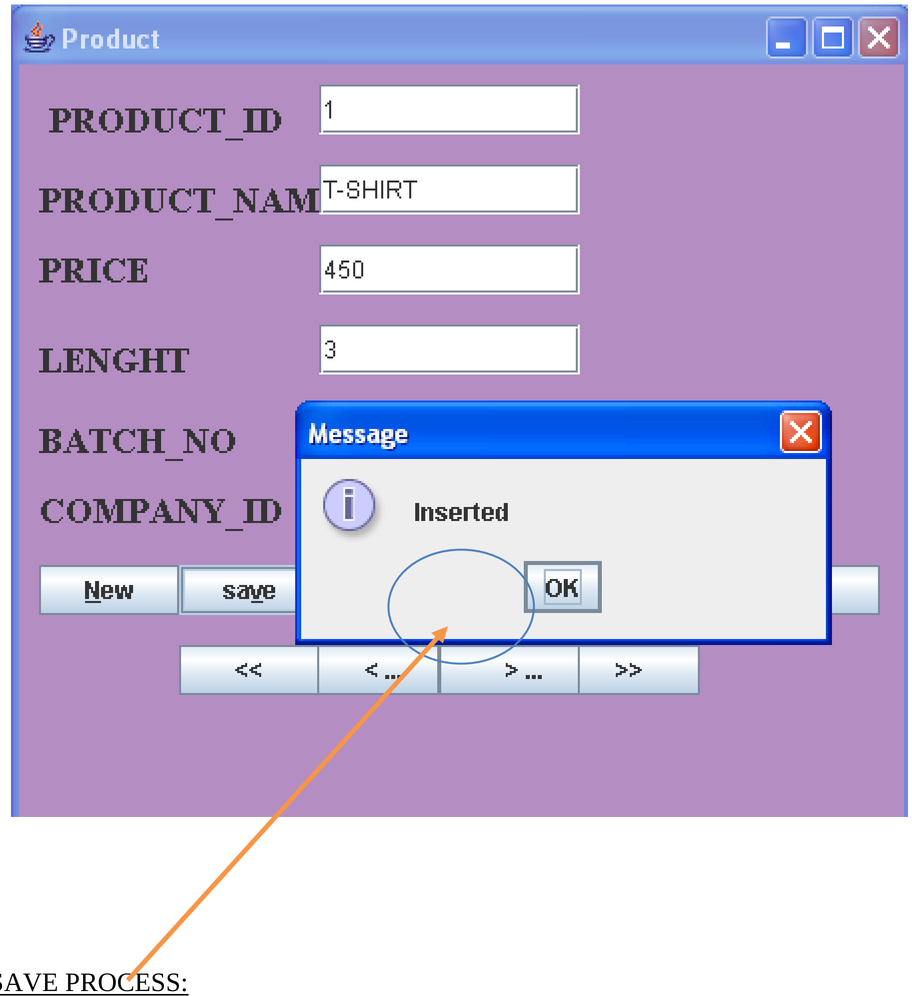
## ALGORITHM OF PRODUCT MASTER

Step 1:- Input the data in the Text Field.

Step 2:- Click on the Save Button

Step 3: Internally "Insert Query is Executed" if we insert the appropriate data.

Step 4:- If data is input successfully, then display the message "inserted" other wise display the Error message.



1
T-SHIRT
450
3
Inserted

SAVE PROCESS:

```
private class saveListener implements ActionListener
        {
                public void actionPerformed(ActionEvent e)
                {
                        try
                        {
```

```
                        str1=t1.getText();
                        str2=t2.getText();
                        str3=t3.getText();
                        str4=t4.getText();
                        str5=t5.getText();
                        str6=t6.getText();

                                statement.executeUpdate("insert into    Product
                    values("+str1+",'"+str2+"',"+str3+","+str4+","+str5+","+str6+")");
                        statement.executeUpdate("commit");
                        statement.executeUpdate("commit");
                        JOptionPane.showMessageDialog(null,"Inserted");
                        }
                        catch(SQLException sqle)
                        {
                                JOptionPane.showMessageDialog(null,"could not Inserted"+sqle);
                        }
                }
        }
```

# IMPLEMENTATION AND MAINTENANCE

The study has been done to know the working of "**SUPER MARKET AUTOMATION SYSTEM"** and the sample data collected. The executive bodies as well as general body members have been consulted to get the sample test data. Apart from this some of beneficiary and donators have been interviewed to get the required information. I have visited some of the project sites where this organization is working to collect the sample data. The data, thus, collected is entered into the system and processed. The processed information in the form of MIS Report is provided to the members of executive body. Proper modification has been done in the application as per the modification in the report processed and the suggestion of members. The process has been repeated several times till the members of executive body are fully satisfied. The test case or test data has been shown in the forms of MIS Report. It can be seen in the annexure enclosed.

First of all the application has been thoroughly tested by applying the test data. After that it got approved from the executive body to be implemented. Users operational manual has been prepared and proper training to the users has been provided. Once the user is well acquainted with the new computerized system it was installed and operational zed along with the existing manual system. Both the computerized system and the manual system will run side by side for at least six months. During this period if any nonconformity will be found out, that will be fully solved and incorporated. If the new system does not contradict the existing system can be fully adopted.

After system task will not over a close eye site will be kept on the functioning of the new system. If any type of problem will find out that will be properly eradicated. This type of software maintenance will be done or not less than one hour after complete operationalization of the system. During this period any type of maintenance needed or required by the management will be done. The organization will depute a technical person to manage the new system and to operate smoothly.

# EVALUATION

Evaluation of the system is performed to identify its strengths and weaknesses. Evaluation measures the system's performance against pre-defined requirements. It determines how well the system continues to meet performance specifications. It also provides information to determine whether major re-design or modification is required.

A post-implementation review is an evaluation of a system in terms of the extent to which the system accomplishes stated objectives and actual project costs exceed initial estimates. It is usually a review of major problems that need converting and those that surfaced during the implementation phase.

The actual evaluation can occur along any of the following dimension, which I have performed in my system.

From Operational point of view there is Assessment of the manner in which the system function, including ease of use, response time, suitability of information formats overall reliability, and level of utilization are performed.

From Organizational point of view there is Identification and measurement of benefits to the organization in such areas as financial concerns (cost, revenue, and profit), operational efficiency, and competitive impact. Includes impact on internal and external information flows are performed.

From User Manager Assessment point of view there is Evaluation of the attitudes between staff and manager within the organization are performed.

The present software has been also evaluated for knowing its actual working in the practical situation. During evaluation period the functioning of the software has been found good.

## SCOPE OF FUTURE APPLICATIONS

In this age of fast growing technology and implementation of the latest technology there is always a scope of future improvement. Also there can be several ways to achieve the target. The main motto is to implement the new concept in the field of development by using the tips and tricks related to the modern, primitive and advance age styles and that makes the difference. There is nothing to do without future applications.

So in this project there are also many scopes for future application. This project can be used for any educational organization. By adding some more features it can be also used by large organization Super Market and mega mat.

Some future scopes of this project are given below: -

- The database can act as future reference for the organization and product's reference.
- Similar garment store can use the system for their own use.
- Employees and staff can be trained in some of the relevant parts so that they apply and understand the software – this makes them more aware of use of IT in day to day official dealings.
- A larger complete user-friendly system can be made in future using this application software.
- Networking support.

## LIMITATIONS OF THE PROJECT

It is honestly tried to satisfy all the requirements of super market in this Project. But there are some limitations of the project. These limitations are given below-

- The system can't take responsibility of unforeseen errors damages and losses due to calamities.
- Changes made in the database may take some time to update or modify.
- Internal processes, working of the SUPER MARKET like daily routine, employee schedule are not shown by the system.
- System is not platform independent may require some specific configuration to work on the system. The system is windows compatible, may not run on other systems.
- Data base is Operating System dependent.
- SGA size: O/S dependent, typically 2-4 GB for 32-bit O/S, > 4 GB for 64 bit O/S .
- No Networking facility.

# BIBLIOGRAPHY

**Books:-**

- The Complete Reference Java (Tata McGraw Hill, Seventh Edition)
- Software Engineering (McGraw Hill, Sixth Edition)
- Data Base Management System (Bipin Chand  Desai)
- Ivan Bayross –Oracle
- Study Materials of IGNOU – JAVA  ,SAD ,RDBMS and DBMS , Software Engineering etc.

**Magazine:-**

**DIGIT**