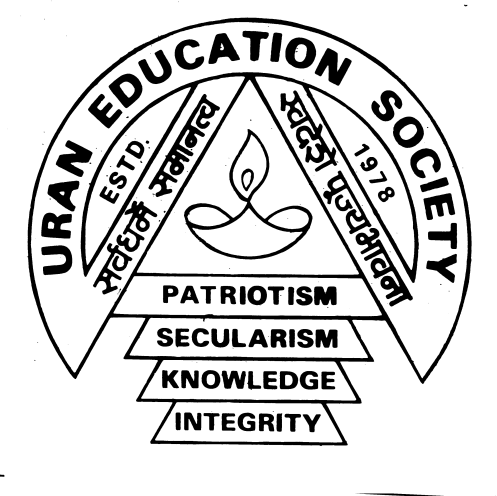
****

**URAN EDUCATION SOCIETY’S**

**COLLEGE OF MANAGEMENT & TECHNOLOGY**

**(Bori, Uran)**

**A**

**Project Report**

**on**

**TOURS & TRAVELS**

**Submitted To:**

**University of Mumbai**

**By**

**Manali Dinesh Haldankar & Yuganti Manohar Devrukhkar**

**T.Y.BSc (IT)**

**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 Tours & Travels Management was successfully done by Manali Dinesh Haldankar & Yuganti Manohar Devrukhkar in the computer laboratory of Uran Education Society’s College of Management & Technology, Bori Uran during the academic year 2012-2013.

Exam seat no: \_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_

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 “Tours & Travels 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-**

**(MANALI.D.HALDANKAR**

**&**

**YUGANTI.M.DEVRUKHKAR)**

**PREFACE**

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

The Software provides help in effective management of deals with the distributors and customers and the employee of the organization 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. | **14** |
| **4** | **Proposed System.** | **15** |
| **5** | **Software-Hardware Specifications.** | **18** |
| **6** | **System Design:** | **21** |
|  | Entity Relationship Diagram | **22** |
|  | Event Table | **23** |
|  | Use Case Diagram | **25** |
|  | Class Diagram | **28** |
|  | Context Diagram | **30** |
|  | Data flow Diagram | **32** |
|  | System Flowchart | **38** |
|  | Sequence Diagram. | **39** |
|  | Collaboration Diagram | **41** |
|  | Structure Diagram | **43** |
|  | Component Diagram | **45** |
|  | Deployment Diagram | **47** |
|  | Activity Diagram | **49** |
| **7** | **System Coding:** | **53** |
|  | Program List | **54** |
|  | Tables Used | **55** |
|  | Report List | **56** |
| **8** | **Program Documentation.** | **58** |
|  | System Level Coding(snap shots) | **59** |
|  | Coding | **65** |
| **9** | **Testing:** | **85** |
| **10** | **Reports** | **89** |
| **11** | **Future Enhancement** | **91** |
| **12** | **Conclusion.** | **93** |
| **13** | **Bibliography.** | **95** |

**Part *1***

***INTRODUCTION***

**INTRODUCTION**

Every person wills to travels the areas he/she has never seen or visited before.

This needs information about the places and famous spots to be visited.

Besides this there is also need of vehicles for travelling, hotels for staying, good restaurants for food etc.

These all small & big things are managed by the traveling agencies on behalf of their customers.

At present all the data are handled manually.

Our aim is to computerized the whole system, which will facilitate easy search and updation activities.

**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  Vehicle 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**

***Preliminary Investigation:-***

*Motto :-*

To give maximum level of comfort and luxury to passengers

# *Foundation :-*

Mr. Karan Thakur founded the esteem Yatra travels in 2004. It is at Panvel. This is one of the most popular Chemicals distributors in Panvel.

# *Availability :-*

It provides all types of chemicals and accessories in reasonable rate.

**ORGANIZATIONAL CHART**

**TOURS &TRAVELS MANAGEMENT**

CUSTOMER

SUPPLIER

EMPLOYEE

**PRESENT SYSTEM**

Customer requirement is first considered then the respective order is placed before the customer. The order is with respect to the reasonable market rate. The order contains following fields:

Customer id: - Customer Number

Customer first name: - Customer first name

Customer last name: - Customer last name

Customer Address: - Customer Address

Customer Phone no.:- Customer Phone no.

Date of Purchase: - Date of Purchase

If the customer agrees with the order provided. He gave the respective order for the corresponding items. The clerk will prepare the order note that contains the following fields:

Order id:-Order Number

Customer id: - Customer Number

Customer first name: - Customer first name

Customer last name: - Customer last name

Customer Phone no.:- Customer Phone no.

Date of Purchase: - Date of Purchase

Chemicals

Accessories

After worker is done the fulfillment of the order, the Owner create bill for order. That contain following field:

Bill Id:-Bill Number

Customer id: - Customer Number

Customer name: - Customer name

Customer Phone no.:- Customer Phone no.

Date of Bill: - Date of Bill

Chemicals

Accessories

All work is done manually and records are safely stored in registers.

**LIMITATION OF PRESENT SYSTEM**

* In Present system each an every record is maintains in registers, so there will be lot of redundancy in maintaining 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 occur.
* 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**

**The Benefits of Computerized System:**

Since maintaining the records of order, bills and payment details are very difficult. So I have taken the charge of designing the system. Maintaining the information regarding the necessary vehicles will be easy in the computerized system. The shop has decided to keep the department manually. Hence I am designing the system for them.

In Yatra travels customer is involved everywhere & Calculation is done

related to customer.

**OBJECTIVES**

1. Data management
2. Data retrieval at will
3. Data modification
4. Save time thereby increasing performance
5. Reducing work process time

## 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, vehicle 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 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, vehicles, 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(min).
* Base Memory 640kb.
* Cache Memory 256kb.
* CPU Clock 150 MHz
* Display Type EGA/VGA.
* Hard Disk 100MB(min.).

**Part *6***

***SYSTEM  
DESIGN***

ENTITY RELATIONSHIP DIAGRAM

**SHOP**

**VEHICLES**

**CUSTOMER**

**STAFF**

**Event table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Event** | Trigger | **Source** | **Activity** | **Response** | **Destination** |
| 1) | **Customer enquires for vehicle.** | Get information about vehicle | **Customer** | **Enquiring about vehicle** | **Availability status** | **Customer** |
| **2)** | **Registration** | **Enquiry** | **Customer** | **Registration customer** | **Registration confirmed** | **Customer** |
| **3)** | **Checking availability** | **Get Vehicle details** | **Staff** | **Checking availability of vehicle** | **Availability status** | **Staff** |
| **4)** | **Order vehicle** | **Availability status** | **Staff** | **Ordering vehicle** | **Order received** | **Staff** |
| **5)** | **Paying advance** | **Registration** | **Customer** | **Paying advance** | **Advance paid** | **Staff** |
| **6)** | **Order fulfillment** | **Vehicle ordered** | **Staff** | **Providing vehicle/service** | **Confirmation** | **Customer** |
| **7)** | **Generate Customer Bill** | **Providing vehicle/service** | **Staff** | **Generating Bill** | **Bill Generated** | **Staff** |
| **8)** | **Pay bill** | **Receiving bill** | **Customer** | **Paying bill** | **Bill received** | **Staff** |
| **9)** | **Generate Receipt** | **Bill paid** | **Staff** | **Generating**  **Receipt** | **Receipt Generated** | **Customer** |
| **10)** | **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.



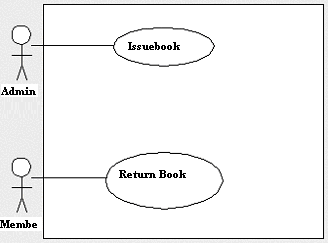
an actor in a use case diagram

**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

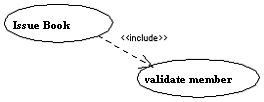
**System boundary:** A system boundary defines the scope of what a system will be.

:

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

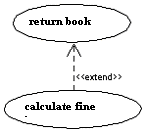
### Relationships in Use Cases

**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

**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 and interfaces

### 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:

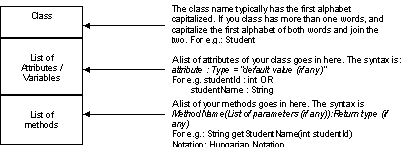
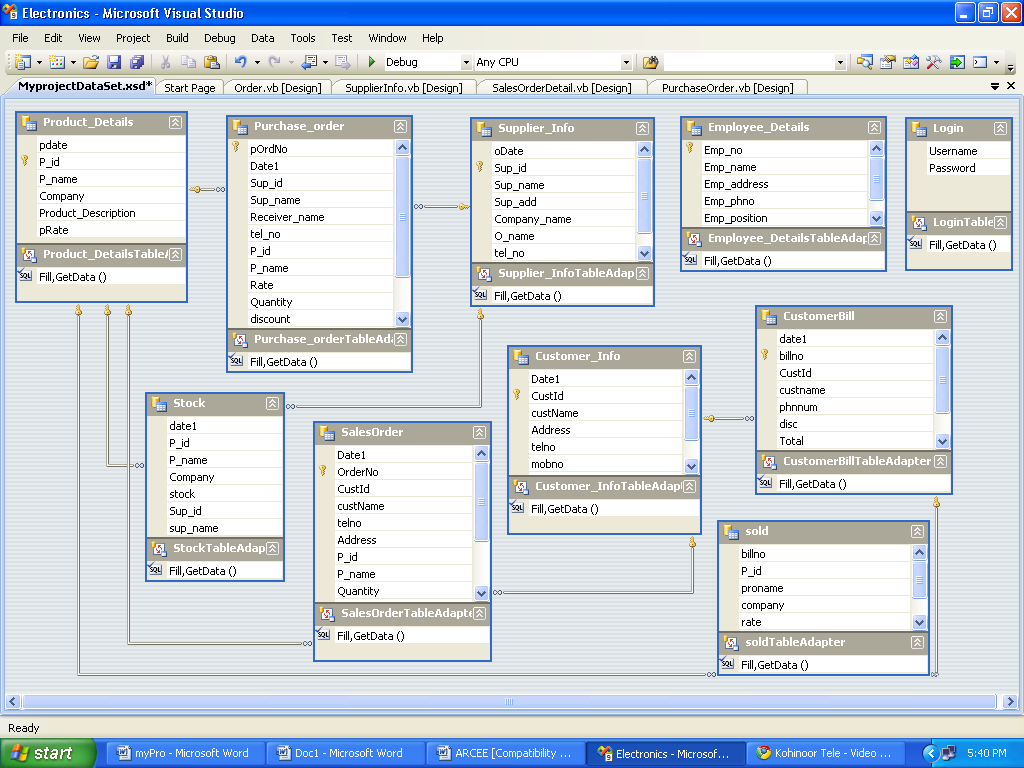


Figure 4.1.1—the structure of a class

**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:**

**Two wheelers**

SALES MANAGEMENT SYSTEM

GET RECIEPT AVAILIBILITY

ORDER

STATUS OF

STOCK

ENQUIRES ITEMS

# CUSTOMER

DETAILS

ORDER GRANTED SUPPLY

PAYS BILL

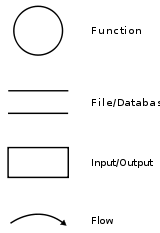
# Data flow diagram

# 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 of [data processing](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**

**1.LIST OF CUSTOMER RECORDS**

**1**

Customer Information

Create New Information

**Management**

**Customer**

Update

Information

Item inquiry

Availability

**Customer**

**Stock**

**Product**

**Customer**

**Remaining**

**Order Details**

Create new order

**Customer**

Order

Confirmation

1. **Add New Vehicle Entry**

**3**

Make New

Product Entry

New **Vehicle** Entry

Product Details

User

Vehicle confirmation

## Process No.: 3

**Process Name:** New vehicle

**Inflow:** Vehicle Detail

**Outflow:** Selected Vehicle

**Process Description (Structure English):**

**If** the vehicle is new **then**

Get new vehicle details

Make new vehicle entry

Send the vehicle details to vehicle

Store the selected vehicle in to the item table

**End if**

1. **Make New Purchase Entry**

**4**

Update

Purchase

Payment

New Purchase Entry

User

Product Details

Purchase confirmation

Customer Details

## Process No.: 4

**Process Name:** New purchase entry

**Inflow:** Payment details

**Outflow:** Selected payment

**Process Description (Structure English):**

Get new purchase details

Update purchase entry

**If** purchase is updated **then**

Owner gives the payment details to the customer

Customer gives the vehicle details and his details to the Owner

**End if**

1. **Make New Sales Entry**

**4**

Update

Stock

Payment

New Sales Entry

User

Product Details

Sales confirmation

Customer Details

## Process No.: 5

**Process Name:** New sales entry

**Inflow:** Payment details

**Outflow:** Selected payment

**Process Description (Structure English):**

Get new sales details

Update sales entry

**If** sales is updated **then**

Owner gives the payment details to the Customer

Customer pays the bill to the Owner

**End if**

1. **Update Supplier Entry**

**5**

Update

Supplier

Entry

Update **Supplier**

Supplier

Supplier

Change Confirmation

## Process no:.6

**Process Name:** Update supplier

**Inflow:** Supplier Detail

**Outflow:** Change confirmation

**Process Description (Structure English):**

**If** supplier exists **then**

Update the supplier details

Send the supplier details to supplier

Store the selected supplier in to the supplier table

**End if**

1. **Update Customer Entry**

**6**

Update

Customer

Entry

Update **Customer**

Customer

User

hange Confirmat change confirmation

## Process no: 7

**Process Name:** Update Customer

**Inflow:** Customer Detail

**Outflow:** Change Confirmation

**Process Description (Structure English):**

**If** customer exists **then**

Update the customer details

Send the customer details to customer

**End if**

**8.Generate bill report**

User

**8**

The Sales

Report Generated

Sales\_master

Produce Sales Report

Sales Report

## Process no:.8

## Process name: Generate bill report

**Inflow:** Vehicle sales report

**Outflow:** Sales report

**Process Description (Structure English):**

User sends vehicle sales report

**If** sales report is valid **then**

The sales report is generated

It is stored in the sales master

The user gets the sales report

**End if**

**SYSTEM**– **FLOWCHART**

**YATRA TRAVELS**

**MANAGEMENT**

**SYSTEM**

**CUSTOMER**

**DATABASE**

**ORDER**

**DATABASE**

**VEHICLE**

**DATABASE**

**BILL**

**DATABASE**

**STOCK**

**DATABASE**

**MAINTAIN CUSTOMER DETAILS**

**CUSTOMER ORDER DETAIL**

**MAINTAIN VEHICLE DETAILS**

**BILL CREATION PROGRAM**

**MAINTAIN PRODUCT DETAILS**

**ORDER**

**REPORT**

**BILL**

**REPORT**

**STOCK**

**REPORT**

**CUSTOMER**

**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. ordervehicle

5.Check Availability

6.Availabile

7.Inform Availability

8.pay advance

9.Provide vehicle

10.Generate Bill

11.pay bill

12.generate reciept

13.give receipt

14.reciept received

15.Update Record

16.Check Stock

17.Order New Vehicle

18.Generate Bill

19.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 vehicle

1.Enquiry

Staff

Customer Customer Customer

2.Give information

3.Registration 7.Inform availability 9.Provide vehicle 15.Upadate record

13.give receipt

18.Generate bill

6.Availability

17. NewOrder

5.Check Availability 10.Generate bill

11.Generate receipt

16.Check stock

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 subactivities 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 [hierarchical diagram](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 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 vehicle

CUSTOMER DETAILS

ENQUIRING ABOUT VEHICLE

CREATE ORDER

BILL GENERATION

BILL PAID

GET CUSTOMER DETAILS

RETRIEVE CUSTOMER DETAILS

Generate customer details

VEHICLE DETAILS

CHECKING AVAILIBILITY

Details of order

Info about customer

Generate bill

Info about vehicle

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,

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.vb

Customer-Info.vb

YATRA TRAVEL SHOWROOM DATABASE

Employee- Details.vb

vehicle-Details.vb

Stock.vb

SalesOrder-Details.vb

PurchaseOrder-Details.vb

CustomerBill

.vb

#### 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.vb

Login.vb

EmployeeDetails.vb

Stock.vb

Database Server

ProductDetails.vb

PurchaseOrder-Details.vb

SalesOrder-Details.vb

CustomerBill.vb

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**

Customer Enquiry

Customer Registration

Staff Checks Availability

Order Cancelled

Customer pays advance

Staff fulfills the order

Customer Pays Bill

Staff Generates Receipt

Staff Update Records

Staff Check Available Stock

Staff Places New Order

not

yes

Staff Enquiry

Staff Order

Supplier supply product

Owner makes payment

Supplier generatereport

**Part *7***

***SYSTEM  
CODING***

|  |  |  |
| --- | --- | --- |
| **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 |
| 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 | VehicleDetails.frm | This form is used to store the details of vehicle and stock details. |
| 12 | About.frm | This form display information of developer |

PROGRAM LIST

**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 | Vehicle\_Details | To maintain Vehicle Details |
| 6 | Stock | To maintain Stock Details |
| 7 | SalesOrder | To maintain Sales Order |
| 9 | PurchaseOrder | To maintain PurchaseOrder |
| 10 | Sold | To maintain the sold 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: VehicleDetails**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Key** |
| vehicle\_id | Integer | Primary Key |
| VehicleName | 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**

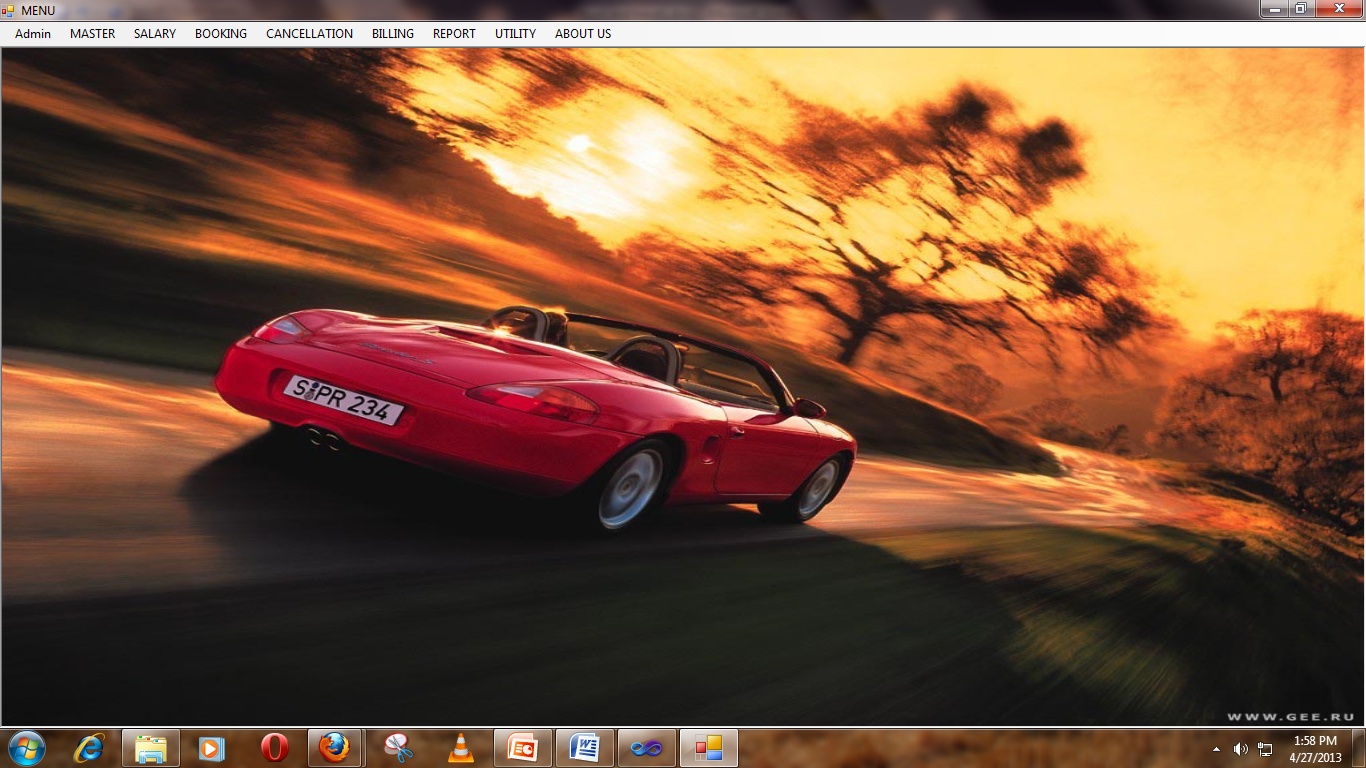
**Description:**

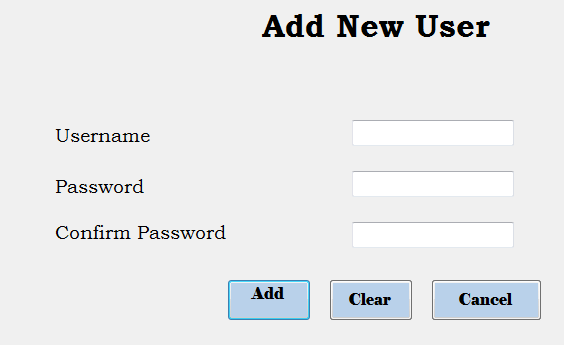
|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Key** |
| Date | datetime | --------- |
| OrdNo | Integer | Primary Key |
| CustId | Integer | Foreign key |
| Cust Name | Varchar | ---------- |
| Tel No | Varchar | -------- |
| Address | number | ---------- |
| Pro Id | Integer | Foreign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

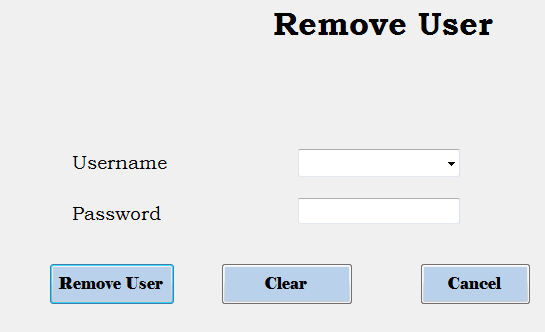
**Part *8***

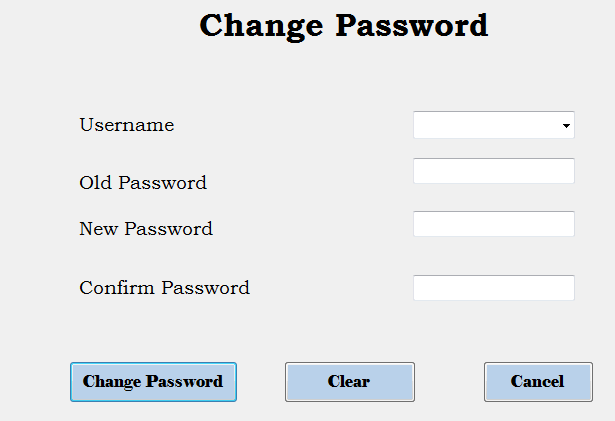
***PROGRAM  
DOCUMENTATION***

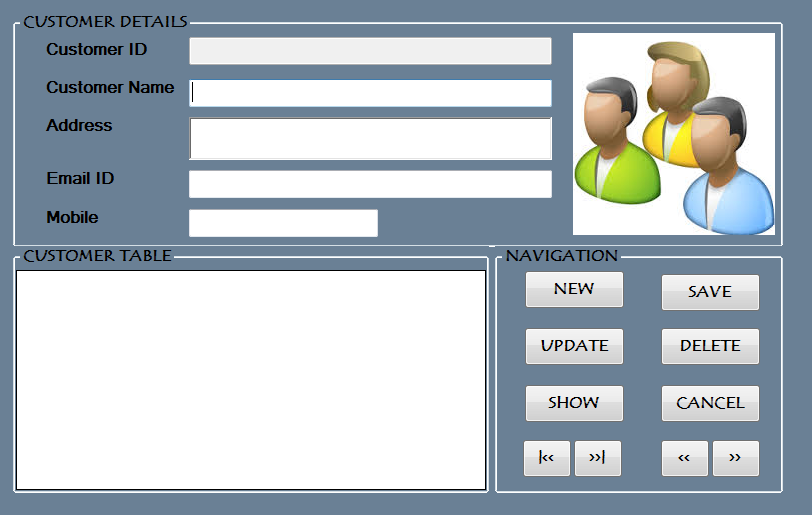
**SYSTEM LEVEL CODING(SNAP-SHOTS)**

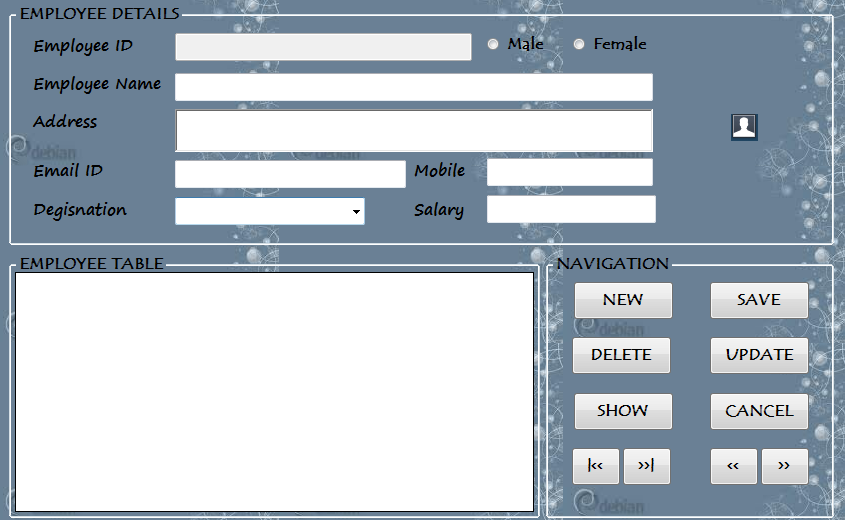
****

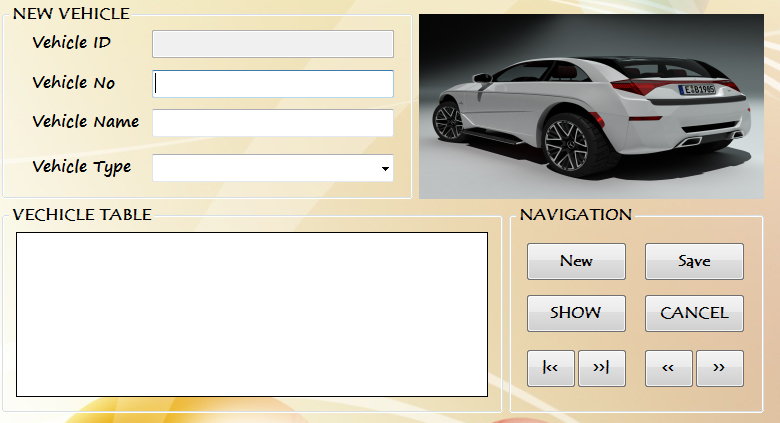
****

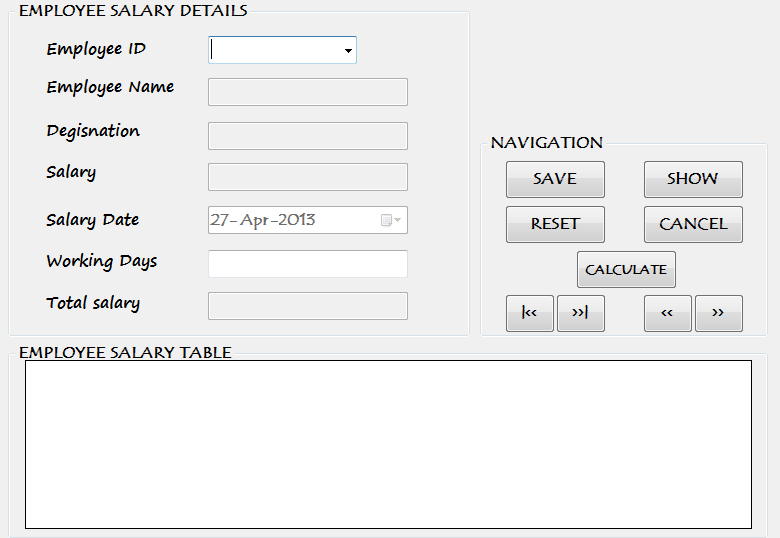
****

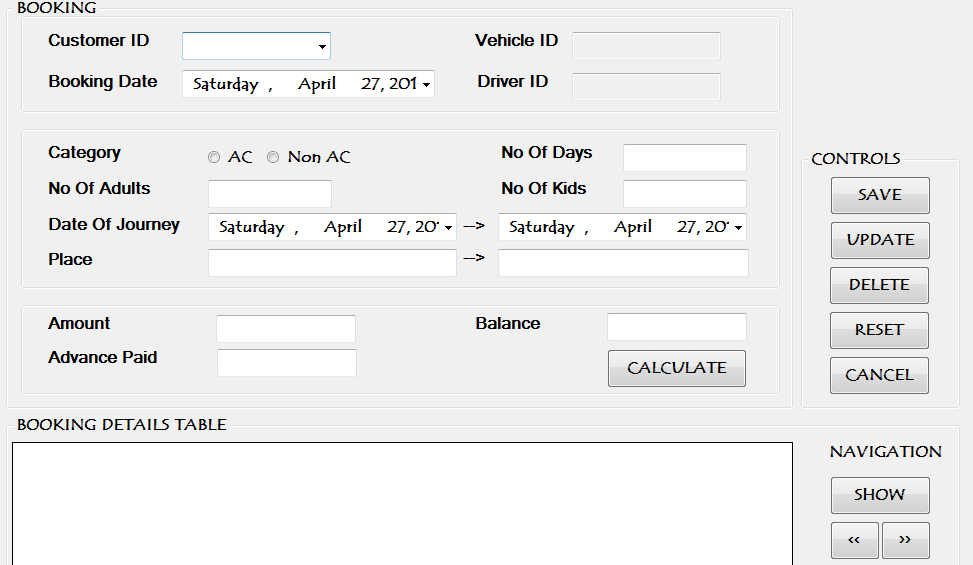
****

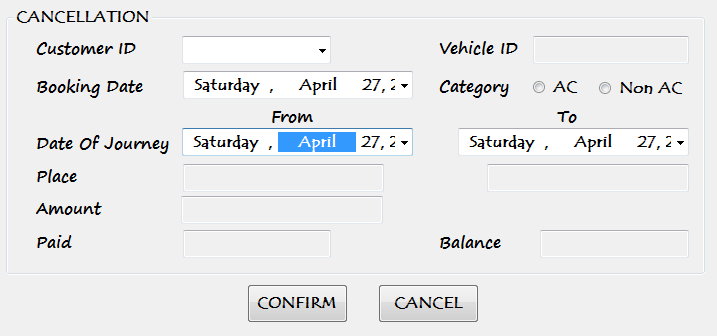
******

******

******

******

******

******

**CODING**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Linq**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

namespacetours

**{**

publicpartialclassChangePassword **:** Form

**{**

publicChangePassword**()**

**{**

InitializeComponent**();**

**}**

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

SqlCommandquery**;**

SqlDataReaderreader**;**

ConnectionStatestate=newConnectionState**();**

privatevoidbutton1\_Click**(**objectsender**,** EventArgse**)**

**{**

this.Close**();**

**}**

privatevoidbutton7\_Click**(**objectsender**,** EventArgse**)**

**{**

cmbuser.Text="Select"**;**

txtoldPass.Text=""**;**

txtPass.Text=""**;**

txtcPass.Text=""**;**

**}**

privatevoidbutton6\_Click**(**objectsender**,** EventArgse**)**

**{**

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

conn.Close**();**

else

**{**

conn.Open**();**

StringPass=txtoldPass.Text.Trim**();**

StringUser=cmbuser.Text.Trim**();**

StringNPass=txtPass.Text.Trim**();**

StringCPass=txtcPass.Text.Trim**();**

StringU=null**,** P=null**;**

query=newSqlCommand**(**"select \* from tbl\_Login where User\_Name='"+cmbuser.Text.Trim**()** +"'"**,** conn**);**

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

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

**{**

U=reader**[**0**]**.ToString**();**

P=reader**[**1**]**.ToString**();**

**}**

reader.Close**();**

if **(**U.Equals**(**User**)** &&P.Equals**(**Pass**))**

**{**

if **(**NPass.Equals**(**CPass**))**

**{**

query=newSqlCommand**(**"update tbl\_Login set Password='"+txtcPass.Text.Trim**()** +"' where User\_Name='"+cmbuser.Text.Trim**()** +"'"**,** conn**);**

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

MessageBox.Show**(**"Your Passward has been Changed...."**);**

cmbuser.Text="Select"**;**

txtoldPass.Text=""**;**

txtcPass.Text=""**;**

txtPass.Text=""**;**

cmbuser.Focus**();**

**}**

else

**{**

MessageBox.Show**(**"Your New Password & COnfirm Password are not matching..."**);**

txtcPass.Text=""**;**

txtPass.Text=""**;**

txtPass.Focus**();**

**}**

**}**

else

**{**

MessageBox.Show**(**"Previous Password Mismatch..."**);**

txtoldPass.Text=""**;**

txtcPass.Text=""**;**

txtPass.Text=""**;**

txtoldPass.Focus**();**

**}**

reader.Close**();**

conn.Close**();**

**}**

**}**

privatevoidChangePassword\_Load**(**objectsender**,** EventArgse**)**

**{**

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

conn.Close**();**

else

**{**

conn.Open**();**

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

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

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

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

reader.Close**();**

conn.Close**();**

**}**

**}**

**}**

**}**

**CUSTOMER DETAILS**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Linq**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

namespacetours

**{**

publicpartialclasscust\_details **:** Form

**{**

publiccust\_details**()**

**{**

InitializeComponent**();**

**}**

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

SqlCommandquery**;**

SqlDataReaderreader**;**

DataSetds=newDataSet**();**

SqlDataAdapterda**;**

publicvoidFill**()**

**{**

conn.Close**();**

txtcid.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Cust.C\_Id"**);**

txtcname.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Cust.Name"**);**

txtcadr.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Cust.Address"**);**

txtcmail.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Cust.Email"**);**

txtcmob.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Cust.Mob"**);**

**}**

privatevoidbutton2\_Click**(**objectsender**,** EventArgse**)**

**{**

this.Close**();**

**}**

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

**{**

conn.Open**();**

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

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

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

**{**

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

txtcid.Text="C-"+String.Format**(**"{0:0000}"**,** value**); ;**

**}**

reader.Close**();**

conn.Close**();**

**}**

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

**{**

conn.Open**();**

query=newSqlCommand**(**"insert into tbl\_Cust values('"+txtcname.Text.Trim**()** +"','"+txtcadr.Text.Trim**()** +"','"+txtcmail.Text.Trim**()** +"','"+txtcmob.Text.Trim**()** +"')"**,** conn**);**

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

MessageBox.Show**(**"Records saved Successfully.."**);**

reader.Close**();**

conn.Close**();**

txtcid.Text=""**;**

txtcname.Text=""**;**

txtcadr.Text=""**;**

txtcmail.Text=""**;**

txtcmob.Text=""**;**

**}**

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

**{**

conn.Open**();**

query=newSqlCommand**(**"update tbl\_Cust set Name='"+txtcname.Text.Trim**()** +"',Address='"+txtcadr.Text.Trim**()** +"',Email='"+txtcmail.Text.Trim**()** +"',Mob='"+txtcmob.Text.Trim**()** +"' where C\_Id="+txtcid.Text.Trim**()** +" "**,** conn**);**

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

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

reader.Close**();**

conn.Close**();**

txtcid.Text=""**;**

txtcname.Text=""**;**

txtcadr.Text=""**;**

txtcmail.Text=""**;**

txtcmob.Text=""**;**

**}**

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

**{**

conn.Open**();**

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

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

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

reader.Close**();**

conn.Close**();**

**}**

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

**{**

conn.Open**();**

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

da=newSqlDataAdapter**(**query**);**

da.Fill**(**ds**,** "tbl\_Cust"**);**

dgcust.DataSource=ds**;**

dgcust.DataMember="tbl\_Cust"**;**

Fill**();**

conn.Close**();**

**}**

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

**{**

intn=this.BindingContext**[**ds**,** "tbl\_Cust"**]**.Count-1**;**

this.BindingContext**[**ds**,** "tbl\_Cust"**]**.Position=n**;**

**}**

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

**{**

this.BindingContext**[**ds**,** "tbl\_Cust"**]**.Position=0**;**

**}**

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

**{**

this.BindingContext**[**ds**,** "tbl\_Cust"**]**.Position--**;**

**}**

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

**{**

this.BindingContext**[**ds**,** "tbl\_Cust"**]**.Position++**;**

**}**

**}**

**}**

**EMPLOYEE DETAILS**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Linq**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

namespacetours

**{**

publicpartialclassemp\_details **:** Form

**{**

publicemp\_details**()**

**{**

InitializeComponent**();**

**}**

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

SqlCommandquery**;**

SqlDataReaderreader**;**

DataSetds=newDataSet**();**

SqlDataAdapterda**;**

publicvoidFill**()**

**{**

conn.Close**();**

txtcid.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Emp.E\_Id"**);**

txtcname.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Emp.Name"**);**

txtcadr.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Emp.Address"**);**

txtcmail.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Emp.Email"**);**

txtcmob.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Emp.Mob"**);**

cmbdgst.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Emp.Designation"**);**

txtsal.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Emp.Salary"**);**

**}**

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

**{**

this.Close**();**

**}**

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

**{**

conn.Open**();**

query=newSqlCommand**(**"Select E\_Id from tbl\_Emp"**,** conn**);**

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

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

**{**

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

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

**}**

reader.Close**();**

conn.Close**();**

**}**

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

**{**

StringGender=null**;**

if **(**rbtnfemale.Checked==true**)**

Gender="Female"**;**

else

Gender="Male"**;**

conn.Open**();**

query=newSqlCommand**(**"insert into tbl\_Emp values('"+txtcname.Text.Trim**()** +"','"+txtcadr.Text.Trim**()** +"','"+txtcmail.Text.Trim**()** +"','"+txtcmob.Text.Trim**()** +"','"+Gender+"','"+cmbdgst.Text.Trim**()**+"','"+txtsal.Text.Trim**()**+"')"**,** conn**);**

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

MessageBox.Show**(**"Records saved Successfully.."**);**

reader.Close**();**

conn.Close**();**

txtcid.Text=""**;**

txtcname.Text=""**;**

txtcadr.Text=""**;**

txtcmail.Text=""**;**

txtcmob.Text=""**;**

rbtnfemale.Checked=false**;**

rbtnmale.Checked=false**;**

cmbdgst.Text=""**;**

txtsal.Text=""**;**

**}**

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

**{**

conn.Open**();**

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

da=newSqlDataAdapter**(**query**);**

da.Fill**(**ds**,** "tbl\_Emp"**);**

dgemp.DataSource=ds**;**

dgemp.DataMember="tbl\_Emp"**;**

Fill**();**

conn.Close**();**

**}**

privatevoiddgemp\_CellContentClick**(**objectsender**,** DataGridViewCellEventArgse**)**

**{**

StringGender=null**;**

conn.Open**();**

query=newSqlCommand**(**"select \* from tbl\_Emp where E\_Id="+txtcid.Text.Trim**()** +""**,** conn**);**

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

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

**{**

Gender=reader**[**5**]**.ToString**();**

**}**

reader.Close**();**

conn.Close**();**

if **(**Gender.Equals**(**"Female"**))**

rbtnfemale.Checked=true**;**

else

rbtnmale.Checked=true**;**

**}**

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

**{**

conn.Open**();**

query=newSqlCommand**(**"delete from tbl\_Emp where E\_Id="+txtcid.Text.Trim**()** +""**,** conn**);**

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

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

reader.Close**();**

conn.Close**();**

txtcid.Text=""**;**

txtcname.Text=""**;**

txtcadr.Text=""**;**

txtcmail.Text=""**;**

txtcmob.Text=""**;**

rbtnfemale.Checked=false**;**

rbtnmale.Checked=false**;**

cmbdgst.Text=""**;**

txtsal.Text=""**;**

**}**

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

**{**

StringGender=null**;**

if **(**rbtnfemale.Checked==true**)**

Gender="Female"**;**

else

Gender="Male"**;**

conn.Open**();**

query=newSqlCommand**(**"update tbl\_Emp set Name='"+txtcname.Text.Trim**()** +"',Address='"+txtcadr.Text.Trim**()** +"',Email='"+txtcmail.Text.Trim**()** +"',Mob='"+txtcmob.Text.Trim**()** +"',Gender='"+Gender+"',Designation='"+cmbdgst.Text.Trim**()** +"',Salary='"+txtsal.Text.Trim**()** +"' where E\_Id="+txtcid.Text.Trim**()** +" "**,** conn**);**

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

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

reader.Close**();**

conn.Close**();**

**}**

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

**{**

intn=this.BindingContext**[**ds**,** "tbl\_Emp"**]**.Count-1**;**

this.BindingContext**[**ds**,** "tbl\_Emp"**]**.Position=n**;**

**}**

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

**{**

this.BindingContext**[**ds**,** "tbl\_Emp"**]**.Position=0**;**

**}**

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

**{**

this.BindingContext**[**ds**,** "tbl\_Emp"**]**.Position--**;**

**}**

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

**{**

this.BindingContext**[**ds**,** "tbl\_Emp"**]**.Position++**;**

**}**

**}**

**}**

**EMPLOYEE SALARY**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Linq**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

namespacetours

**{**

publicpartialclassemp\_sal **:** Form

**{**

publicemp\_sal**()**

**{**

InitializeComponent**();**

**}**

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

SqlCommandquery**;**

SqlDataReaderreader**;**

DataSetds=newDataSet**();**

SqlDataAdapterda**;**

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

**{**

this.Close**();**

**}**

publicvoidFill**()**

**{**

conn.Close**();**

cmbeid.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Salary.Id"**);**

txtename.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Salary.Name"**);**

cmbdgst.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Salary.Designation"**);**

txtesal.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Salary.Salary"**);**

dateTimePicker1.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Salary.Date"**);**

txtnod.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Salary.WDay"**);**

txtetot.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Salary.ASalary"**);**

**}**

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

**{**

conn.Open**();**

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

da=newSqlDataAdapter**(**query**);**

da.Fill**(**ds**,** "tbl\_Salary"**);**

dataGridView1.DataSource=ds**;**

dataGridView1.DataMember="tbl\_Salary"**;**

Fill**();**

conn.Close**();**

**}**

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

**{**

cmbeid.Text="Select"**;**

txtename.Text=""**;**

cmbdgst.Text=""**;**

txtesal.Text=""**;**

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

txtnod.Text=""**;**

txtetot.Text=""**;**

**}**

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

**{**

DateTimePickerob=newDateTimePicker**();**

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

ob.Text= **(**ob.Value.AddMonths**(**-2**))**.ToShortDateString**(); ;**

conn.Open**();**

query=newSqlCommand**(**"Select \* from tbl\_Salary where Id="+cmbeid.Text.Trim**()** +""**,** conn**);**

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

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

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

conn.Close**();**

if **(**ob.Value.Month==dateTimePicker1.Value.Month**)**

MessageBox.Show**(**"Salary Already Credited for this month..."**);**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"insert into tbl\_Salary values("+cmbeid.Text.Trim**()** +",'"+txtename.Text.Trim**()** +"','"+cmbdgst.Text.Trim**()** +"','"+txtesal.Text.Trim**()** +"','"+dateTimePicker1.Text.Trim**()** +"',"+txtnod.Text.Trim**()** +",'"+txtetot.Text.Trim**()** +"')"**,** conn**);**

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

MessageBox.Show**(**"Salary Credit for this month..."**);**

reader.Close**();**

conn.Close**();**

**}**

cmbeid.Text="Select"**;**

txtename.Text=""**;**

cmbdgst.Text=""**;**

txtesal.Text=""**;**

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

txtnod.Text=""**;**

txtetot.Text=""**;**

**}**

privatevoidemp\_sal\_Load**(**objectsender**,** EventArgse**)**

**{**

conn.Open**();**

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

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

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

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

reader.Close**();**

conn.Close**();**

**}**

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

**{**

conn.Open**();**

query=newSqlCommand**(**"select \* from tbl\_Emp where E\_Id="+cmbeid.Text.Trim**()** +""**,** conn**);**

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

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

**{**

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

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

txtesal.Text=reader**[**7**]**.ToString**();**

**}**

reader.Close**();**

conn.Close**();**

**}**

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

**{**

intn=DateTime.DaysInMonth**(**dateTimePicker1.Value.Year**,** dateTimePicker1.Value.Month-1**);**

inta=Int32.Parse**(**txtnod.Text**);**

doubleSal=Double.Parse**(**txtesal.Text**);**

if **(**a>n**)**

**{**

MessageBox.Show**(**"Month Days exceeded from the actual days.."**);**

txtnod.Text=""**;**

txtnod.Focus**();**

**}**

else

**{**

txtetot.Text= **((**Sal\*a**)** /n**)**.ToString**();**

**}**

**}**

**}**

**}**

**LOGIN FORM**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Linq**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

usingSystem.Data.SqlClient**;**

namespacetours

**{**

publicpartialclasslogin **:** Form

**{**

publiclogin**()**

**{**

InitializeComponent**();**

**}**

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

SqlCommandq**;**

SqlDataReaderreader**;**

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

**{**

Application.Exit**();**

**}**

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

**{**

StringUser=txtUser.Text.Trim**();**

StringPassword=txtPass.Text.Trim**();**

StringU=null**;**

StringP=null**;**

conn.Open**();**

q=newSqlCommand**(**"Select \* from tbl\_Login where User\_Name='"+txtUser.Text.Trim**()** +"'"**,** conn**);**

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

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

**{**

U=reader**[**0**]**.ToString**();**

P=reader**[**1**]**.ToString**();**

**}**

reader.Close**();**

conn.Close**();**

if **(**User.Equals**(**""**)** ||Password.Equals**(**""**))**

MessageBox.Show**(**"FIll the Fields...."**);**

else

**{**

if **(**User.Equals**(**U**)** &&Password.Equals**(**P**))**

**{**

MessageBox.Show**(**"Login Successfully.."**);**

this.Visible=false**;**

mdia=newmdi**();**

a.Show**();**

**}**

else

**{**

MessageBox.Show**(**"User Name Or Password Mismatch..."**);**

txtUser.Text=""**;**

txtPass.Text=""**;**

txtUser.Focus**();**

**}**

**}**

**}**

privatevoidlogin\_Load**(**objectsender**,** EventArgse**)**

**{**

**}**

**}**

**}**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Linq**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

namespacetours

**{**

publicpartialclassmdi **:** Form

**{**

publicmdi**()**

**{**

InitializeComponent**();**

**}**

privatevoidtoolTip1\_Popup**(**objectsender**,** PopupEventArgse**)**

**{**

**}**

privatevoidmenu\_Load**(**objectsender**,** EventArgse**)**

**{**

**}**

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

**{**

cust\_detailsob=newcust\_details**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

drivr\_detailsob=newdrivr\_details**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

privatevoidvEHICLEToolStripMenuItem\_Click\_1**(**objectsender**,** EventArgse**)**

**{**

**}**

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

**{**

System.Diagnostics.Process.Start**(**"calc"**);**

**}**

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

**{**

System.Diagnostics.Process.Start**(**"C:\\WINDOWS\\system32\\timedate.cpl"**);**

**}**

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

**{**

new\_vecob=newnew\_vec**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

vec\_detailsob=newvec\_details**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

vec\_mentob=newvec\_ment**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

bookingob=newbooking**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

cancellationob=newcancellation**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

privatevoidePLYOEEDETAILSMenuItem1\_Click**(**objectsender**,** EventArgse**)**

**{**

emp\_detailsob=newemp\_details**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

privatevoidbILLINGToolStripMenuItem1\_Click**(**objectsender**,** EventArgse**)**

**{**

billob=newbill**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

emp\_salob=newemp\_sal**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

driv\_salob=newdriv\_sal**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

emp\_atndob=newemp\_atnd**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

drivr\_atndob=newdrivr\_atnd**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

**}**

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

**{**

AddUserob=newAddUser**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

RemoveUserob=newRemoveUser**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

ChangePasswordob=newChangePassword**();**

ob.MdiParent=this**;**

ob.Show**();**

**}**

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

**{**

Application.Exit**();**

**}**

**}**

**}**

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.ComponentModel**;**

usingSystem.Data**;**

usingSystem.Drawing**;**

usingSystem.Linq**;**

usingSystem.Text**;**

usingSystem.Windows.Forms**;**

usingSystem.Data.SqlClient**;**

namespacetours

**{**

publicpartialclassnew\_vec **:** Form

**{**

publicnew\_vec**()**

**{**

InitializeComponent**();**

**}**

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

SqlCommandquery**;**

SqlDataReaderreader**;**

DataSetds=newDataSet**();**

SqlDataAdapterda**;**

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

**{**

this.Close**();**

**}**

publicvoidFill**()**

**{**

txtvid.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Veh.V\_Id"**);**

txtvno.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Veh.V\_No"**);**

txtvname.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Veh.V\_Name"**);**

cmbvtyp.DataBindings.Add**(**"Text"**,** ds**,** "tbl\_Veh.Type"**);**

**}**

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

**{**

conn.Open**();**

query=newSqlCommand**(**"Select V\_Id from tbl\_Veh"**,** conn**);**

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

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

**{**

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

txtvid.Text="V-"+String.Format**(**"{0:0000}"**,** value**); ;**

**}**

reader.Close**();**

conn.Close**();**

**}**

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

**{**

if **(**txtvname.Text.Equals**(**""**)** ||txtvno.Text.Equals**(**""**)** ||cmbvtyp.Text.Equals**(**""**))**

MessageBox.Show**(**"Please Fill all the fields..."**);**

else

**{**

conn.Open**();**

query=newSqlCommand**(**"insert into tbl\_Veh values('"+txtvno.Text.Trim**()** +"','"+txtvname.Text.Trim**()** +"','"+cmbvtyp.Text.Trim**()** +"','Free')"**,** conn**);**

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

reader.Close**();**

conn.Close**();**

MessageBox.Show**(**"Vehical added Successfully.."**);**

**}**

**}**

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

**{**

conn.Open**();**

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

da=newSqlDataAdapter**(**query**);**

da.Fill**(**ds**,**"tbl\_Veh"**);**

dgvtb.DataSource=ds**;**

dgvtb.DataMember="tbl\_Veh"**;**

Fill**();**

conn.Close**();**

**}**

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

**{**

intn=this.BindingContext**[**ds**,** "tbl\_Veh"**]**.Count-1**;**

this.BindingContext**[**ds**,** "tbl\_Veh"**]**.Position=n**;**

**}**

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

**{**

this.BindingContext**[**ds**,** "tbl\_Veh"**]**.Position=0**;**

**}**

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

**{**

this.BindingContext**[**ds**,** "tbl\_Veh"**]**.Position++**;**

**}**

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

**{**

this.BindingContext**[**ds**,** "tbl\_Veh"**]**.Position--**;**

**}**

**}**

**}**

**Part 9**

***TESTING***

**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 localized 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**

**Regression testing** focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software 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. before integration 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***

***DEVELOPMENT***

**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 “TOURS &TRAVELS” 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# .NET**  **Julia case Bradley,Anita C Millspough**  **Tata Mc-Graw Hill Edition** |
| **3.** | **Book name :**  **Author :**  **Publication :** | **Database programming with c# .NETin 21 days**  **Curtis Smith &Michale Amundsen**  **MacMillan Computer Publish ,USA** |

**Thank You**

**THE -END**