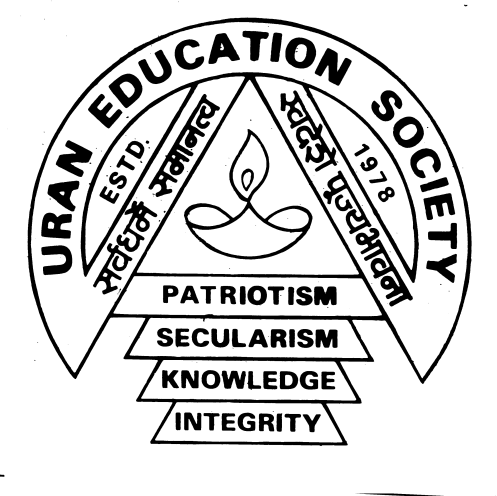
****

**URAN EDUCATION SOCIETY’S**

**COLLEGE OF MANAGEMENT & TECHNOLOGY**

**(Bori, Uran-400702)**

**Project Report on**

**GYM MANAGEMENT**

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENT FOR THE AWARD OF DEGREE OF

BACHOLAR OF SCIENCE IN INFORMATION TECHNOLOGY

SUBMITTED BY

**Tejash Jagannath Mhatre & Ashish Chandrakant Chavan**

UNDER THE GUIDANCE OF

**PROF. ROHAN R GHOSALKAR**

BACHELOR OF INFORMATION TECHNOLOGY

SEMESTER-VI

**2012-2013**

**Certificate**

This is to certify that the experimental work entered in this journal as per the syllabus in B.Sc. (Information Technology) TY Semester-VI prescribed by University of Mumbai for the project of Gym Management was successfully done by Tejash Jagannath Mhatre & Ashish Chandrakant Chavan in the computer laboratory of Uran Education Society’s College of Management & Technology, Bori Uran during the academic year 2012-2013.

Exam seat no: \_\_\_SEM vI\_\_\_\_\_

Project Guide: External Examiner:

(Internal)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Coordinator of B.Sc.IT) Principal

**Acknowledgement**

I am delighted to take an opportunity in introducing my software based product entitled as “Gym Management”. It has been a project of hard work and commences strain. At the outset I wish to express my sincere and heartfelt gratitude towards my Professors for the wholehearted support and guidance, the completion of this project would have been possible.

I would especially thank Prof. Rohan Ghosalkar and Principal who helped me towards providing the needed reference, which of intense value.

Lastly I would like to thank all those who directly and indirectly helped in completion of this project.

**Date:**

**Place:**

**Completed by**

**(TEJASH.J.MHATRE**

**&**

**ASHISH.C.CHAVAN)**

**PREFACE**

This project is developed with an aim to provide an improved Information and Management System for the “Gym management”.

The Software provides help in effective management of deals with the distributors and customers and the employee of the organisation with additional features of security and reduced data redundancy and inconsistency.

The special features are the Invoice Details, Search Criteria and the generation of Reports and Bills.

The following documentation provides an insight of existing system, its limitations, its scope of improvement and logic with its intricate details to achieve those improvements.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Title** | **Page No.** |
| **1** | **Introduction** | **6** |
| **2** | **Gantt Chart** | **8** |
| **3** | **Organizational Overview:** | **10** |
|  | History | **11** |
|  | Organizational Chart. | **12** |
|  | Present System. | **13** |
|  | Limitations of present system. | **13** |
| **4** | **Proposed System.** | **14** |
| **5** | **Software-Hardware Specifications.** | **17** |
| **6** | **System Design:** | **20** |
|  | Entity Relationship Diagram | **21** |
|  | Event Table. | **22** |
|  | Use Case Diagram | **24** |
|  | Class Diagram | **27** |
|  | Context Level Diagram | **30** |
|  | Data flow Diagram | **31** |
|  | System Flowchart | **37** |
|  | Sequence Diagram. | **38** |
|  | Collaboration Diagram | **40** |
|  | Structure Diagram | **42** |
|  | Component Diagram | **44** |
|  | Deployment Diagram | **46** |
|  | Activity Digram | **48** |
| **7** | **System Coding:** | **52** |
|  | Program List | **53** |
|  | Tables Used | **54** |
|  | Report List | **58** |
| **8** | **Program Documentation:** | **59** |
|  | System Level Coding(snap shots) | **60** |
|  | System level coding | **70** |
| **9** | **Testing** | **87** |
| **10** | **Report** | **91** |
| **11** | **Feature Enhancement** | **93** |
| **12** | **Conclusion.** | **95** |
| **13** | **Biblography** | **97** |

**Part 1**

***INTRODUCTION***

**INTODUCTION-**

Several years ago, when the organization had to go its work manually at all. Therefore the organization faced problems in terms of time consuming, every time handle paper work, etc.

To solve these problems most of the organizations shared their work using computers.

Therefore they needed the help of the computerized system.

Here we presented the system which will solve almost all problems.

Our system provides various facilities such as:

* User friendly interface.
* Database Security.
* Proper and well Organizational Reports.

**Part *2***

***GANTT CHART***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Dec** | **Jan** | **Feb** | **Mar** |
|  | **1 2 3 4** | **1 2 3 4** | **1 2 3 4** | **1 2 3 4** |
| 1. **Planning Phase**   Define the problem  Work breakdown structure  Product Project Schedule |  |  |  |  |
| 1. **Analysis Phase**   Gather Information  Define System Requirement  Objectives and Feasibility |  |  |  |  |
| 1. **Design Phase**   Design Database  Design System  Flow |  |  |  |  |
| 1. **Coding Phase**   Write Code for  Individual  Component  Integrate the components |  |  |  |  |
| 1. **Testing Phase**   Unit Testing  System Testing  Integration Testing |  |  |  |  |
| 1. **Implementation Phase** |  |  |  |  |

**Part *3***

***ORGANIZATIONAL***

***OVERVIEW***

**HISTORY**

This is the Gym Management system and it will give you all the information about the fitness and registration of customer. This system gives you all kind of exercise as you want. This organization provides you the information about diet and Exercise. Here you can make the body Healthy and Fit .You can learn the many types of exercises.

Any person can join this Gym Management and make him/her Healthy and Fit. In this application all Exercise are explain in simple textual format.

It is mainly design for a normal person who wish make his/her body fit and perfect but can’t afford a trainer or instructor. For a brief and detailed information the diagrams like ERD ,DFD’S ,Flow chart , structure charts, etc., have been include. Data dictionary is also provided for better understanding of the software.

Using the software one can easily maintain his/her body by mere button clicks. Ease of navigation and user friendliness is some of the other salient features of “GYM MANAGEMENT”.

**ORGANIZATIONAL CHART**

**GYM MANAGEMENT SYSTEM**

CUSTOMER

SUPPLIER

EMPLOYEE

**PRESENT SYSTEM**

All work is done manually and records are safely stored in registers. If any record is to be searched, a large volume of information has to be traced. This is time consuming. The data stored is not properly organized.

**LIMITATION OF PRESENT SYSTEM**

* In Present system each an every record is maintains in registers, so there will be lot of redundancy in maintaing records.
* Also there is no security due to all records are maintain in register.
* Modification of one record causes to other records related to that record, so work becomes very critical, so some time data loss will be occcur.
* Storage of information is costly.
* Require knowledge about this system.
* M more time required for reading & adding records in register
* Searching any old record of any job done or damaged required more time due to critical system of register.

**Part *4***

***PROPOSED***

***SYSTEM***

**Proposed System**

1. This software is very co-operative with the users and more of user-friendly nature.
2. For security purpose there is password facility and unless we give this password we cannot open or create entry.
3. It uses a normalized data structure minimizing redundancy & inconsistencies to large extent.
4. Automation of various calculation & formulations, for very quick & less time consuming reports.
5. It is reliable since formulations are computerized and presence validation checks leads to error free results.
6. The proposed System eliminates or provide an alternative to the need of an instructor.
7. There is a specific list which displays various exercise for various body parts.
8. Keep track of user using this application.

**APPLICATION OF THE PROPOSED SYSTEM**

1. Give better functionality.
2. Easy to work on the system.
3. Give instant result.
4. Generation of salary report.
5. Updating the customer batch very Quickly.
6. Updating the list of fixed assets Records.
7. Period wise customer fees records.

**ADVANTAGES :-**

* With the help of computerized system, all the records of order, payment , and bill are checked in easy way.
* Operator has to put just ID-no of customer, product and then all information is displayed on the screen.
* If any incorrect data is typed then system throws user understandable errors or messages.
* Using computerized software, we find particular records of customer without any time delay.
* Operator easily handles the Ozone fitness software by computerized system.
* All calculations are done automatically. Thus there are no any errors in calculation.
* Operator can see the data reports of each customer, payment and order.
* Records of the customers, products, order and payment details are safely stored in computer, which cannot be mismatched.

**Part 5**

***SOFTWARE,***

***HARDWARE***

***SPECIFICATION***

**SOFTWARE – SPECIFICATIONS**

Software requirements for this system are as listed follows:

* FRONTEND : C#. NET
* BACKEND : SQL 2005
* OPERATING SYSTEM : Windows XP

**HARDWARE SPECIFICATIONS**

Minimum hardware requirements for this System are listed below:

* CPU Type Pentium-IV.
* Base Memory 640kb.
* Extended Memory 15360kb.
* Cache Memory 256kb.
* CPU Clock 150 MHz
* Display Type EGA/VGA.
* Hard Disk 100MB.

**Part *6***

***SYSTEM  
DESIGN***

**ENTITY RELATIONSHIP DIAGRAM**

**GYM**

**PRODUCT**

**CUSTOMER**

**STOCK**

**STAFF**

**SUPPLER**

**OWNER**

**EVENT TABLE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Event** | Trigger | **Source** | **Activity** | **Response** | **Destination** |
| 1) | **Customer enquires for product.** | Get information about product | **Customer** | **Enquaring about product** | **Availability status** | **Customer** |
| **2)** | **Registration** | **Enquiary** | **Customer** | **Registrating customer** | **Registration confirmed** | **Customer** |
| **3)** | **Checking availability** | **Get Product details** | **Staff** | **Checking availability of product** | **Availability status** | **Staff** |
| **4)** | **Order product** | **Availability status** | **Staff** | **Ordering product** | **Order recieved** | **Staff** |
| **5)** | **Paying advance** | **Registration** | **Customer** | **Paying advance** | **Advance paid** | **Staff** |
| **6)** | **Order fullfillment** | **Product ordered** | **Staff** | **Providing product/service** | **Confirmation** | **Customer** |
| **7)** | **Generate Customer Bill** | **Providing product/service** | **Staff** | **Generating Bill** | **Bill Generated** | **Staff** |
| **8)** | **Pay bill** | **Receiving bill** | **Customer** | **Paying bill** | **Bill recieved** | **Staff** |
| **9)** | **Generate Reciept** | **Bill paid** | **Staff** | **Generating**  **Reciept** | **Reciept Generated** | **Customer** |
| **10)** | **Check stock** | **Check inventry** | **Staff** | **Checking availability of stock** | **Stock checked** | **Staff** |
| **11)** | **Management place order to supplier** | **New order** | Staff | **Ordering product** | **Order confirmation** | **supplier** |
| **12)** | **Management pay bill to supplier** | **Product Ordered** | **Staff** | **Making**  **payment details** | **Payment recieved** | **supplier** |
| **13)** | **Generate Report** | **Check Record** | **Staff** | **Generating report** | **Report Generated** | **Staff** |

**UML Diagrams**

**UML Diagram Classification—Static, Dynamic, and Implementation**

* **Static:** The static characteristic of a system is essentially the structural aspect of the system. The static characteristics define what parts the system is made up of.
* **Dynamic:** The behavioral features of a system; for example, the ways a system behaves in response to certain events or actions are the dynamic characteristics of a system.
* **Implementation:** The implementation characteristic of a system is an entirely new feature that describes the different elements required for deploying a system.

**The UML diagrams that fall under each of these categories are**:

* **Static**
  + Use case diagram
  + Class diagram
* **Dynamic**
  + Object diagram
  + State diagram
  + Activity diagram
  + Sequence diagram
  + Collaboration diagram
* **Implementation**
  + Component diagram
  + Deployment diagram

**USE CASE DIAGRAM**

The Use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The Use case diagram shows which actors interact with each use case.

**Elements of a Use Case Diagram**

**Actors:** An actor portrays any entity (or entities) that perform certain roles in a given system. The different roles the actor represents are the actual business roles of users in a given system. An actor in a use case diagram interacts with a use case.



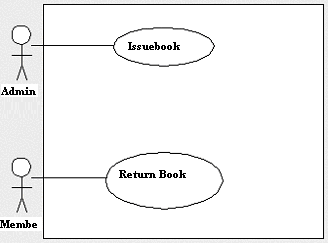
*an actor in a use case diagram*

**Use case:** A use case in a use case diagram is a visual representation of a distinct business functionality in a system. The key term here is "distinct business functionality." To choose a business process as a likely candidate for modeling as a use case, you need to ensure that the business process is discrete in nature

2

*use cases in a use case diagram*

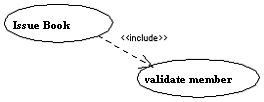
**System boundary:** A system boundary defines the scope of what a system will be.

*:*

*a use case diagram depicting the system boundary of a library management application*

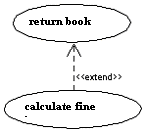
**Relationships in Use Cases**

**Include:** When a use case is depicted as using the functionality of another use case in a diagram, this relationship between the use cases is named as an *include* relationship.

*:*

*An example of an include relationship*

**Extend:** In an *extend* relationship between two use cases, the child use case adds to the existing functionality and characteristics of the parent use case



*An example of an extend relationship.*

USE CASE DIAGRAM

Customer

Staff

Supplier

**CLASS DIAGRAM**

By definition, a class diagram is a diagram showing a collection of classes and interfaces, along with the collaborations and relationships among classes andinterfaces

**Elements of a Class Diagram**

A class diagram is composed primarily of the following elements that represent the system's business entities:

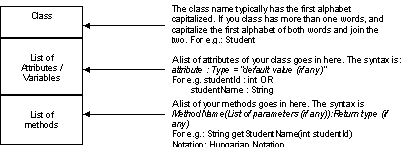
***Class*:** A class represents an entity of a given system that provides an encapsulated implementation of certain functionality of a given entity. These are exposed by the

class to other classes as *methods*.

Apart from business functionality, a class also has properties that

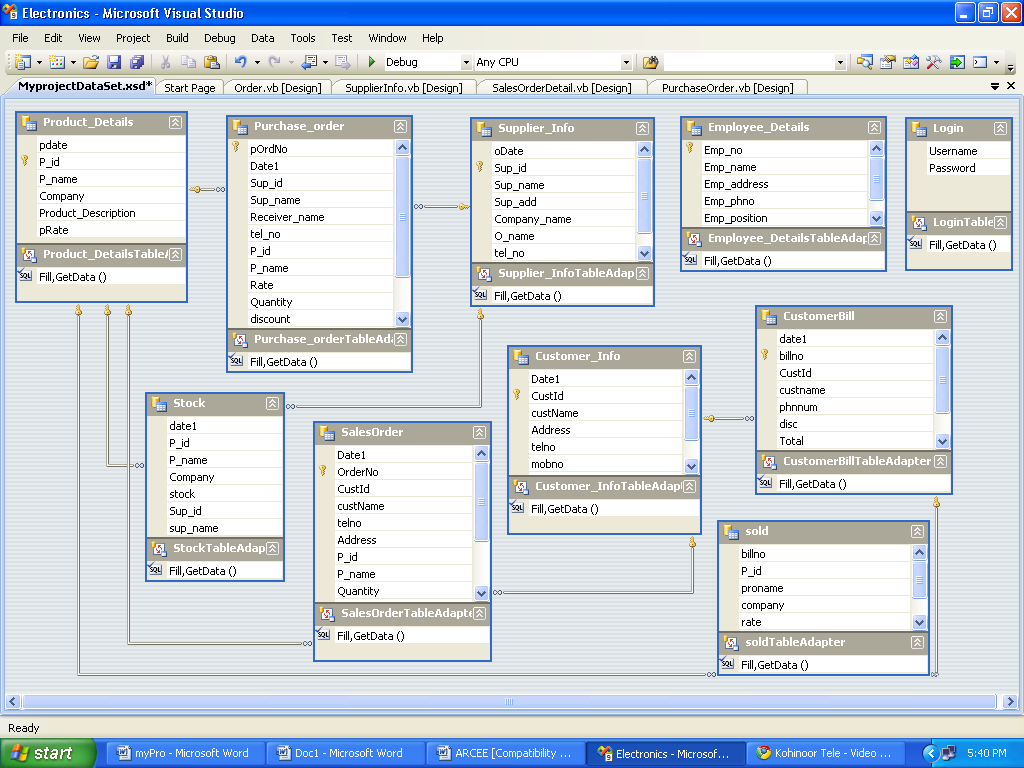
reflect unique features of a class. The properties of a class are called *attributes*. Simply put, individual members of a family of our family tree example are analogous to classes in a class diagram

A class is represented by a rectangle. The following diagram shows a typical class in a class diagram:



*Figure 4.1.1—the structure of a class*

**Class Diagram**

****

**System Context Diagram**

System Context Diagram are [diagrams](http://en.wikipedia.org/wiki/Diagram) used in [systems design](http://en.wikipedia.org/wiki/Systems_design) to represent the more important external actors that interact with the system at hand.

The objective of a system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of system requirements and constraints"

System context diagrams are related to [Data Flow Diagram](http://en.wikipedia.org/wiki/Data_Flow_Diagram), and show the interactions between a system and other actors with which the system is designed to interface. System context diagrams can be helpful in understanding the context which the system will be part of.

Context diagrams are used early in a project to get agreement on the scope under investigation. Context diagrams are typically included in a requirements document. These diagrams must be read by all project stakeholders and thus should be written in plain language, so the stakeholders can understand items within the document.

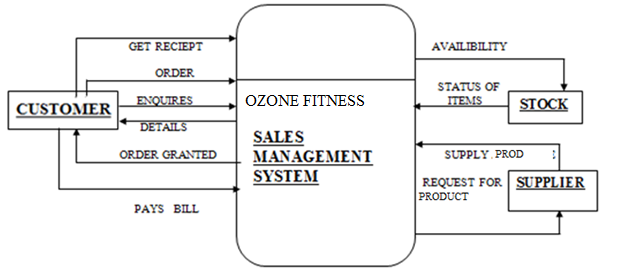
***Building blocks***

Context diagrams can be developed with the use of two types of building blocks:

* ***labeled boxes***: one in the center representing the system and around it multiple boxes for each external actor
* ***Relationship***: labeled lines between the entities and system

For example, "customer places order." Context diagrams can also use many different drawing types to represent external entities. They can use [ovals](http://en.wikipedia.org/wiki/Oval_(geometry)), [stick figures](http://en.wikipedia.org/wiki/Stick_figure), [pictures](http://en.wikipedia.org/wiki/Picture), [clip art](http://en.wikipedia.org/wiki/Clip_art) or any other representation to convey meaning. Decision trees and data storage are represented in system flow diagram

**CONTEXT-LEVEL DIAGRAM:**

****

Gym Management

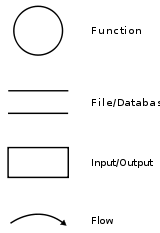
**Data flow diagram**

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an [information system](http://en.wikipedia.org/wiki/Information_system). DFDs can also be used for the [visualization](http://en.wikipedia.org/wiki/Data_visualization) of [dataprocessing](http://en.wikipedia.org/wiki/Data_processing)

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a [flowchart](http://en.wikipedia.org/wiki/Flowchart), which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD)

Data-flow-diagram-notation

[](http://en.wikipedia.org/wiki/File:Data-flow-diagram-notation.svg)

**DATA FLOW DIAGRAM**

**4**

Create new order

**3**

Lookup item

Availability

**1.LIST OF CUSTOMER RECORDS**

**1**

Customer Information

Create New Information

**Management**

**Customer**

Update

Information

**2**

Supplier Information

**Supplier**

**Management**

Update

**Stock**

Information

Create New

Update

Information

Item inquiry

**Customer**

**Stock**

Item

Availability

Details

**Product**

**Customer**

**Remaining**

**Order Details**

Create new order

**Customer**

**OT**

**5**

Record fulfillment details

Record

**Customer**

Detail

**OWEN**

**6**

Updating Stock

Updating

**Sales person**

**Stock**

Stock

Stock Updated

**Customer**

**Bill**

**OT**

Report

**7**

Produce order

Summaryreport

**Sales person**

Details

**DFDFragments**

1. **Add New Customer Entry**

**2**

Make New Customer

Entry

User

New **Customer** Entry

Customer confirmation

Customer Details

**Add New Supplier Entry**

**1**

Make

New Supplier

Entry

Supplier

Supplier

User

New **Supplier** Entry

Supplier confirmation

***Process No.:*** *2*

**Process Name:** New supplier

**Inflow:** Supplier Detail

**Outflow:** Selected Supplier

**Process Description (Structure English):**

**If** the supplier is new **then**

Add new supplier details

Make new supplier entry

Send the supplier details to supplier

Store the selected supplier in to the supplier table

Supplier gives the confirmation to the user

**End if**

* + 1. **Add New Product Entry**

**3**

Make New

Product Entry

New **Product** Entry

Product Details

User

Product confirmation

1. **Make New Purchase Entry**

**4**

Update

Purchase

Payment

New Purchase Entry

User

Product Details

Purchase confirmation

Customer Details

1. **Make New Sales Entry**

**4**

Update

Stock

Payment

New Sales Entry

User

Product Details

Sales confirmation

Customer Details

1. **Update Supplier Entry**

**5**

Update

Supplier

Entry

Update **Supplier**

Supplier

Supplier

Change Confirmation

1. **Update Customer Entry**

**6**

Update

Customer

Entry

Update **Customer**

Customer

User

hangeConfirmat change confirmation

1. **Generate bill report**

User

**8**

The Sales

Report Generated

Sales\_master

Produce Sales Report

Sales Report

**SYSTEM** – **FLOWCHART**

**SUPPLIER**

**DATABASE**

**GYM MANAGEMENT**

**SYSTEM**

**CUSTOMER**

**DATABASE**

**ORDER**

**DATABASE**

**PRODUCT**

**DATABASE**

**BILL**

**DATABASE**

**STOCK DATABASE**

**MAINTAIN SUPPLIER DETAILS**

**MAINTAIN CUSTOMER DETAILS**

**CUSTOMER ORDER DETAIL**

**MAINTAIN PRODUCT DETAILS**

**BILL CREATION PROGRAM**

**MAINTAIN PRODUCT DETAILS**

**ORDER REPORT**

**BILL REPORT**

**STOCK REPORT**

**CUSTOMER**

**REPORT**

**SUPPLIER**

**REPORT**

**REMAINING ORDER REPORT**

**Sequence diagram**

A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behavior of a system.

A Sequence diagram is two-dimensional in nature. On the horizontal axis, it shows the life of the object that it represents, while on the vertical axis, it shows the sequence of the creation or invocation of these objects. Because it uses class name and object name references, the Sequence diagram is very useful in elaborating and detailing the dynamic design and the sequence and origin of invocation of objects. Hence, the Sequence diagram is one of the most widely used dynamic diagrams in UML.

**Elements of a Sequence diagram**

A Sequence diagram consists of the following behavioral elements:

|  |  |
| --- | --- |
| **Element and its description** | **Symbol** |
| **Object:** The primary element involved in a sequence diagram is an Object—an instance of a class. A Sequence diagram consists of sequences of interaction among different objects over a period of time. An object is represented by a named rectangle. The name to the left of the ":" is the object name and to its right is the class name. | UML08T1 |
| **Message:** The interaction between different objects in a sequence diagram is represented as messages. A message is denoted by a directed arrow. Depending on the type of message, the notation differs. In a Sequence diagram, you can represent simple messages, special messages to create or destroy objects, and message responses. | UML08T2 |

**SEQUENCE DIAGRAM**

Customer Staff Shop Supplier

1. Enquiry
2. Give Information
3. Registration
4. orderproduct

5. Check Availability

6. Availabile

7. Inform Availability

8. Pay advance

9.Provide product

10. Generate Bill

11. Pay bill

12. Generate reciept

13. Give receipt

14. Reciept received

15. Update Record

16. Check Stock

17. Order New Product

18. Generate Bill

1919. Pay Bill

20. Generate report

**COLLABORATION DIAGRAM**

A Collaboration diagram is very similar to a Sequence diagram in the purpose it achieves; in other words, it shows the dynamic interaction of the objects in a system. A distinguishing feature of a Collaboration diagram is that it shows the objects and their association with other objects in the system apart from how they interact with each other. The association between objects is not represented in a Sequence diagram.

A Collaboration diagram is easily represented by modeling objects in a system and representing the associations between the objects as links. The interaction between the objects is denoted by arrows. To identify the sequence of invocation of these objects, a number is placed next to each of these arrows.

**Defining a Collaboration diagram**

A sophisticated modeling tool can easily convert a collaboration diagram into a sequence diagram and the vice versa. Hence, the elements of a Collaboration diagram are essentially the same as that of a Sequence diagram.

**Elements of a Collaboration diagram**

A Collaboration diagram consists of the following elements:

|  |  |
| --- | --- |
| **Element and its description** | **Symbol** |
| **Object:** The objects interacting with each other in the system. Depicted by a rectangle with the name of the object in it, preceded by a colon and underlined. | UML09T1 |
| **Relation/Association:** A link connecting the associated objects. Qualifiers can be placed on either end of the association to depict cardinality. | UML09T2 |
| **Messages:** An arrow pointing from the commencing object to the destination object shows the interaction between the objects. The number represents the order/sequence of this interaction. | UML09T3 |

**COLLABORATION DIAGRAM**

14.Reciept received

11.Pay bill

8.Pay advance

4.Order product

1.Enquiry

Staff

Customer Customer Customer

2.Give information

3.Registration 7.Inform availability 20.generate report 9.Provide product 15.Upadate record

13.give receipt

18.Generate bill

6.Availability

17. New Order

19.Pay bill

5.Check Availability 10.Generate bill

11.Generate receipt

16.Check stock

Supplier

shop

**Structure chart**

A structure chart is a [top-down modular design](http://en.wikipedia.org/wiki/Top-down_design) tool, constructed of squares representing the different modules in the [system](http://en.wikipedia.org/wiki/System), and lines that connect them.

This chart is used in [structured programming](http://en.wikipedia.org/wiki/Structured_programming) to arrange the program modules in a tree structure. Each module is represented by a box, which contains the module's name. The tree structure visualizes the relationships between the modules

represent the connection and or ownership between activities and sub activities as they are used in [organization charts](http://en.wikipedia.org/wiki/Organization_chart)

 Programmers use a structure chart to build a program in a manner similar to how an architect uses a blueprint to build a house. In the design stage, the chart is drawn and used as a way for the client and the various software designers to communicate. During the actual building of the program (implementation), the chart is continually referred to as the master-plan" A structure chart depicts[[2]](http://en.wikipedia.org/wiki/Structure_chart#cite_note-IRT-1)

* the size and complexity of the system, and
* number of readily identifiable functions and modules within each function and
* whether each identifiable function is a manageable entity or should be broken down into smaller components.

A structure chart is also used to [diagram](http://en.wikipedia.org/wiki/Diagram) associated elements that comprise a run stream or thread. It is often developed as a [hierarchicaldiagram](http://en.wikipedia.org/w/index.php?title=Hierarchical_diagram&action=edit&redlink=1), but other representations are allowable. The representation must describe the breakdown of the [configuration system](http://en.wikipedia.org/wiki/Configuration_system) into [subsystems](http://en.wikipedia.org/wiki/Subsystem) and the lowest manageable level. An accurate and complete structure chart is the key to the determination of the configuration items, and a visual representation of the configuration system and the internal interfaces among its CIs. During the configuration control process, the structure chart is used to identify CIs and their associated artifacts that a proposed change may impact

**STRUCTURE CHART**

**SALES MANAGEMENT SYSTEM**

Info about customer

Product details

Info about product

CUSTOMER DETAILS

ENQUIRING ABOUT GYM COURSE

CREATE ORDER

BILL GENERATION

BILL PAID

GET CUSTOMER DETAILS

RETRIEVE CUSTOMER DETAILS

Generate customer details

COURSE DETAILS

CHECKING AVAILIBILITY

Details of order

Info about customer

Generate bill

Info about product

Generate product details

CUSTOMER DETAILS

LIST PRODUCT DETAILS

PLACE ORDER

Generate customer details

Generate product details

Generate Order Details

**Component Diagram**

The different high-level reusable parts of a system are represented in a Component diagram. A component is one such constituent part of a system. In addition to representing the high-level parts, the Component diagram also captures the inter-relationships between these parts.

So, how are component diagrams different from the previous UML diagrams that we have seen? The primary difference is that Component diagrams represent the implementation perspective of a system. Hence, components in a Component diagram reflect grouping of the different design elements of a system, for example, classes of the.

You can model different types of components based on their use and applicability in a system. Components that you can model in a system can be simple executable components or library components that represent system and application libraries used in a system. You also can have file components that represent the source code files of an application or document files that represent, for example, the user interface files such as HTML or JSP files. Finally, you can use components to represent even the database tables of a system as well!

**Elements of a Component Diagram**

A Component diagram consists of the following elements:

|  |  |
| --- | --- |
| **Element and its description** | **Symbol** |
| **Component:** The objects interacting with each other in the system. Depicted by a rectangle with the name of the object in it, preceded by a colon and underlined. | UML10TAB01 |
| **Class/Interface/Object:** Similar to the notations used in class and object diagrams | UML10TAB02 |
| **Relation/Association**: Similar to the relation/association used in class diagrams | UML10TAB03 |

**COMPONENT DIAGRAM**

Login.

Customer-Info.

Supplier-Info.

BOUTIQUE DATABASE

Employee- Details.

Product-Details.

Stock.

SalesOrder-Details.

PurchaseOrder-Details.

CustomerBill

**DEPLOYMENT DIAGRAM**

The deployment diagram provides a different perspective of the application. The deployment diagram captures the configuration of the runtime elements of the application.

This diagram is by far more useful when a system is built and ready to be deployed. But, this does not mean that you should start on your deployment diagram after your system is built. This deployment diagram then evolves and is revised until the system is built. It is always a best practice to have visibility of what your deployment environment is going to be before the system is built so that any deployment-related issues are identified to be resolved and not crop up at the last minute.

The basic deployment diagram element is the node. The node represents the environment in which a component or a set of components execute. This means that a node in a deployment diagram can represent a multitude of things—physical hardware such as a server machine, a system software like an operating system, or even application infrastructure software like a Web server, application server, database server, and so forth. The different nodes in the deployment diagram can be interconnected to represent interdependencies, thus providing a deployment diagram that is easy to comprehend and provides the complete deployment environment of a system.

**Elements of a Deployment Diagram**

A deployment diagram consists of the following elements:

|  |  |
| --- | --- |
| **Element and its description** | **Symbol** |
| **Node:** The element that provides the execution environment for the components of a system. Depicted by a cube with the name of the object in it, preceded by a colon, and underlined. | UML11T01 |
| **Connection:** Similar to the relation/association used in class diagrams to define the interconnection between nodes. | UML11T02 |

**DEPLOYMENT DIAGRAM**

CustomerInfo.

Login.

EmployeeDetails.

Stock.

Database Server

SupplierInfo

ProductDetails.

PurchaseOrder-Details.

SalesOrder-Details.

CustomerBill.

LAN

HUB1

HUB2

Terminal

Printer

Printer

Terminal

**Activity Diagram**

The easiest way to visualize an Activity diagram is to think of a flowchart of a code. The flowchart is used to depict the business logic flow and the events that cause decisions and actions in the code to take place.

Activity diagrams represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state. So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows.

An Activity diagram talks more about these transitions and activities causing the changes in the object states.

**Elements of an Activity diagram**

An Activity diagram consists of the following behavioral elements:

|  |  |
| --- | --- |
| **Element and its description** | **Symbol** |
| **Initial Activity:** This shows the starting point or first activity of the flow. Denoted by a solid circle. This is similar to the notation used for InitialState. | UML07T1 |
| **Activity:** Represented by a rectangle with rounded (almost oval) edges. | UML07T2 |
| **Decisions:** Similar to flowcharts, a logic where a decision is to be made is depicted by a diamond, with the options written on either sides of the arrows emerging from the diamond, within box brackets. | UML07T3 |
| **Signal:** When an activity sends or receives a message, that activity is called a signal. Signals are of two types: Input signal (Message receiving activity) shown by a concave polygon and Output signal (Message sending activity) shown by a convex polygon. | UML07T4 |
| **Concurrent Activities:** Some activities occur simultaneously or in parallel. Such activities are called concurrent activities. For example, listening to the lecturer and looking at the blackboard is a parallel activity. This is represented by a horizontal split (thick dark line) and the two concurrent activities next to each other, and the horizontal line again to show the end of the parallel activity. | UML07T5 |
| **Final Activity:** The end of the Activity diagram is shown by a bull's eye symbol, also called as a final activity. | UML07T6 |

**ACTIVITY DIAGRAM**

UML07T1

Customer Enquiry

Customer Registration

StaffChecks Availability

Order Cancelled

Customer paysadvance

Staff fulfills the order

Customer Pays Bill

Staff Generates Receipt

Staff Update Records

Staff Check AvailableStock

Staff Places New Order

not

yes

Staff Enquiry

Staff Order

Supplier supply product

Owner makes payment

Supplier generatereport

**Part *7***

***SYSTEM  
CODING***

**PROGRAM LIST**

|  |  |  |
| --- | --- | --- |
| **NO** | **FORM NAME** | **PURPOSE** |
| 1 | Frmslash.frm | This form display the title of system name  of organization & copyright of the system |
| 2 | Login.frm | This form is used for security with the help of password |
| 3 | Mdimain.frm | This is the multiple document interface for all the form in the project |
| 4 | CustomerInfo.frm | This form is used to store the customer information |
| 5 | SupplierInfo.frm | This form is used to store the supplier information |
| 6 | Employee.frm | This form is used to store the Employee information |
| 7 | CustomerBill.frm | This form is used to generate the bill for customer |
| 8 | SalesOrder .frm | This form is used to add new sales order and retrieve order by Customer |
| 9 | PurchaseOrder .frm | This form is used to add new Purchaseorder |
| 10 | ProductDetails.frm | This form is used to store the details of product and stock details. |
| 12 | About.frm | This form display information of developer |

**TABLE USED**

|  |  |  |
| --- | --- | --- |
| **NO.** | **TABLE NAME** | **PURPOSE** |
| 1 | Customer\_Info | To maintain Customer information |
| 2 | Customer\_Bill | To maintain Customer bill details |
| 3 | EmployeeDetails | To maintain Employee Details |
| 4 | Product\_Details | To maintain Product Details |
| 5 | Supplier\_Info | To maintain Supplier information |
| 6 | Stock | To maintain Stock Details |
| 7 | Customer registration | To maintain registration details |
| 9 | PurchaseOrder | To maintain PurchaseOrder |
| 10 | fees | To maintain the fees details |

**TABLES FIELD**

**Table Name: Customer\_Info**

**Description**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Key** |
| Date1 | Datetime | ---- |
| Cust\_Id | Integer | Primary Key |
| custName | Varchar | ---- |
| Address | Varchar | ---- |
| telno | Varchar | ---- |
| mobno | Varchar | ---- |
| Office\_add | Varchar | ---- |
| Office\_ telno | Varchar | ---- |

**Table Name**: **Supplier\_Info**

**Description:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Key** |
| Date1 | Datetime | ---- |
| Sup\_id | Integer | Primary Key |
| SupName | Varchar | ---- |
| SupAdd | Number | ---- |
| Company | Varchar | ---- |
| city | Varchar | ---- |
| telno | Varchar | ---- |
| mobno | Varchar | ---- |

**Table Name: ProductDetails**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Key** |
| product\_id | Integer | Primary Key |
| ProductName | Varchar | ---- |
| company | Varchar | ---- |
| stock | Integer | ---- |
| Stock | Integer | ------ |
| SupplierId | Integer | Foreign Key |
| Supplier Name | Varchar | ------ |

**Description:**

**Table Name**:**EmployeeDetails**

**Description:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Key** |
| Emp\_Id | Integer | Primary Key |
| Emp\_Name | Varchar | ---- |
| Emp\_Add | Varchar | ---- |
| Emp\_phone | Varchar | ---- |
| Dept | Varchar | ---- |
| Salary | Integer | ---- |
| Qualif | Varchar | ---- |

**Table Name**: **Customer Bill**

**Description:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Key** |
| BillNO | Integer | Primary Key |
| Date | Number | ---------- |
| Cust Id | Integer | Foreign key |
| Cust name | Varchar | --------- |
| Tel No | Varchar | -------- |
| Disc | Varchar | ---------- |
| Total | Integer | --------- |
| Paid | Integer | ---------- |
| Balance | Integer | ------- |

**Table Name**: **SalesOrder**

**Description:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Key** |
| Date | datetime | --------- |
| OrdNo | Integer | Primary Key |
| CustId | Integer | Foreign key |
| Cust Name | Varchar | ---------- |
| Tel No | Varchar | -------- |
| Address | number | ---------- |
| Pro Id | Integer | Foreign key |
| Pro name | Varchar | ------- |
| Qty | Integer | ----------- |
| Rate | Integer | -------------- |
| Disc | Varchar | ---------- |
| Total | Integer | --------- |
| Paid | Integer | ---------- |
| Balance | Integer | ------- |

**Table Name**: **Purchase Order**

**Description:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Key** |
| purchsOrdNO | Integer | Primary Key |
| Date | Number | ---------- |
| Supplier Id | Integer | Foreign key |
| Suppli name | Varchar | --------- |
| Receiver name | Varchar | ------- |
| Tel No | Varchar | -------- |
| Pro Id | Integer | Foreign key |
| Pro Name | Varchar | ---------- |
| Qty | Integer | ----------- |
| Rate | Integer | -------------- |
| Disc | Varchar | ---------- |
| Total | Integer | --------- |
| Paid | Integer | ---------- |
| Balance | Integer | ------- |

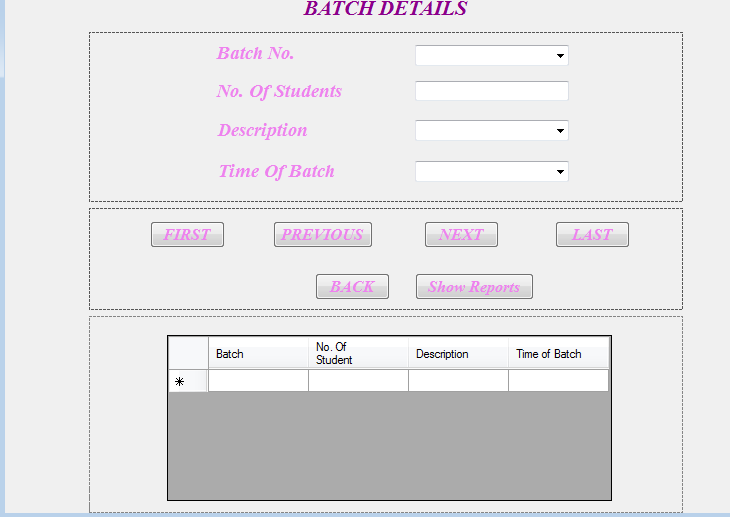
**REPORT LIST**

|  |  |  |
| --- | --- | --- |
| **NO.** | **REPORT NAME** | **PURPOSE** |
| **1** | **CustomerRPT.rpt** | This report is use for to print the customer report by customer id. |
| **2** | **SupplierRPT.rpt** | This report is use for to print the supplier report by supplier id. |
| **3** | **BillRPT.rpt** | *T*his report is use for to generate the bill report for each customer by Bill id. |
| **4** | **PurchaseOrdRPT.rpt** | This report is use for to print the Order Details supplied by Supplier by purchase id. |
| **5** | **SalesOrdRPT.rpt** | This report is use for to print the Order Details sold to customer by sales id. |

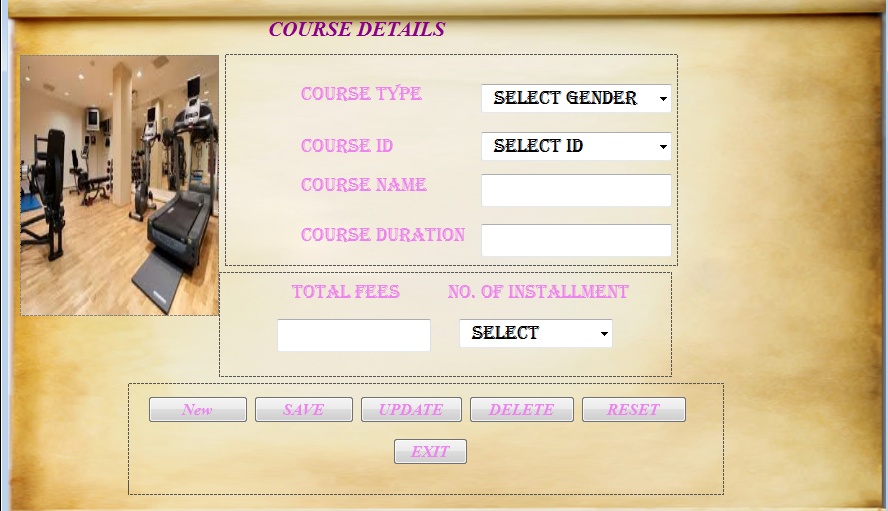
**Part *8***

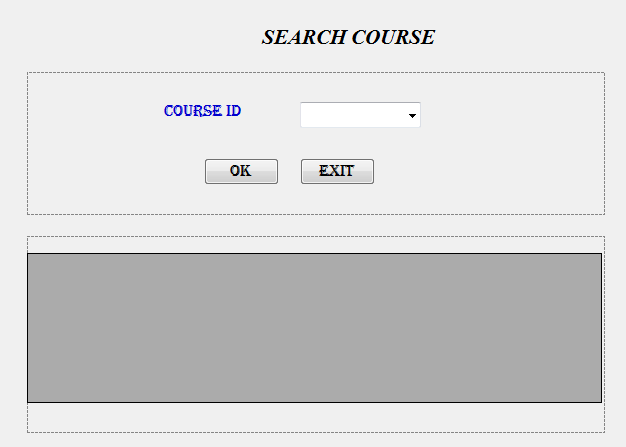
***PROGRAM  
DOCUMENTATION***

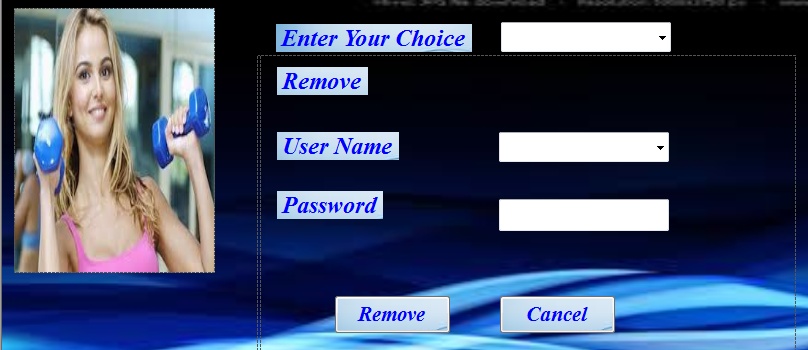
**SYSTEM LEVEL CODING(Snap-shots)**

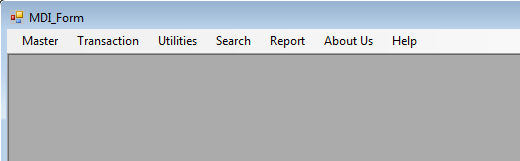
****

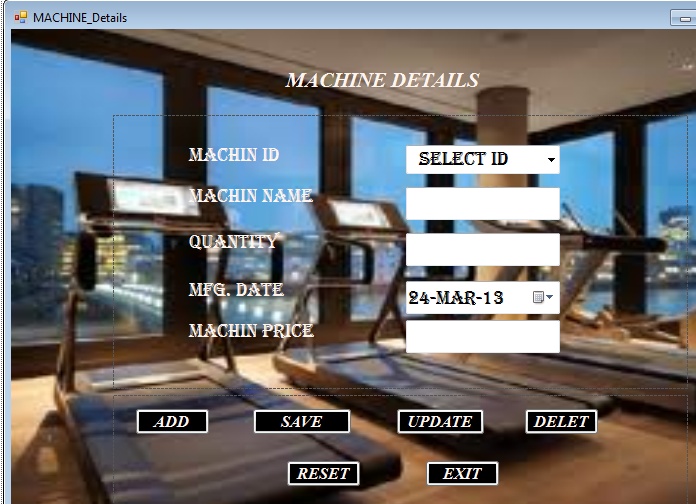
****

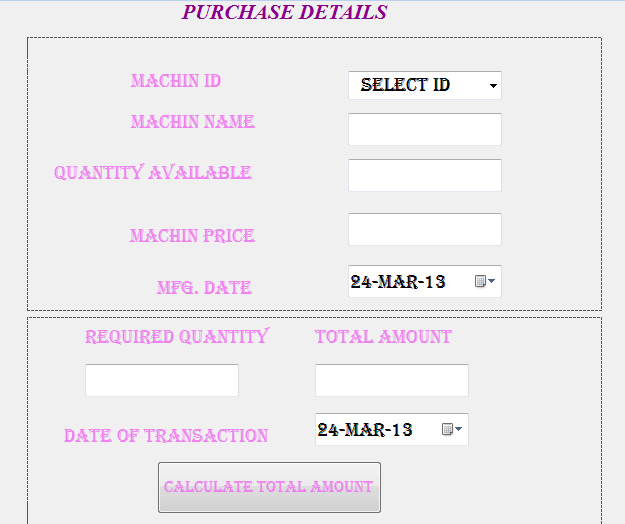
****

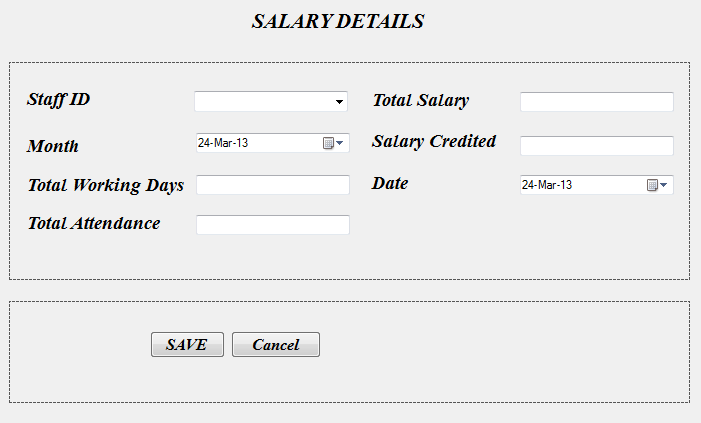
****

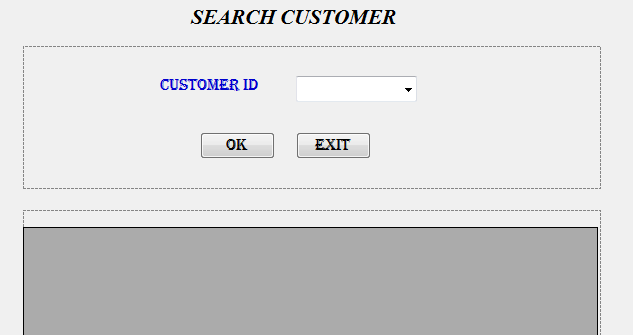
****

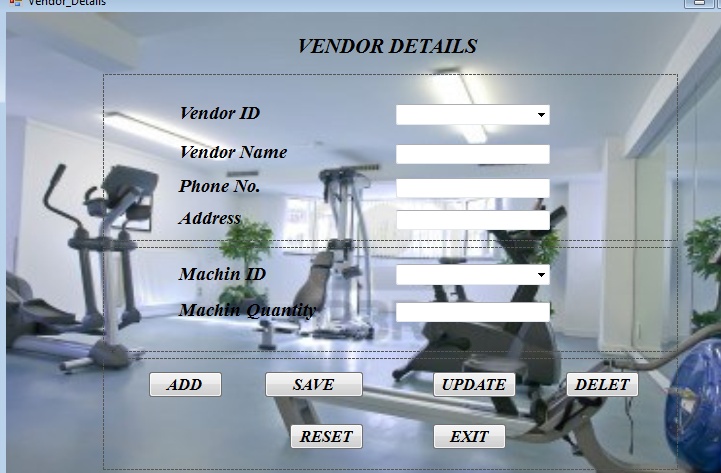
****

****

****

****

****

****

**REPORT FORM:**

****

**CODING**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

namespaceNagi\_construction

**{**

publicpartialclassUpdation\_customer **:** Form

**{**

publicUpdation\_customer**()**

**{**

InitializeComponent**();**

**}**

privatevoidcomboBox2\_SelectedIndexChanged**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidlabvendorname\_Click**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidpanel3courseD\_Paint**(**objectsender**,** PaintEventArgse**)**

**{**

**}**

privatevoidUpdation\_customer\_Load**(**objectsender**,** EventArgse**)**

**{**

**}**

**}**

**}**

**Change password**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

namespaceYours\_Fitness\_Center

**{**

publicpartialclassChange\_Password **:** Form

**{**

publicChange\_Password**()**

**{**

InitializeComponent**();**

**}**

privatevoidChange\_Password\_Load**(**objectsender**,** EventArgse**)**

**{**

**}**

**}**

**}**

**Customer booking**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

namespaceYours\_Fitness\_Center

**{**

publicpartialclassCustomer\_Details **:** Form

**{**

publicCustomer\_Details**()**

**{**

InitializeComponent**();**

**}**

SqlConnectionconn=newSqlConnection**(**"Data Source=SGT-PC;Initial Catalog=FitnessDb;Integrated Security=True;Pooling=False"**);**

SqlCommandquery**;**

SqlDataReaderreader**;**

ConnectionStatestate=newConnectionState**();**

privatevoidCustomer\_Details\_Load**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Cust"**,** conn**);**

reader=query.ExecuteReader**();**

while **(**reader.Read**())**

cmbsid.Items.Add**(**reader**[**0**]**.ToString**());**

reader.Close**();**

conn.Close**();**

**}**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Course"**,** conn**);**

reader=query.ExecuteReader**();**

while **(**reader.Read**())**

cmbcourseEn.Items.Add**(**reader**[**1**]**.ToString**());**

reader.Close**();**

conn.Close**();**

**}**

**}**

privatevoidbutnstudDexit\_Click**(**objectsender**,** EventArgse**)**

**{**

this.Close**();**

**}**

privatevoidbutnstudDreset\_Click**(**objectsender**,** EventArgse**)**

**{**

cmbsid.Text="Select"**;**

txtBstudname.Text=""**;**

txtBstudadd.Text=""**;**

cmbgender.Text="Select"**;**

cmbcourseEn.Text="Select"**;**

cmbBatch.Text="Select"**;**

**}**

privatevoidbutnstudDSave\_Click**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"insert into tbl\_Cust values('"+txtBstudname.Text.Trim**()** +"','"+txtBstudadd.Text.Trim**()** +"','"+cmbgender.Text.Trim**()** +"',"+cmbcourseEn.Text.Trim**()** +",'"+cmbBatch.Text.Trim**()**+"')"**,** conn**);**

reader=query.ExecuteReader**();**

MessageBox.Show**(**"All the records has been submitted successfully into the database..."**);**

reader.Close**();**

conn.Close**();**

**}**

cmbsid.Text="Select"**;**

txtBstudname.Text=""**;**

txtBstudadd.Text=""**;**

cmbgender.Text="Select"**;**

cmbcourseEn.Text="Select"**;**

cmbBatch.Text="Select"**;**

**}**

privatevoidbutnstudDAdd\_Click**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select S\_Id from tbl\_Cust"**,** conn**);**

reader=query.ExecuteReader**();**

while **(**reader.Read**())**

**{**

intvalue=Int32.Parse**(**reader**[**0**]**.ToString**())** +1**;**

cmbsid.Text="S"+String.Format**(**"{0:0000}"**,** value**); ;**

**}**

conn.Close**();**

**}**

**}**

privatevoidcmbsid\_SelectedIndexChanged**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Cust where S\_Id="+cmbsid.Text.Trim**()** +""**,** conn**);**

reader=query.ExecuteReader**();**

while **(**reader.Read**())**

**{**

txtBstudname.Text=reader**[**1**]**.ToString**();**

txtBstudadd.Text=reader**[**2**]**.ToString**();**

cmbgender.Text=reader**[**3**]**.ToString**();**

cmbcourseEn.Text=reader**[**4**]**.ToString**();**

cmbBatch.Text=reader**[**5**]**.ToString**();**

**}**

reader.Close**();**

conn.Close**();**

**}**

**}**

privatevoidbutnstudDupdate\_Click**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"update tbl\_Cust set S\_Name='"+txtBstudname.Text.Trim**()** +"',Address='"+txtBstudadd.Text.Trim**()** +"',Gender='"+cmbgender.Text.Trim**()** +"',C\_Enroll="+cmbcourseEn.Text.Trim**()** +",Batch='"+cmbBatch.Text.Trim**()**+"' Where S\_Id="+cmbsid.Text.Trim**()** +""**,** conn**);**

reader=query.ExecuteReader**();**

MessageBox.Show**(**"Your records has been updated successfully into the database..."**);**

reader.Close**();**

conn.Close**();**

**}**

cmbsid.Text="Select"**;**

txtBstudname.Text=""**;**

txtBstudadd.Text=""**;**

cmbgender.Text="Select"**;**

cmbcourseEn.Text="Select"**;**

cmbBatch.Text="Select"**;**

**}**

privatevoidbutnstudDdelete\_Click**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"delete from tbl\_Cust where S\_Id="+cmbsid.Text.Trim**()** +""**,** conn**);**

reader=query.ExecuteReader**();**

MessageBox.Show**(**"Records Deleted from Database..."**);**

conn.Close**();**

**}**

cmbsid.Text="Select"**;**

txtBstudname.Text=""**;**

txtBstudadd.Text=""**;**

cmbgender.Text="Select"**;**

cmbcourseEn.Text="Select"**;**

cmbBatch.Text="Select"**;**

**}**

**}**

**}**

**Employee details**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

namespaceYours\_Fitness\_Center

**{**

publicpartialclassStaff\_Details **:** Form

**{**

publicStaff\_Details**()**

**{**

InitializeComponent**();**

**}**

SqlConnectionconn=newSqlConnection**(**"Data Source=SGT-PC;Initial Catalog=FitnessDb;Integrated Security=True;Pooling=False"**);**

SqlCommandquery**;**

SqlDataReaderreader**;**

ConnectionStatestate=newConnectionState**();**

privatevoidbutnstaffDexit\_Click**(**objectsender**,** EventArgse**)**

**{**

this.Close**();**

**}**

privatevoidbutnstaffDreset\_Click**(**objectsender**,** EventArgse**)**

**{**

cmbsid.Text="Select Id"**;**

textBstaffname.Text=""**;**

comboBstaffgender.Text="Select Gender"**;**

textBstaffadd.Text=""**;**

comboBQualification.Text="Select"**;**

comboBExperience.Text="Select"**;**

txtsal.Text=""**;**

**}**

privatevoidbutnstaffDAdd\_Click**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select S\_Id from tbl\_Staff"**,** conn**);**

reader=query.ExecuteReader**();**

while **(**reader.Read**())**

**{**

intvalue=Int32.Parse**(**reader**[**0**]**.ToString**())** +1**;**

cmbsid.Text="E"+String.Format**(**"{0:0000}"**,** value**); ;**

**}**

conn.Close**();**

**}**

**}**

privatevoidbutnstaffDSave\_Click**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"insert into tbl\_Staff values('"+textBstaffname.Text.Trim**()** +"','"+comboBstaffgender.Text.Trim**()** +"','"+textBstaffadd.Text.Trim**()** +"','"+comboBQualification.Text.Trim**()** +"','"+comboBExperience.Text.Trim**()** +"',"+txtsal.Text.Trim**()**+")"**,** conn**);**

reader=query.ExecuteReader**();**

MessageBox.Show**(**"All the records has been submitted successfully into the database..."**);**

reader.Close**();**

conn.Close**();**

**}**

cmbsid.Text="Select Id"**;**

textBstaffname.Text=""**;**

comboBstaffgender.Text="Select Gender"**;**

textBstaffadd.Text=""**;**

comboBQualification.Text="Select"**;**

comboBExperience.Text="Select"**;**

txtsal.Text=""**;**

**}**

privatevoidStaff\_Details\_Load**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Staff"**,** conn**);**

reader=query.ExecuteReader**();**

while **(**reader.Read**())**

cmbsid.Items.Add**(**reader**[**0**]**.ToString**());**

reader.Close**();**

conn.Close**();**

**}**

**}**

privatevoidcmbsid\_SelectedIndexChanged**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Staff where S\_Id="+cmbsid.Text.Trim**()** +""**,** conn**);**

reader=query.ExecuteReader**();**

while **(**reader.Read**())**

**{**

textBstaffname.Text=reader**[**1**]**.ToString**();**

comboBstaffgender.Text=reader**[**2**]**.ToString**();**

textBstaffadd.Text=reader**[**3**]**.ToString**();**

comboBQualification.Text=reader**[**4**]**.ToString**();**

comboBExperience.Text=reader**[**5**]**.ToString**();**

txtsal.Text=reader**[**6**]**.ToString**();**

**}**

reader.Close**();**

conn.Close**();**

**}**

**}**

privatevoidbutnstaffDupdate\_Click**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"update tbl\_Staff set S\_Name='"+textBstaffname.Text.Trim**()** +"',Gender='"+comboBstaffgender.Text.Trim**()** +"',Address='"+textBstaffadd.Text.Trim**()** +"',Qualification='"+comboBQualification.Text.Trim**()** +"',Experience='"+comboBExperience.Text.Trim**()** +"',Salary="+txtsal.Text.Trim**()**+" Where S\_Id="+cmbsid.Text.Trim**()** +""**,** conn**);**

reader=query.ExecuteReader**();**

MessageBox.Show**(**"Your records has been updated successfully into the database..."**);**

reader.Close**();**

conn.Close**();**

**}**

cmbsid.Text="Select Id"**;**

textBstaffname.Text=""**;**

comboBstaffgender.Text="Select Gender"**;**

textBstaffadd.Text=""**;**

comboBQualification.Text="Select"**;**

comboBExperience.Text="Select"**;**

txtsal.Text=""**;**

**}**

privatevoidbutnstaffDdelete\_Click**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"delete from tbl\_Course where C\_Id="+cmbsid.Text.Trim**()** +""**,** conn**);**

reader=query.ExecuteReader**();**

MessageBox.Show**(**"Records Deleted from Database..."**);**

conn.Close**();**

**}**

cmbsid.Text="Select Id"**;**

textBstaffname.Text=""**;**

comboBstaffgender.Text="Select Gender"**;**

textBstaffadd.Text=""**;**

comboBQualification.Text="Select"**;**

comboBExperience.Text="Select"**;**

txtsal.Text=""**;**

**}**

**}**

**}**

**Employee payment**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

namespaceYours\_Fitness\_Center

**{**

publicpartialclassSalary\_Details **:** Form

**{**

publicSalary\_Details**()**

**{**

InitializeComponent**();**

**}**

SqlConnectionconn=newSqlConnection**(**"Data Source=SGT-PC;Initial Catalog=FitnessDb;Integrated Security=True;Pooling=False"**);**

SqlCommandquery**;**

SqlDataAdapterreader**;**

SqlDataReaderreader1**;**

ConnectionStatestate=newConnectionState**();**

DataSetds=newDataSet**();**

privatevoidtextBox7\_TextChanged**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidtextBox4\_TextChanged**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidlabel4\_Click**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidlabel3\_Click**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidSalary\_Details\_Load**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Staff"**,** conn**);**

reader1=query.ExecuteReader**();**

while **(**reader1.Read**())**

**{**

cmbsalaryDstafid.Items.Add**(**reader1**[**0**]**.ToString**());**

**}**

reader1.Close**();**

conn.Close**();**

**}**

**}**

privatevoiddateTimePicker1\_ValueChanged**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidbutnSalaryDshowRep\_Click**(**objectsender**,** EventArgse**)**

**{**

this.Close**();**

**}**

privatevoiddateTimePicker1\_Leave**(**objectsender**,** EventArgse**)**

**{**

txtBsalaryDtotworkDays.Text=DateTime.DaysInMonth**(**dateTimePicker1.Value.Year**,** dateTimePicker1.Value.Month**)**.ToString**();**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Staff where S\_Id="+cmbsalaryDstafid.Text.Trim**()** +""**,** conn**);**

reader1=query.ExecuteReader**();**

while **(**reader1.Read**())**

txtBsalaryDsalary.Text=reader1**[**6**]**.ToString**();**

**}**

DateTimePickerob=newDateTimePicker**();**

StringMonth=dateTimePicker1.Value.Month.ToString**();**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Close**();**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Att where S\_Id="+cmbsalaryDstafid.Text.Trim**()** +""**,** conn**);**

reader1=query.ExecuteReader**();**

while **(**reader1.Read**())**

ob.Text=reader1**[**1**]**.ToString**();**

StringMont=ob.Value.Month.ToString**();**

reader1.Close**();**

if **(**Month.Equals**(**Mont**))**

**{**

query=newSqlCommand**(**"Select \* from tbl\_Att where S\_Id="+cmbsalaryDstafid.Text.Trim**()** +""**,** conn**);**

reader1=query.ExecuteReader**();**

while **(**reader1.Read**())**

txtBsalaryDtotAtten.Text=reader1**[**3**]**.ToString**();**

**}**

else

**{**

MessageBox.Show**(**"Please Update Attendance First...."**);**

this.Close**();**

**}**

conn.Close**();**

doubleDays=0.0**,** Sal**,** Res**;**

Days=Double.Parse**(**txtBsalaryDtotAtten.Text**);**

Sal=Double.Parse**(**txtBsalaryDsalary.Text**);**

Res= **(**Days\*Sal**)** /30**;**

txtBsalaryDdatOfPaym.Text=Res.ToString**();**

**}**

**}**

privatevoidbutnsalaryDSave\_Click**(**objectsender**,** EventArgse**)**

**{**

DateTimePickerob=newDateTimePicker**();**

StringMonth=dateTimePicker1.Value.Month.ToString**();**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Close**();**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Sal where S\_Id="+cmbsalaryDstafid.Text.Trim**()** +""**,** conn**);**

reader1=query.ExecuteReader**();**

while **(**reader1.Read**())**

ob.Text=reader1**[**1**]**.ToString**();**

StringMont=ob.Value.Month.ToString**();**

conn.Close**();**

if **(**Month.Equals**(**Mont**))**

**{**

MessageBox.Show**(**"Your Salary for this month already credited..."**);**

this.Close**();**

**}**

else

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Close**();**

conn.Open**();**

query=newSqlCommand**(**"insert into tbl\_Sal values("+cmbsalaryDstafid.Text.Trim**()** +",'"+txtBsalaryDsalary.Text.Trim**()** +"','"+txtBsalaryDdatOfPaym.Text.Trim**()** +"',"+txtBsalaryDtotAtten.Text.Trim**()** +",'"+dateTimePicker1.Text.Trim**()**+"')"**,** conn**);**

reader1=query.ExecuteReader**();**

MessageBox.Show**(**"All the records has been submitted successfully into the database..."**);**

reader1.Close**();**

conn.Close**();**

**}**

**}**

**}** conn.Close**();**

**}**

**}**

**}**

**Purchase details**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

namespaceYours\_Fitness\_Center

**{**

publicpartialclassPurchase\_Details **:** Form

**{**

publicPurchase\_Details**()**

**{**

InitializeComponent**();**

**}**

SqlConnectionconn=newSqlConnection**(**"Data Source=SGT-PC;Initial Catalog=FitnessDb;Integrated Security=True;Pooling=False"**);**

SqlCommandquery**;**

SqlDataReaderreader**;**

ConnectionStatestate=newConnectionState**();**

privatevoidpanel3courseD\_Paint**(**objectsender**,** PaintEventArgse**)**

**{**

**}**

privatevoidpanl1courceD\_Paint**(**objectsender**,** PaintEventArgse**)**

**{**

**}**

privatevoidlabcourseDuration\_Click**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidlabcourseno\_Click**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidpictureBox2\_Click**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidpictureBox3\_Click**(**objectsender**,** EventArgse**)**

**{**

**}**

privatevoidPurchase\_Details\_Load**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Prod"**,** conn**);**

reader=query.ExecuteReader**();**

while **(**reader.Read**())**

cmbMachinid.Items.Add**(**reader**[**0**]**.ToString**());**

reader.Close**();**

conn.Close**();**

**}**

**}**

privatevoidbutnpurDexit\_Click**(**objectsender**,** EventArgse**)**

**{**

this.Close**();**

**}**

privatevoidcmbMachinid\_SelectedIndexChanged**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Prod where M\_Id="+cmbMachinid.Text.Trim**()** +""**,** conn**);**

reader=query.ExecuteReader**();**

while **(**reader.Read**())**

**{**

txtmname.Text=reader**[**1**]**.ToString**();**

txtqty.Text=reader**[**2**]**.ToString**();**

dateTimePicker1.Text=reader**[**3**]**.ToString**();**

txtprice.Text=reader**[**4**]**.ToString**();**

**}**

reader.Close**();**

conn.Close**();**

**}**

**}**

privatevoidbutnpurchaseRep\_Click**(**objectsender**,** EventArgse**)**

**{**

intqty=Int32.Parse**(**txtreq.Text.Trim**());**

doubleprice=Double.Parse**(**txtprice.Text.Trim**());**

doubleres=qty\*price**;**

txttamount.Text=res.ToString**();**

**}**

privatevoidbutnpurDreset\_Click**(**objectsender**,** EventArgse**)**

**{**

cmbMachinid.Text="Select Id"**;**

txtmname.Text=""**;**

txtqty.Text=""**;**

dateTimePicker1.Text=System.DateTime.Now.ToShortDateString**();**

txtprice.Text=""**;**

txtreq.Text=""**;**

txttamount.Text=""**;**

dateTimePicker2.Text=System.DateTime.Now.ToShortDateString**();**

**}**

privatevoidbutnpurDSave\_Click**(**objectsender**,** EventArgse**)**

**{**

if **(**state==ConnectionState.Open**)**

conn.Close**();**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"insert into tbl\_Pur values("+cmbMachinid.Text.Trim**()** +",'"+txtmname.Text.Trim**()** +"',"+txtqty.Text.Trim**()** +","+txtprice.Text.Trim**()** +",'"+dateTimePicker1.Text.Trim**()** +"',"+txtreq.Text.Trim**()** +","+txttamount.Text.Trim**()** +",'"+dateTimePicker2.Text.Trim**()** +"')"**,** conn**);**

reader=query.ExecuteReader**();**

MessageBox.Show**(**"All the records has been submitted successfully into the database..."**);**

reader.Close**();**

conn.Close**();**

**}**

**}**

**}**

**}**

**Testing**

**Part *9***

***TESTING***

There are many approaches to software testing. Reviews, walkthroughs, or inspections are considered as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing can be (and unfortunately in practice often is) omitted. Dynamic testing takes place when the program itself is used for the first time (which is generally considered the beginning of the testing stage). Dynamic testing may begin before the program is 100% complete in order to test particular sections of code (modules or discrete functions). Typical techniques for this are either using stubs/drivers or execution from a debugger environment. For example, spreadsheet programs are, by their very nature, tested to a large extent interactively ("on the fly"), with results displayed immediately after each calculation or text manipulation.

**Testing Methods**

**The box approach**

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

**White box testing** is when the tester has access to the internal data structures and algorithms including the code that implement these.

**Types of white box testing**

The following types of white box testing exist:

* API testing (application programming interface) - testing of the application using public and private APIs
* Code coverage - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
* Fault injection methods - improving the coverage of a test by introducing faults to test code paths
* Mutation testing methods
* Static testing - White box testing includes all static testing

**Black box testing**

* Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing

**Testing levels**

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test.

**Unit testing**

**Unit testing** refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These type of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Unit testing is also called *component testing*.

**Integration testing**

**Integration testing** is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be localised more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

**System testing**

System testing tests a completely integrated system to verify that it meets its requirements.

**System integration testing**

System integration testing verifies that a system is integrated to any external or third-party systems defined in the system requirements.

**Regression testing** focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncoversoftware regressions, or old bugs that have come back. Such regressions occur whenever software functionality that was previously working correctly stops working as intended. Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the risk of the added features. They can either be complete, for changes added late in the release or deemed to be risky, to very shallow, consisting of positive tests on each feature, if the changes are early in the release or deemed to be of low risk.

**Acceptance testing**

Acceptance testing can mean one of two things:

1. A smoke test is used as an acceptance test prior to introducing a new build to the main testing process, i.e. beforeintegration or regression.
2. Acceptance testing performed by the customer, often in their lab environment on their own hardware, is known as user acceptance testing (UAT). Acceptance testing may be performed as part of the hand-off process between any two phases of development.

**Part 10**

**REPORTS**

REPORT LIST

|  |  |  |
| --- | --- | --- |
| **NO.** | **REPORT NAME** | **PURPOSE** |
| **2** | **EmployeeRPT.rpt** | This report is use for to print the supplier report by supplier id. |
| **3** | **BillRPT.rpt** | *T*his report is use for to generate the bill report for each customer by Bill id. |

**Part *11***

***FUTURE***

***ENHANCEMENT***

**FUTURE ENHANCEMENT**

Being a computer system, the system has lots of scope. It not only carried out work faster but also efficiently .A lots of manual work like searching ,personal alias is reduced to a large extent. As this is computerized, system the manager is completely depends on computer for accessing details about students. Here incase there is power failure or some hardware problem which cannot be fixed easily, the manager cannot continue this work. He gets struck in the middle of the day and this dealing need to be postponed.

Also the software need to be maintained properly from time to time, i.e if it requires to be updated or modified etc. then the manager need to take care of it.

**Part *12***

***CONCLUSION***

**CONCLUSION**

The developed system GYM MANAGEMENT is successfully implemented. The system is successfully tested and also it provides nearly all requirements of user. The System provide quick convent timely information and output with better accuracy with using less time, less manual efforts, less cost, but up to this time. There is security of keeping data, and also it is flexible enough to accommodate further change of future requirements. It also provides some kind of help to user about application through help file.

* System helps to store all the data about the customer order transaction in computer and there is no need to do paper work.
* Data is going to be preserved carefully for longer period hence proper backup is required otherwise there is chance of losing entries or data.

**Part *13***

***BIBLIOGRAPHY***

**BIBLIOGRAPHY**

**Books referred**

|  |  |  |
| --- | --- | --- |
| Sr.No. | Type | Description |
| **1.** | **Book name :**  **Author :**  **Publication :** | **C# 2008**  **Murach**  **Tata Ma-Graw Hill Edition** |
| **2.** | **Book name :**  **Author :**  **Publication :** | **Programming in C# 6.0**  **Julia case Bradley,Anita C Millspough**  **Tata Mc-Graw Hill Edition** |
| **3.** | **Book name :**  **Author :**  **Publication :** | **Database programming with C# in 21 days**  **Curtis Smith &Michale Amundsen**  **MacMillan Computer Publish ,USA** |

**Thank You**

**THE -END**