

## **1.INTRODUCTION**

The project entitled “Gas Agency Management System” is done to make the manual process easier by making it a computerized system for billing and maintaining stock. The Gas Agencies get the order request through phone calls or by personal from their customers and deliver the gas cylinders to their address based on their demand and previous delivery date.

This process is made computerized and the customer’s name, address and stock details are stored in a database. Since a customer order for gas can be accepted only after completing a certain period from the previous delivery. This can be calculated and billed easily through this. There are two types of delivery like domestic purpose use delivery and commercial purpose use delivery. The bill rate and capacity differs for both. This can be easily maintained and charged accordingly. The stock of gas and all its details are processed swiftly.

### **1.1 PROBLEM DEFINITION**

A gas agency is a business that distributes gas cylinders to households and commercial customers. The agency manages the distribution of gas cylinders, maintains customer accounts, tracks inventory, and handles delivery and billing processes. The existing manual system used by many gas agencies is inefficient and prone to errors. This results in delays in delivery, inaccurate billing, and poor customer service.

### **1.2 SCOPE OF THE PROJECT**

- Improve the efficiency of gas cylinder distribution and delivery processes
- Increase accuracy in billing and customer account management
- Provide real-time visibility into inventory levels and customer orders
- Reduce manual errors and minimize operational costs
- Enhance customer satisfaction and loyalty by providing a seamless user experience.

## **1.3 MODULES IN THE PROJECT**

- Admin Login
- Adding Customers
- View Customers
- Booking
- Delivery
- View Delivery
- Sock
- Report

**Admin Login :** The module provides an interface for the administrator to enter their username and password to gain access to the system. The login page should be secure and prevent unauthorized access to the system..

**Adding Customers :** This module allows the gas agency to add new customers to the system by providing their personal details such as name, address, contact number, and gas connection details. The system can also generate a unique customer ID for each customer

**View Customer :** This module allows the gas agency to view the details of all customers in the system. The information includes the customer's personal details, gas connection details, and other relevant information.

**Booking :** This module enables the gas agency to manage the booking of gas cylinders by customers. The system can accept bookings, assign delivery dates, and track the booking status. This module allows authorized personnel to create new gas cylinder booking requests for customers. The module typically includes fields for the customer's details, the delivery address, the quantity of gas cylinders required, and the requested delivery date. The module may also include features such as booking confirmation, booking modification

**Delivery :** This module enables the gas agency to schedule and track the delivery of gas cylinders to customers. It includes features such as assigning delivery personnel, tracking delivery status, and updating delivery information in real-time. The module may also include features such as delivery optimization, delivery routing, and delivery scheduling.

**View Delivery :** This module allows the gas agency to view the delivery details of all customers. The system can display the delivery status, delivery date and other relevant information.

**Stock :** This module allows the gas agency to manage the inventory of gas cylinders, and other related items. The system can track the number of items in stock. This module manages the inventory of gas cylinders. It includes features such as tracking the stock levels of cylinders, generating alerts when stock levels are low, and managing the refill process. The module may also include features such as stock forecasting, stock replenishment, and stock optimization.

**Report :** This module generates reports on various aspects of the business, such as sales, inventory levels, delivery performance, and customer payment history. The module typically includes options to filter and customize reports based on specific criteria. The module may also include features such as data visualization and report sharing.

These are the modules designed.

## **2. SYSTEM STUDY**

### **2.1 STUDY OF EXISTING SYSTEM**

The existing system is a manual system. In the present Gas Agency system, all the activities are done manually. All data entry is performed by writing data into the book, paper documents. The bills are prepared manually, so there is a chance for occurring errors and the calculations are not so accurate. Various information's such as customer details, stock details and customer requirements are handled manually. When there is need for retrieving details searching is unavoidable this is a difficult task searching the records manually .This is also too much time consuming when we want to retrieve details according to some specific conditions.

### **2.2 FEASIBILITY STUDY**

Feasibility study is test of system proposal according to its workability, impact on organization, ability to meet the needs ,effective use of resources. During the study, the problem definition is crystallized and aspects of the problem to be included in this system are determined. The result of the feasibility study is a formal proposal. If the proposal is accepted, we continue with the project.

#### **FEASIBILTY CONSTRAINTS**

- **ECONOMIC FEASIBILITY**

Economic analysis is most frequently used method for evaluating the effectiveness of a candidate system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. The proposed system is economically feasible one. We do not want to keep lot of books for storing the data. By manipulating data using computer reduces cost. We do not want lot of employees; we simply want one to operate it Administrator.

- **TECHNICAL FEASIBILITY**

Technical feasibility centers on the existing computer system and to what extent it can support the proposed system. It involves financial considerations to accommodate technical enhancements. If the budget is a serious constraint, then the project is judged not feasible. Here we need only a computer working in low speed to accomplish the task.

- **BEHAVIOUR FEASIBILITY**

People inherently resist change and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have toward the development computerized system. The computer installation have something to do with turnover, retraining and changes .In the proposed system, it behaves very feasibly. It is very easy to train the people in the proposed system. We simply want to tell the purpose of each button and about a little data to enter.

### **2.3 STUDY OF PROPOSED SYSTEM.**

The proposed system has got the following advantages over the existing system:

- Users will receive better and quick service.
- Security is ensured by protecting the system with passwords.
- Normalized database tables eliminate data redundancy.
- Provision for minimizing errors in data entry.
- Efficient data storage.
- Real-time response and user-friendliness.
- Time saving.

### **3. SYSTEM DESIGN**

System design is a transaction from user-oriented documents to document oriented programmers or database personnel , it emphasis on translating performance specification into design specification and involves conceiving, planning and then carrying out the plan by generating the necessary reports and outputs. Design phase act as a bridge between the software requirement specification and implementation phase, which satisfies the requirements .In the design phase the detailed design of the system selected in the study phase is accomplished. Major steps in design are

- Method of data Captures and data input.
- Modification to be done to convert the existing system to the proposed system
- Operations to be performed to produce output and maintain the file
- Design input and output forms
- Output to be produced

A modular approach has been adapted in the development of the proposed system .Each module is designed for a specific application and they are operated independently.

**System Architecture:** The system architecture is the blueprint of the Gas Agency Management System, which includes the hardware, software, and network components. The system architecture design should be based on the requirement analysis and consider the scalability, reliability, and performance requirements. The system architecture should also be flexible and adaptable to accommodate future changes and upgrades.

**Database Design:** The database design is a critical aspect of the system design, as it involves the organization, storage, and management of the system data. The database design should be based on the requirement analysis and consider the data security, integrity, and availability requirements. The database design should also be optimized for query performance and scalability.

**Requirement Analysis:** The requirement analysis is the first step in the system design phase, which involves gathering and analyzing the requirements of the gas agency and its customers. This step is critical as it determines the system's scope, features, and functionalities. The requirement analysis process should involve various stakeholders, including the gas agency management, gas delivery agents, customers, and technical experts, to ensure that all perspectives are considered.

### **3.1 Entity Relationship (ER)DIAGRAM**

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education, and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships, and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

#### **Uses of entity relationship diagrams:**

**DATABASE DESIGN:** ER diagrams are used to model and design relational databases, in terms of logic and business rules (in a logical data model) and in terms of the specific technology to be implemented (in a physical data model.) In software engineering, an ER diagram is often an initial step in determining requirements for an information systems project. It's also later used to model a particular database or databases. A relational database has an equivalent relational table and can potentially be expressed that way as needed

**DATABASE TROUBLESHOOTING:** ER diagrams are used to analyse existing databases to find and resolve problems in logic or deployment. Drawing the diagram should reveal where it's going wrong.

### **BUSINESS INFORMATION SYSTEMS:**

The diagrams are used to design or to analyse relational databases used in business processes. Any business process that uses fielded data involving entities, actions and interplay can potentially benefit from a relational database. It can streamline processes, uncover information more easily and improve results.

### **BUSINESS PROCESS RE-ENGINEERING (BPR):**

ER diagrams help in analyzing databases used in business process re-engineering and in modeling a new database setup.

### **EDUCATION:**

Databases are today's method of storing relational information for educational purposes and later retrieval, so ER Diagrams can be valuable in planning those data structures.

### **RESEARCH:**

Since so much research focuses on structured data, ER diagrams can play a key role in setting up useful databases to analyze the data.

## **THE COMPONENTS AND FEATURES OF AN ER DIAGRAM**

ER Diagrams are composed of entities, relationships, and attributes. They also depict cardinality, which defines relationships in terms of numbers.

### **ENTITY**

A definable thing such as a person, object, concept, or event that can have data stored about it. Think of entities as nouns. Examples: a member , student, car or product. Typically shown as a rectangle.



ENTITY

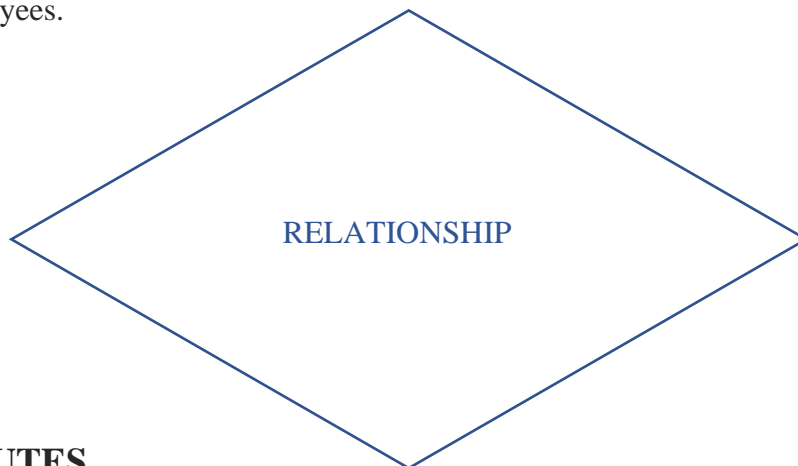


A weak entity is an entity that must be defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



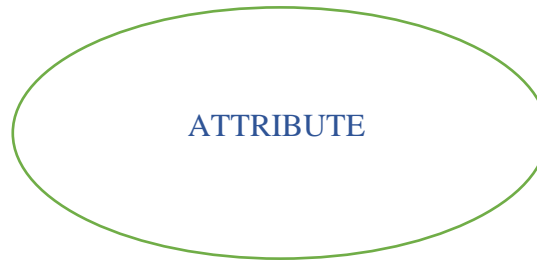
### ACTIONS

Actions are represented by diamond shapes, show how two entities share information in the database. In some cases, entities can be self-linked. For example, employees can supervise other employees.



### ATTRIBUTES

Attributes which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute. A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values. A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary. An employee entity can have multiple skill values. A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.



**Multivalued** attribute can have more than one value. For example, an employee entity can have multiple skill values.



**Derived** attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.



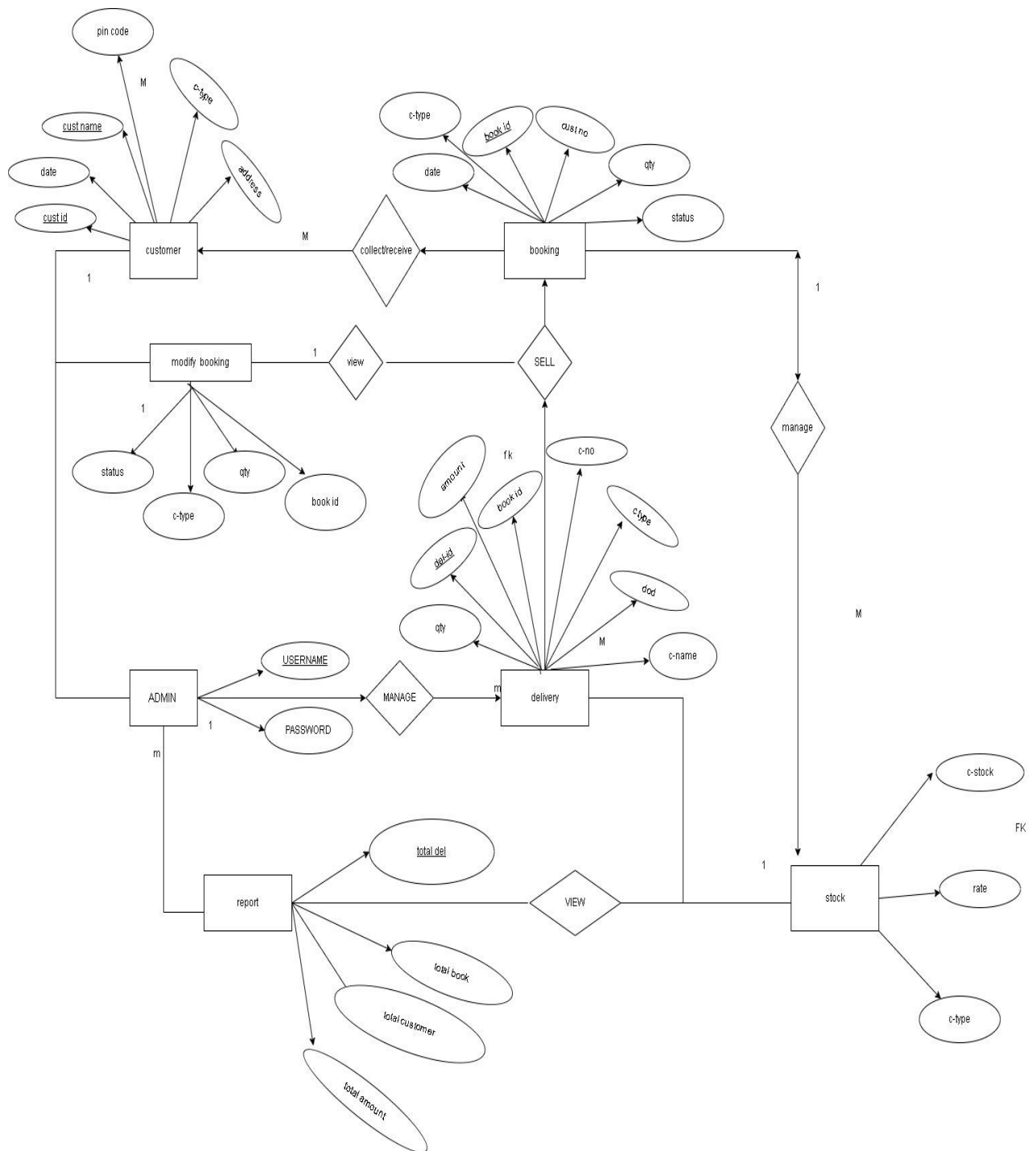
## CONNECTING LINES

The solid lines that connect attributes to show the relationships of entities in the diagram.

## CARDINALITY

It specifies how many instances of an entity relate to one instance of another entity. Ordinarily is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinarily describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinarily specifies the absolute minimum number of relationships.

**ER DIAGRAM :**



### **3.2 DATA FLOW DIAGRAM (DFD)**

Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to Systems Design. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job and shows the flow of data between the various parts of the system.

Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users can visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams can be drawn up and compared with the new system's data flow diagrams to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately influences the structure of the whole. In the course of developing a set of levelled data flow diagrams the analyst/designer is forced to address how the system may be decomposed into component sub-systems.

There are different notations to draw data flow diagrams, defining different visual representations for processes, data stores, data flow, and external entities.

#### **PHYSICAL VS. LOGICAL DFD**

A logical DFD captures the data flows that are necessary for a system to operate. It describes the processes that are undertaken, the data required and produced by each process, and the stores needed to hold the data. On the other hand, a physical DFD shows how the system is implemented, either at the moment (Current Physical DFD), or how the

designer intends it to be in the future (Required Physical DFD). Thus, a Physical DFD may be used to describe the set of data items that appear on each piece of paper that move around an office, and the fact that a particular set of pieces of paper are stored together in a filing cabinet. It is quite possible that a Physical DFD will include references to data that are duplicated, or redundant, and that the data stores, if implemented as a set, would constitute an unnormalized (or denormalized) relational database. In contrast, a Logical DFD attempts to capture the data flow aspects of a system in a form that has neither redundancy nor duplication.

## **HISTORY**

The original developer of structured design, based on Martin and Estrin's "Data Flow Graph" model of computation. Starting in the 1970s, data flow diagrams (DFD) became a popular way to visualize the major steps and data involved in software system processes. DFDs were usually used to show data flow in a computer system, although they could in theory be applied. DFD were useful to document the major data flows or to explore a new high-level design in terms of data flow.

## **DATA FLOW**

Represented by a unidirectional arrow. Data Flows show how data is moved through the System. Data Flows are labelled with a description of the data that is being passed through it. In our course, we need to understand and be able to draw 2 types of Data Flow Diagrams, they are Level-0 and Level 1 DFD 's. In this blog, I will hopefully make it easier to understand the differences between the two types of DFD 's and help understand how to draw a DFD for the exam. Firstly, I will look at level-0 DFD 's and give an example. Then I will look at Level 1 DFD 's and give an example. A level-0 DFD is the most basic form of DFD. It aims to show how the entire system works at a glance. There is only one process in the system and all the data flows either into or out of this process. Level-0 DFD 's demonstrates the interactions between the process and external entities. They do not contain Data Stores. When drawing Level-0 DFD 's, we must first identify the process, all the external entities and all the data flows. We must also state any assumptions we make about the system. It is advised that we

draw the process in the middle of the page. We then draw our external entities in the corners and finally connect our entities to our process with the data flows.

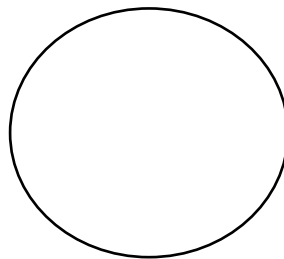
## DATA FLOW SYMBOLS AND THEIR MEANINGS

DFDs only involve four symbols.

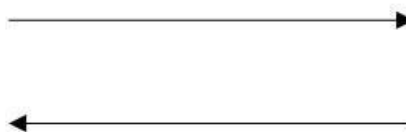
They are:

- Process
- Data flow
- Data Store
- External entity

**PROCESS:** Transform of incoming data flow(s) to outgoing flow(s).



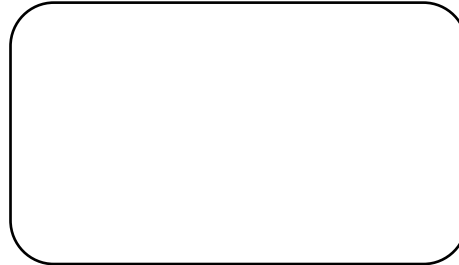
**DATA FLOW:** Movement of data in the system.



**DATA STORE:** Data repositories for data that are not moving. It may be as simple as a buffer or a queue or as sophisticated as a relational database.



**EXTERNAL ENTITY:** Sources of destinations outside the specified system boundary.



## STEPS TO CONSTRUCT DATA FLOW DIAGRAMS

**Four steps are commonly used to construct a DFD**

- Process should be named and numbered for easy reference. Each name should be representative of the process.
- The destination of flow is from top to bottom and from left to right.
- When a process is exploded in to lower level details they are numbered.
- The names of data stores, sources and destinations are written in capital letters.

## RULES FOR CONSTRUCTING A DATA FLOW DIAGRAM

- Arrows should not cross each other.
- Squares, circles, and files must bear names.
- Decomposed data flow squares and circles can have same names.
- Draw all data flow around the outside of the diagram.

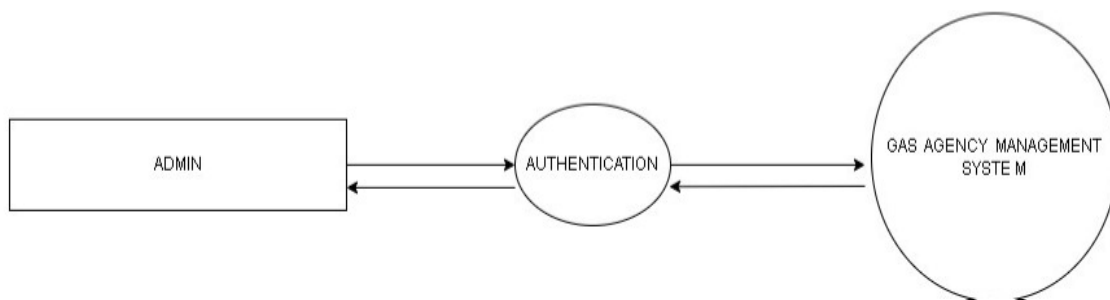
## DFD levels

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

**DFD Level 0** is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modelled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and develops.

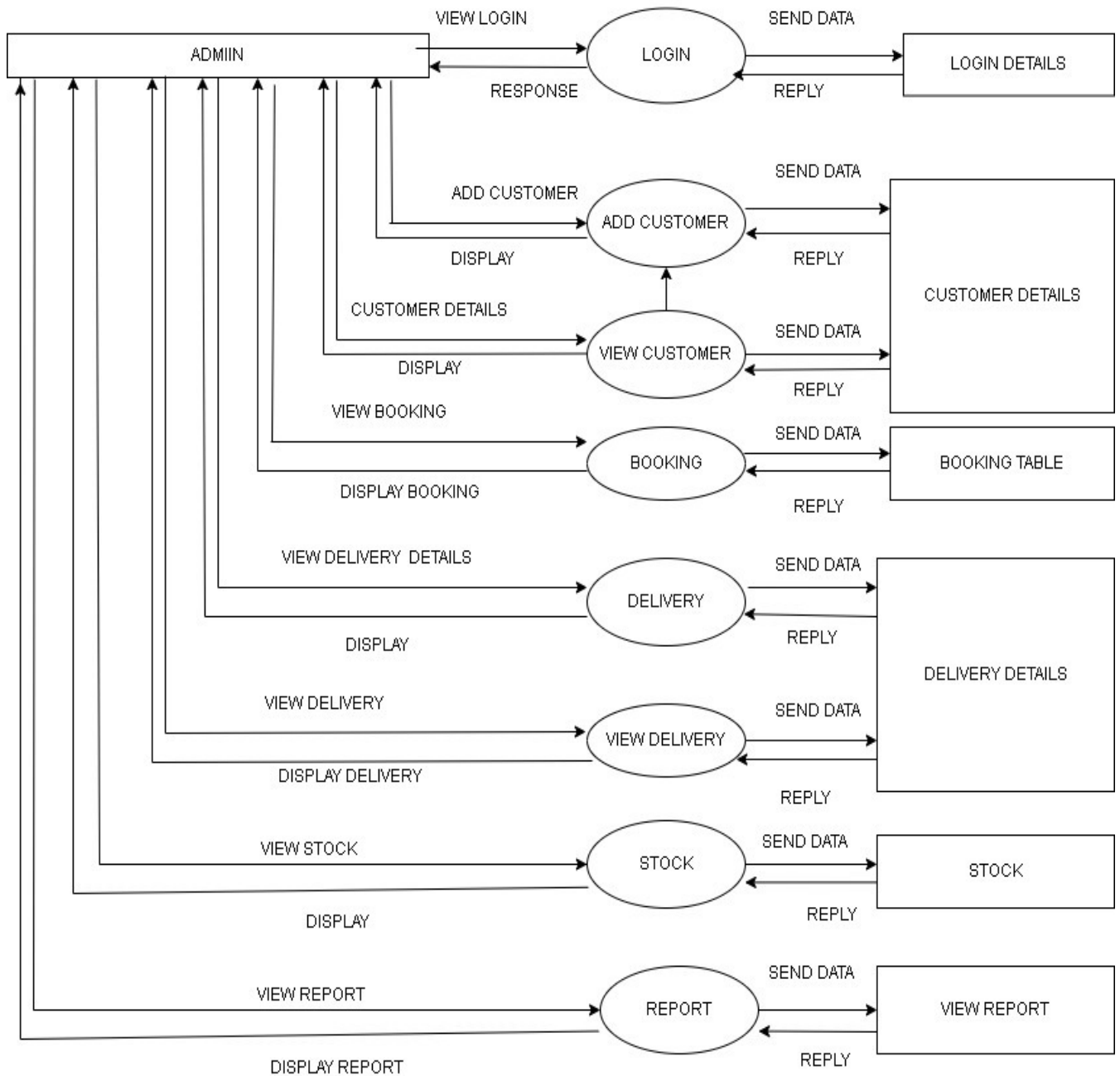
**DFD Level 1** provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.

### **DFD Level 0**





**DFD Level 1**



### **3.3 GANTT CHART**

A Gantt chart is a type of bar chart, devised by Henry Gantt in the 1910s, that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities.

#### **Historical Development**

The first known tool of this type was developed in 1896 by Karol Adamiecki, who called it a Harmon gram. Adamiecki did not publish his chart until 1931, however, and only in Polish, which limited both its adoption and recognition of his authorship. The chart is named after Henry Gantt (1861–1919), who designed his chart around the years 1910–1915. One of the first major applications of Gantt charts was by the United States during World War I, at the instigation of General William Crosier in the 1980s, personal computers allowed widespread creation of complex and elaborate Gantt charts. The first desktop applications were intended mainly for project managers and project schedulers. With the advent of the Internet and increased collaboration over networks at the end of the 1990s, Gantt charts became a common feature of web-based applications, including collaborative groupware.

#### **GANTT CHART BENEFITS**

##### **CLARITY:**

One of the biggest benefits of a Gantt chart is the tool's ability to boil down multiple tasks and timelines into a single document. Stakeholders throughout an organization can easily understand where teams are in a process while grasping the ways in which independent elements come together toward project completion.

##### **COMMUNICATION:**

Teams can use Gantt charts to replace meetings and enhance other status updates. Simply clarifying chart positions offers an easy, visual method to help team members understand task progress.

**MANAGEABILITY:**

For project managers handling complex assignments, like software publishing or event planning, the benefits of Gantt charts include externalizing assignments. By visualizing all of the pieces of a project puzzle, managers can make more focused, effective decisions about resources and timetables.

**EFFICIENCY:**

Another one of the benefits of Gantt charts is the ability for teams members to leverage each other's deadlines for maximum efficiency. For instance, while one team member waits on the outcome of three other tasks before starting a crucial piece of the assignment, he or she can perform other project tasks. Visualizing resource usage during projects allows managers to make better use of people, places, and things.

**ACCOUNTABILITY:**

When project teams face major organizational change, documenting effort and outcomes becomes crucial to career success. Using Gantt charts during critical projects allows both project managers and participants to track team progress, highlighting both big wins and major failures during professional review periods; team members who frequently exceed expectations can leverage this documentation into larger raises or bonuses.

**GANTT CHART IMPORTANCE**

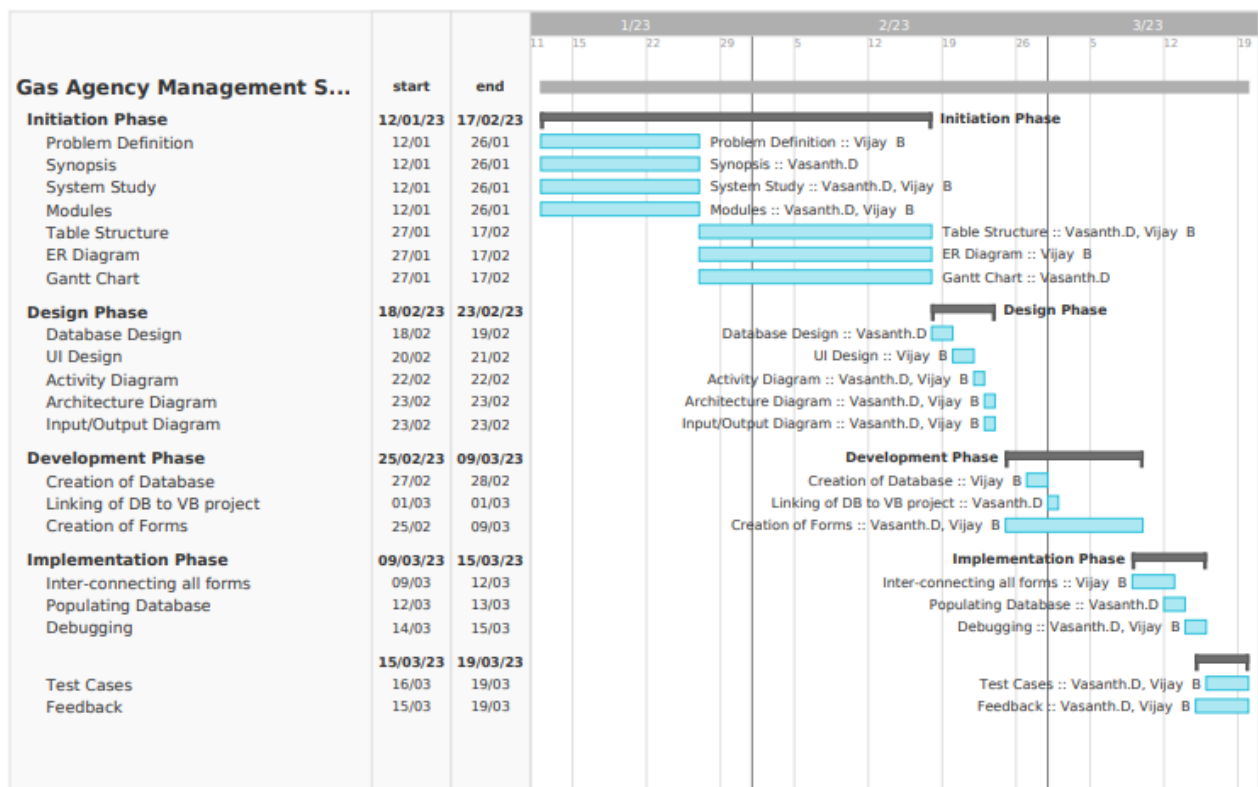
The project's summary and terminal elements, which combine to form the project's internal structure, are shown on the Gantt chart. Many charts will also depict the precedence rankings and dependencies of various tasks within the project. The charts can illustrate the start and finish project terminal elements in project management. It can also show summary elements and terminal dependencies. The smallest task tracked as part of the project effort is known as a terminal element. Gantt chart represents the tasks in most modern project scheduling packages. However other management applications use simpler communication tools such as message boards, to-do lists and simple scheduling etc., therefore, they do not use Gantt charts as heavily.

The way to create this chart begins by determining and listing the necessary activities. Next, sketch out how you expect the chart to look. List which items depend on others and what activities take place when.

For each activity, list how many man-hours it will require, and who is responsible. Lastly, determine the throughput time.

This technique's primary advantage is its good graphical overview that is easy to understand for nearly all project participants and stakeholders. Its primary disadvantage is its limited applicability for many projects since projects are often more complex than can be effectively communicated with this chart.

**GANTT CHART:**



### 3.4 INPUT/OUTPUT DESIGN

- Login page :

```

Public Class Form1
    Private Sub Guna2Button1_Click(sender As Object, e As EventArgs) Handles
Guna2Button1.Click
        If Username.Text.Length <= 0 Then
            MsgBox("Please enter Username!", MsgBoxStyle.Information)
            Username.Focus()
            Exit Sub
        ElseIf Password.Text.Length <= 0 Then
            MsgBox("Please enter Password!", MsgBoxStyle.Critical)
            Password.Focus()
            Exit Sub
        End If
        Dim str As String = "Data Source=VIJAY_BASKARAN\SQLEXPRESS;Initial
Catalog=Gas;Integrated Security=True"
        Dim sql As String = "select count(*) from Login where Username=@Username
and Password=@Password"
        Using Conn As New SqlConnection(str)
            Using cmd As New SqlCommand(sql, Conn)
                Conn.Open()
                cmd.Parameters.AddWithValue("@Username", Username.Text)
                cmd.Parameters.AddWithValue("@Password", Password.Text)
                Dim value = cmd.ExecuteScalar()
                If value > 0 Then
                    MessageBox.Show("Login sucessfull")
                    Home.Show()
                    Me.Hide()

                Else
                    MessageBox.Show("Invalid login please check")
                    Username.Clear()
                    Password.Clear()

                End If
            End Using
        End Using
    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    End Sub

    Private Sub Username_KeyDown(sender As Object, e As KeyEventArgs) Handles
Username.KeyDown
        If e.KeyCode = Keys.Enter Then
            e.SuppressKeyPress = True
            Password.Focus()
        End If
    End Sub
End Class

```

- **Home page :**

```
Public Class Home
```

```
Private Sub Label6_Click(sender As Object, e As EventArgs) Handles Label6.Click
```

```
End Sub
```

```
Private Sub Label1_Click(sender As Object, e As EventArgs) Handles Label1.Click
```

```
Add.Show()
```

```
Me.Hide()
```

```
End Sub
```

```
Private Sub Label2_Click(sender As Object, e As EventArgs) Handles Label2.Click
```

```
Booking.Show()
```

```
Me.Hide()
```

```
End Sub
```

```
Private Sub Label3_Click(sender As Object, e As EventArgs) Handles Label3.Click
```

```
Delivery.Show()
```

```
Me.Hide()
```

```
End Sub
```

```
Private Sub Label4_Click(sender As Object, e As EventArgs) Handles Label4.Click
```

```
Stock.Show()
```

```
Me.Hide()
```

```
End Sub
```

```
Private Sub Label5_Click(sender As Object, e As EventArgs) Handles Label5.Click
```

```
Report.Show()
```

```
Me.Hide()
```

```
End Sub
```

```
Private Sub Button1_Click(sender As Object, e As EventArgs)
```

```
End Sub
```

```
Private Sub PictureBox1_Click(sender As Object, e As EventArgs)
```

```
Add.Show()
```

```
Me.Hide()
```

```
End Sub
```

```
Private Function GetDebuggerDisplay() As String
```

```
Return ToString()
```

```
End Function
```

```
Private Sub Label7_Click(sender As Object, e As EventArgs) Handles Label7.Click
```

```
View.Show()
```

```
Me.Hide()
```

```
End Sub
```

```
Private Sub PictureBox2_Click(sender As Object, e As EventArgs) Handles
```

```
PictureBox2.Click
```

```
Add.Show()
```

```
Me.Hide()
```

```
End Sub
```

```
Private Sub PictureBox6_Click(sender As Object, e As EventArgs) Handles
```

```
PictureBox6.Click
```

```
Booking.Show()
```

```
Me.Hide()
```

```
End Sub

Private Sub PictureBox5_Click(sender As Object, e As EventArgs) Handles
PictureBox5.Click
    Delivery.Show()
    Me.Hide()
End Sub

Private Sub PictureBox4_Click(sender As Object, e As EventArgs) Handles
PictureBox4.Click
    Stock.Show()
    Me.Hide()
End Sub

Private Sub PictureBox3_Click(sender As Object, e As EventArgs) Handles
PictureBox3.Click
    Report.Show()
    Me.Hide()
End Sub

Private Sub PictureBox1_Click_1(sender As Object, e As EventArgs) Handles
PictureBox1.Click
    Application.Exit()
End Sub

Private Sub Label8_Click(sender As Object, e As EventArgs) Handles Label8.Click
    Dview.Show()
    Me.Hide()
End Sub

End Class
```

- **Add customers :**

```
Imports System.Data.SqlClient
Imports System.Windows.Forms.VisualStyles
Imports System.Windows.Forms.VisualStyles.VisualStyleElement

Public Class Add
    Private Sub PictureBox2_Click(sender As Object, e As EventArgs) Handles PictureBox2.Click
        Home.Show()
        Me.Hide()
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If Guna2TextBox1.Text = "" Or Guna2TextBox2.Text = "" Or Guna2TextBox3.Text = "" Or Guna2TextBox4.Text = "" Or Guna2TextBox5.Text = "" Or Guna2TextBox6.Text = "" Or Guna2TextBox7.Text = "" Then
            MsgBox("Please enter all the details.", MsgBoxStyle.Information)
        ElseIf Guna2TextBox4.Text.Length <> 10 Then
            MsgBox("Mobile number should be 10 digits.", MsgBoxStyle.Information)
        Else
            Dim conn As New SqlConnection("Data Source=VIJAY_BASKARAN\SQLEXPRESS;Initial Catalog=Gas;Integrated Security=True")
            Dim cmd As New SqlCommand("If not exists (Select * From Cust where ID=@ID) Insert Into Cust (ID,FName,LName,MobNo,Add1,Add2,Pin,DOC,CType)Values(@ID,@FName,@LName,@MobNo,@Add1,@Add2,@Pin,@DOC,@CType)", conn)
            cmd.Parameters.AddWithValue("ID", Guna2TextBox1.Text.Trim)
            cmd.Parameters.AddWithValue("FName", Guna2TextBox2.Text.Trim)
            cmd.Parameters.AddWithValue("LName", Guna2TextBox3.Text.Trim)
            cmd.Parameters.AddWithValue("MobNo", Guna2TextBox4.Text.Trim)
            cmd.Parameters.AddWithValue("Add1", Guna2TextBox5.Text.Trim)
            cmd.Parameters.AddWithValue("Add2", Guna2TextBox6.Text.Trim)
            cmd.Parameters.AddWithValue("Pin", Guna2TextBox7.Text.Trim)
            cmd.Parameters.AddWithValue("DOC", DateTimePicker1.Value)
            cmd.Parameters.AddWithValue("CType", Guna2ComboBox1.Text.Trim)
            conn.Open()
            If cmd.ExecuteNonQuery = 1 Then
                MsgBox("Customer details entered Successfully", MsgBoxStyle.Information)
            Else
                MsgBox(" Customer details already exists!!", MsgBoxStyle.Information)
            End If
            conn.Close()
        End If
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        Guna2TextBox1.Clear()
        Guna2TextBox2.Clear()
        Guna2TextBox3.Clear()
        Guna2TextBox4.Clear()
        Guna2TextBox5.Clear()
        Guna2TextBox6.Clear()
        Guna2TextBox7.Clear()
        DateTimePicker1.ResetText()
        Guna2ComboBox1.ResetText()
    End Sub
End Class
```



```

Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
    Me.Hide()
    View.Show()
End Sub
End Class

```

- **View Customers :**

```
Imports System.Data.SqlClient
```

```
Public Class View
```

```

    Private Sub PictureBox2_Click(sender As Object, e As EventArgs) Handles
PictureBox2.Click
        Me.Hide()
        Home.Show()
    End Sub

```

```

    Private Sub Button2_Click_1(sender As Object, e As EventArgs) Handles
Button2.Click
        Guna2TextBox1.Clear()
        Guna2TextBox2.Clear()
        Guna2TextBox3.Clear()
        Guna2TextBox4.Clear()
        Guna2TextBox5.Clear()
        Guna2TextBox6.Clear()
        Guna2TextBox7.Clear()
        Guna2TextBox8.Clear()
        Guna2TextBox9.Clear()
    End Sub

```

```

    Private Sub Button3_Click_1(sender As Object, e As EventArgs) Handles
Button3.Click
        If Guna2TextBox1.Text = "" Then
            MsgBox("Please enter a value.", MsgBoxStyle.Information)
        Else
            Dim conn As New SqlConnection("Data
Source=VIJAY_BASKARAN\SQLEXPRESS;Initial Catalog=Gas;Integrated Security=True")
            conn.Open()
            Dim cmd As New SqlCommand("Select
FName,LName,MobNo,Add1,Add2,Pin,DOC,CType from Cust where Id=@Id", conn)
            cmd.Parameters.AddWithValue("ID", Guna2TextBox1.Text)
            Dim myreader As SqlDataReader
            myreader = cmd.ExecuteReader
            If myreader.Read Then
                Guna2TextBox2.Text = myreader("FName")
                Guna2TextBox3.Text = myreader("LName")
                Guna2TextBox4.Text = myreader("Mobno")
                Guna2TextBox5.Text = myreader("Add1")
                Guna2TextBox6.Text = myreader("Add2")
                Guna2TextBox7.Text = myreader("Pin")
                Guna2TextBox8.Text = myreader("DOC")
                Guna2TextBox9.Text = myreader("Ctype")
            End If
        End If
    End Sub
End Class

```

- **Booking :**

```
Imports Guna.UI2.WinForms
Imports System.Data.SqlClient
Imports System.Text
```

```
Public Class Booking
```

```
    Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
        Home.Show()
        Me.Hide()
    End Sub
```

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If String.IsNullOrEmpty(TextBox1.Text) Or
String.IsNullOrEmpty(TextBox2.Text) Or String.IsNullOrEmpty(TextBox4.Text)
Or NumericUpDown1.Value = 0 Then
            MsgBox("Please fill all required fields", MsgBoxStyle.Exclamation)
            Return
        End If
        If String.IsNullOrEmpty(ComboBox1.Text) Then
            MsgBox("Please select the Stock status.", MsgBoxStyle.Exclamation)
            Return
        End If
```

```
        Dim conn As New SqlConnection("Data Source=VIJAY_BASKARAN\SQLEXPRESS;Initial
Catalog=Gas;Integrated Security=True")
```

```
        Dim qty As Integer = NumericUpDown1.Value
        Dim cty As String = TextBox5.Text
```

```
        conn.Open()
```

```
        ' Select data from Book table
```

```
        Dim selectCmd As New SqlCommand("SELECT * FROM Book WHERE Bid=@Bid", conn)
        selectCmd.Parameters.AddWithValue("@Bid", TextBox1.Text.Trim)
        Dim reader As SqlDataReader = selectCmd.ExecuteReader()
        Dim dataExists As Boolean = reader.HasRows
        reader.Close()
```

```
        If Not dataExists Then
```

```
            ' Insert data into Book table
```

```
            Dim insertCmd As New SqlCommand("INSERT INTO Book
(Bid,Id,DOB,Qty,Price,CType,Cstatus)
VALUES(@Bid,@ID,@DOB,@Qty,@Price,@CType,@Cstatus)", conn)
```

```
            insertCmd.Parameters.AddWithValue("@Bid", TextBox1.Text.Trim)
            insertCmd.Parameters.AddWithValue("@ID", TextBox2.Text.Trim)
            insertCmd.Parameters.AddWithValue("@Cstatus", ComboBox1.Text.Trim)
            insertCmd.Parameters.AddWithValue("@Price", TextBox4.Text.Trim)
            insertCmd.Parameters.AddWithValue("@CType", cty)
            insertCmd.Parameters.AddWithValue("@DOB", DateTimePicker1.Value)
            insertCmd.Parameters.AddWithValue("@Qty", qty)
```

```
            If insertCmd.ExecuteNonQuery() = 1 Then
                MsgBox("Your booking is confirmed", MsgBoxStyle.Information)
```

```
            Else
                MsgBox("Error inserting data into Book table", MsgBoxStyle.Critical)
```

```
            End If
```

```
        Else
```

```
            MsgBox("Data already exists!!", MsgBoxStyle.Information)
```

```
        End If
```

```

' Update data in stock table
Dim updateCmd As New SqlCommand("UPDATE stock SET Qty = Qty - @Qty WHERE CType
= @CType", conn)
updateCmd.Parameters.AddWithValue("@CType", cty)
updateCmd.Parameters.AddWithValue("@Qty", qty)
updateCmd.ExecuteNonQuery()

conn.Close()
End Sub

Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
    TextBox2.Clear()
    ComboBox1.ResetText()
    TextBox4.Clear()
    NumericUpDown1.ResetText()
    TextBox5.Clear()
End Sub

Private Sub Booking_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Dim rnd As New Random()
    Dim str As String = "ABCDEFGHJKLMNOPQRSTUVWXYZ123456789"
    Dim result As New StringBuilder()

    For i As Integer = 1 To 7
        Dim index As Integer = rnd.Next(0, str.Length)
        result.Append(str(index))
    Next
    TextBox1.Text = result.ToString()
End Sub

Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
    If String.IsNullOrEmpty(TextBox2.Text) Then
        MsgBox("Please enter the customer ID.", MsgBoxStyle.Information)
        Return
    End If
    Dim connString As String = "Data Source=VIJAY_BASKARAN\SQLEXPRESS;Initial
Catalog=Gas;Integrated Security=True"
    Dim conn As New SqlConnection(connString)
    Dim cmd As New SqlCommand("SELECT COUNT(*) FROM Book WHERE Id=@Id", conn)
    Dim cmd2 As New SqlCommand("SELECT Cust.CType, stock.Ppc FROM Cust INNER JOIN
stock ON Cust.CType = stock.CType WHERE Cust.Id=@Id", conn)
    Dim cmd3 As New SqlCommand("SELECT COUNT(*) FROM Cust WHERE Id=@Id", conn)
    cmd.Parameters.AddWithValue("@Id", TextBox2.Text)
    cmd2.Parameters.AddWithValue("@Id", TextBox2.Text)
    cmd3.Parameters.AddWithValue("@Id", TextBox2.Text)
    conn.Open()
    Dim count As Integer = Convert.ToInt32(cmd.ExecuteScalar())
    Dim custCount As Integer = Convert.ToInt32(cmd3.ExecuteScalar())
    Dim reader As SqlDataReader = cmd2.ExecuteReader()
    If custCount = 0 Then
        MsgBox("Invalid customer ID")
    ElseIf reader.Read() Then
        ' Retrieve CType and Price from reader
        Dim cTy As String = reader("CType").ToString()
        Dim price As Decimal = Convert.ToDecimal(reader("ppc"))

        ' Display CType and Price
        TextBox5.Text = cTy
        TextBox4.Text = price

        If count > 0 Then

```

```

        MsgBox("This month's booking is done")
    Else
        MsgBox("Proceed to book")
    End If
End If
reader.Close()
conn.Close()
End Sub

End Class

```

- **Stock :**

```

Imports System.Data.SqlClient
Imports System.Windows.Forms.VisualStyles.VisualStyleElement
Imports Guna.UI2.WinForms

```

```
Public Class Stock
```

```

    Private Sub PictureBox2_Click(sender As Object, e As EventArgs) Handles
PictureBox2.Click
        Home.Show()
        Me.Hide()
    End Sub

```

```

    Private Sub ComboBox1_DropDown(sender As Object, e As EventArgs) Handles
ComboBox1.DropDown
        Dim connectionString As String = "Data
Source=VIJAY_BASKARAN\SQLEXPRESS;Initial Catalog=Gas;Integrated Security=True"
        Dim connection As New SqlConnection(connectionString)
        Dim query As String = "SELECT CType,Ppc FROM Stock"
        Dim adapter As New SqlDataAdapter(query, connection)
        Dim dataTable As New DataTable()
        adapter.Fill(dataTable)
        ComboBox1.DataSource = dataTable
        ComboBox1.DisplayMember = "CType"
        ComboBox1.ValueMember = "Ppc"
    End Sub

```

```

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim conn As New SqlConnection("Data Source=VIJAY_BASKARAN\SQLEXPRESS;Initial
Catalog=Gas;Integrated Security=True")
        conn.Open()
        Dim cmd As New SqlCommand("Select CType,Qty,Ppc from Stock where
Ctype=@Ctype", conn)
        cmd.Parameters.AddWithValue("Ctype", ComboBox1.Text)
        Dim myreader As SqlDataReader
        myreader = cmd.ExecuteReader
        If myreader.Read Then
            TextBox2.Text = myreader("Qty")
            TextBox1.Text = myreader("Ppc")
        End If
    End Sub

```

```

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        TextBox2.Clear()
        TextBox1.Clear()
        ComboBox1.ResetText()
    End Sub

```

```

Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
    Dim connectionString As String = "Data
Source=VIJAY_BASKARAN\SQLEXPRESS;Initial Catalog=Gas;Integrated Security=True"
    Dim conn As New SqlConnection(connectionString)
    Dim CType As String
    Dim newQuantity, newPpc As Integer

    If Integer.TryParse(TextBox3.Text, newQuantity) AndAlso
Integer.TryParse(TextBox4.Text, newPpc) Then
        CType = ComboBox2.Text
        Dim query As String = "UPDATE Stock SET Qty = Qty + @NewQuantity, PPC =
@NewPpc WHERE CType = @CType"
        Dim cmd As New SqlCommand(query, conn)
        cmd.Parameters.AddWithValue("@NewQuantity", newQuantity)
        cmd.Parameters.AddWithValue("@NewPpc", newPpc)
        cmd.Parameters.AddWithValue("@Ctype", CType)
        conn.Open()
        cmd.ExecuteNonQuery()
        MsgBox("Stock Updated Successfully")
    Else
        MessageBox.Show("Please enter a valid Quantity and Price Per Unit value.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End If

    conn.Close()
End Sub

Private Sub ComboBox2_DropDown(sender As Object, e As EventArgs) Handles
ComboBox2.DropDown
    Dim connectionString As String = "Data
Source=VIJAY_BASKARAN\SQLEXPRESS;Initial Catalog=Gas;Integrated Security=True"
    Dim connection As New SqlConnection(connectionString)
    Dim query As String = "SELECT CType,Qty FROM Stock"
    Dim adapter As New SqlDataAdapter(query, connection)
    Dim dataTable As New DataTable()
    adapter.Fill(dataTable)
    ComboBox2.DataSource = dataTable
    ComboBox2.DisplayMember = "CType"

End Sub

Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
    TextBox3.Clear()
    TextBox4.Clear()
    ComboBox2.ResetText()
End Sub
End Class

```

- **Delivery :**

```
Imports Guna.UI2.WinForms
Imports System.Data.SqlClient
Imports System.Windows.Forms.VisualStyles.VisualStyleElement

Public Class Delivery

    Private Sub PictureBox2_Click(sender As Object, e As EventArgs) Handles PictureBox2.Click
        Home.Show()
        Me.Hide()
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim conn As New SqlConnection("Data Source=VIJAY_BASKARAN\SQLEXPRESS;Initial
Catalog=Gas;Integrated Security=True")
        If String.IsNullOrEmpty(TextBox2.Text.Trim()) OrElse
String.IsNullOrEmpty(TextBox4.Text.Trim()) OrElse
String.IsNullOrEmpty(TextBox5.Text.Trim()) OrElse
String.IsNullOrEmpty(ComboBox1.Text.Trim()) OrElse
String.IsNullOrEmpty(ComboBox2.Text.Trim()) Then
            MsgBox("Please fill all required fields!", MessageBoxButtons.OK,
MessageBoxIcon.Warning)
        Else
            Dim cmd As New SqlCommand("If not exists (Select * From Delivery where
DID=@DID)
            Insert Into
Delivery(DID,CName,Dod,Dadd,MobNo,Dstat,Pays,T_amt)Values(@DID,@CName,@Dod,@Dadd,@MobN
o,@Dstat,@Pays,@T_amt)", conn)
            cmd.Parameters.AddWithValue("DID", TextBox1.Text.Trim)
            cmd.Parameters.AddWithValue("CName", TextBox2.Text.Trim)
            cmd.Parameters.AddWithValue("Dod", DateTimePicker1.Value)
            cmd.Parameters.AddWithValue("Dadd", TextBox4.Text.Trim)
            cmd.Parameters.AddWithValue("MobNo", TextBox5.Text.Trim)
            cmd.Parameters.AddWithValue("Dstat", ComboBox1.Text.Trim)
            cmd.Parameters.AddWithValue("Pays", ComboBox2.Text.Trim)
            cmd.Parameters.AddWithValue("T_amt", 0)
            Dim currentValue As Integer = Integer.Parse(TextBox1.Text)
            currentValue += 1
            TextBox1.Text = currentValue.ToString()
            conn.Open()
            If cmd.ExecuteNonQuery() Then
                MsgBox("Data Inserted Successfully!", MessageBoxButtons.OK,
MessageBoxIcon.Information)
            Else
                MsgBox("Data already exists!!", MsgBoxStyle.Information)
            End If
            conn.Close()
            TextBox2.ResetText()
            DateTimePicker1.ResetText()
            TextBox4.ResetText()
            TextBox5.ResetText()
            ComboBox1.ResetText()
            ComboBox2.ResetText()
        End If
    End Sub

    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        Me.Hide()
        Dview.Show()
    End Sub
End Class
```

```

Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
    TextBox1.Clear()
    TextBox2.Clear()
    TextBox3.Clear()
    DateTimePicker1.ResetText()
    TextBox4.Clear()
    TextBox5.ResetText()
    TextBox6.ResetText()
    ComboBox1.ResetText()
    ComboBox2.ResetText()
End Sub

Private Sub TextBox6_KeyDown(sender As Object, e As KeyEventArgs) Handles
TextBox6.KeyDown
    If e.KeyCode = Keys.Enter Then ' Check if Enter key is pressed
        If TextBox6.Text = "" Then
            MsgBox("Please enter customer ID.", MsgBoxStyle.Information)
        Else
            Dim conn As New SqlConnection("Data
Source=VIJAY_BASKARAN\SQLEXPRESS;Initial Catalog=Gas;Integrated Security=True")
            conn.Open()
            Dim cmd As New SqlCommand("SELECT Book.Id, Book.Qty, Book.Price,
Cust.Id, Cust.Fname, Cust.Lname, Cust.MobNo, Cust.Add1, Cust.Add2 FROM Book INNER JOIN
Cust ON Book.Id = Cust.Id WHERE Book.Id=@Id", conn)
            cmd.Parameters.AddWithValue("@Id", TextBox6.Text)
            Dim myreader As SqlDataReader
            myreader = cmd.ExecuteReader
            If myreader.Read Then
                TextBox2.Text = myreader("Fname") & " " & myreader("Lname")
                TextBox4.Text = myreader("Add1") & " , " & myreader("Add2")
                TextBox5.Text = myreader("MobNo")
                TextBox3.Text = (myreader("Price") * myreader("Qty")).ToString()
            Else
                MsgBox("No customer found with this ID.", MsgBoxStyle.Information)
            End If
            conn.Close()
        End If
    End If
End Sub

End Class

```

- **View Delivery :**

```
Imports System.Data.SqlClient
```

```
Imports System.Windows.Forms.VisualStyleElement
```

```
Imports Guna.UI2.WinForms
```

```
Public Class Dview
```

```
    Private cmd As Object
```

```
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
```

```
        TextBox1.Clear()
```

```
        TextBox2.Clear()
```

```
        DateTimePicker1.ResetText()
```

```
        TextBox4.Clear()
```

```
        TextBox5.Clear()
```

```
        TextBox6.Clear()
```

```
        TextBox7.Clear()
```

```
    End Sub
```

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
```

```
        If TextBox1.Text = "" Then
```

```
            MsgBox("Please enter Delivery ID.", MsgBoxStyle.Information)
```

```
        Else
```

```
            Dim conn As New SqlConnection("Data Source=VIJAY_BASKARAN\SQLEXPRESS;Initial Catalog=Gas;Integrated Security=True")
```

```
            conn.Open()
```

```
            Dim cmd As New SqlCommand("Select DID,CName,Dod,Dadd,MobNo,Dstat,Pays from Delivery where DID=@DID", conn)
```

```
            cmd.Parameters.AddWithValue("DID", TextBox1.Text)
```

```
            Dim myreader As SqlDataReader
```

```
            myreader = cmd.ExecuteReader
```

```
            If myreader.Read Then
```

```
                TextBox2.Text = myreader("CName")
```

```
                DateTimePicker1.Text = myreader("Dod")
```

```
                TextBox4.Text = myreader("Dadd")
```

```
                TextBox5.Text = myreader("MobNo")
```

```
                TextBox6.Text = myreader("Dstat")
```

```
                TextBox7.Text = myreader("Pays")
```

```
            Else
```

```
                MsgBox("Invalid Delivery ID", MsgBoxStyle.Exclamation)
```

```
            End If
```

```
        End If
```

```
    End Sub
```

```
    Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
```

```
        Home.Show()
```

```
        Me.Hide()
```

```
    End Sub
```



- **Report :**

```
Imports System.Data.SqlClient
```

```
Imports System.Windows.Forms.VisualStyleElement
```

```
Public Class Report
```

```
    Dim connString As String = "Data Source=VIJAY_BASKARAN\SQLEXPRESS;Initial Catalog=Gas;Integrated Security=True"
```

```
    Dim conn As New SqlConnection(connString)
```

```
    Dim con As New SqlConnection
```

```
    Dim cmd As New SqlCommand
```

```
    Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
```

```
        Home.Show()
```

```
        Me.Hide()
```

```
    End Sub
```

```
    Private Sub Records_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        LoadData()
```

```
        Dim connectionString As String = New SqlConnectionStringBuilder() With {
            .DataSource = "VIJAY_BASKARAN\SQLEXPRESS", .InitialCatalog = "Gas",
            .IntegratedSecurity = True}.ConnectionString
```

```
        Dim connection As New SqlConnection(connectionString)
```

```
        connection.Open()
```

```
        Dim command As New SqlCommand("SELECT Book.ID, Delivery.CName, Book.Bid, Book.DOB, Book.Qty, Book.Price, Book.Ctype, Delivery.Dod, Delivery.Dstat, Delivery.Pays FROM Book JOIN Delivery ON ID = ID", connection)
```

```
        Dim data As New DataTable()
```

```
        Dim adapter As New SqlDataAdapter(command)
```

```
        adapter.Fill(data)
```

```
        DataGridView1.DataSource = data
```

```
        connection.Close()
```

```
    End Sub
```

```
    Private Sub LoadData()
```

```
        conn.Open()
```

```
        Dim query As String = "SELECT * from Book , Delivery "
```

```
        Dim da As New SqlDataAdapter(query, conn)
```

```
        Dim dt As New DataTable()
```

```
        da.Fill(dt)
```

```
        DataGridView1.DataSource = dt
```

```
        conn.Close()
```

```
    End Sub
```

```
    Private Sub TextBox1_KeyDown(sender As Object, e As KeyEventArgs) Handles TextBox1.KeyDown
```

```
        If e.KeyCode = Keys.Enter Then
```

```
            Dim ID As Integer
```

```
            If Integer.TryParse(TextBox1.Text, ID) Then
```

```
                FilterDataGridView(ID)
```

```
            End If
```

```
        End If
```

```
    End Sub
```

```
Private Sub FilterDataGridView(ID As Integer)
    Dim dataTable As DataTable = CType(DataGridView1.DataSource, DataTable)
    If dataTable IsNot Nothing AndAlso dataTable.Rows.Count > 0 Then
        Try
            dataTable.DefaultView.RowFilter = "ID = " & ID.ToString()
        Catch ex As Exception
            MessageBox.Show("Error: " & ex.Message)
        End Try
    End If
End Sub
Private Sub DateTimePicker1_ValueChanged(sender As Object, e As EventArgs) Handles
DateTimePicker1.ValueChanged
    Dim filterDate As String = DateTimePicker1.Value.ToString("yyyy-MM-dd")
    Dim dataTable As DataTable = CType(DataGridView1.DataSource, DataTable)
    If dataTable IsNot Nothing AndAlso dataTable.Rows.Count > 0 Then
        Try
            dataTable.DefaultView.RowFilter = "DOB = '" & filterDate & "'"
        Catch ex As Exception
            MessageBox.Show("Error: " & ex.Message)
        End Try
    End If
End Sub
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    TextBox1.ResetText()
    DateTimePicker1.ResetText()
End Sub
```

## **4. SYSTEM CONFIGURATION**

### **4.1 HARDWARE REQUIREMENTS**

|                  |                           |
|------------------|---------------------------|
| RAM              | 4.00GB                    |
| HARD DISK        | SSD                       |
| PROCESSOR        | Intel(R) Celeron(R) N4020 |
| PROCESSING SPEED | CPU 1.10GHz 1.10 GHz      |

### **4.2 SOFTWARE REQUIREMENTS**

|                  |                                 |
|------------------|---------------------------------|
| FRONT END        | VB.NET                          |
| BACK END         | MICROSOFT SQL SERVER 19         |
| TOOLS            | VISUAL STUDIO 2022              |
| OPERATING SYSTEM | WINDOWS 11 Home Single Language |
| DOCUMENTATION    | MICROSOFT WORD 365              |

## **5. DETAILS OF SOFTWARE**

### **5.1 OVERVIEW OF FRONTEND**

VB.NET as frontend, Microsoft Visual Studio 2022 is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silver light. It can produce both native code and managed code.

Visual Studio supports different programming languages and allows the code editor and debugger to support nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, Visual C++ and VB.NET. Support for other languages such as Python, Ruby, Node.js, and M among others is available via language services installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Java (and J#) were supported in the past.

Microsoft provides a free version of Visual Studio called the Community edition that supports plug-in and is available at no cost for all users. Support for programming languages is added by using a specific Package called a Language Service. A language service defines various interfaces which the VS Package implementation can implement to add support for various functionalities. Functionalities that can be added this way include syntax coloring, statement completion, brace matching, parameter information tooltips, member lists and error markers for background compilation. If the interface is implemented, the functionality will be available for the language.

### **FEATURES**

- Boolean Conditions
- Automatic Garbage Collection Standard Library
- Assembly Versioning
- Properties and Events

## ADVANTAGES

- The structure of the Basic programming language is very simple, particularly as to the executable code.
- VB is not only a language but primarily an integrated, interactive development environment.
- The VB-IDE has been highly optimized to support rapid application development (—RAD $\parallel$ ). It is particularly easy to develop and to connect them to handler functions provided by the application.
- The graphical user interface of the VB-IDE provides intuitively appealing views for the management of the program structure in the large and the various types of entities (classes, modules, procedures, forms, ...).
- VB provides a comprehensive interactive and context-sensitive online help system. When editing program texts the technology informs you in a little popup window about the types of constructs that may be entered at the current cursor location.
- VB is a component integration language which is attuned to Microsoft 's Component Object Model (—COM $\parallel$ ).
- COM components can be written in different languages and then integrated using VB.
- Interfaces of COM components can be easily called remotely via Distributed COM (—DCOM $\parallel$ ), which makes it easy to construct distributed applications.

## DISADVANTAGES

- Visual basic is a proprietary programming language written by Microsoft, so programs written in Visual basic cannot, easily, be transferred to other operating systems.
- There are some, fairly minor disadvantages compared with C. C has better declaration of arrays – it's possible to initialize an array of structures in C at declaration time; this is impossible in VB

## **THE SOFTWARE INCLUDES**

- Vb.net as a front end • SQL server as a back end
- Maintaining User details.
- Maintaining Payment details.
- Maintaining Student details.
- Maintaining Student Receipt details.
- Maintaining Courses with Username details.

The main objective of this system is to provide a simple course registering software which will be easy for the students to manage the main source is the management which is done at the admin side had more easy task for the admin to control simple registrations of the students the providing of Receipt is also available and can be retrieve by the student and easy payments can be done by only UPI Scan and Pay and with Cards.

## **5.2 OVERVIEW OF BACKEND**

SQL server as backend, SQL stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database or retrieve data from a database. Some common relational database management systems that use SQL are:

Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc.

Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually used on their system. However, the standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used.

## **DATA TYPES**

- Many data types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes long, FLOAT, DOUBLE, CHARVARCHAR, BINARY, VARBINARY, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, and Open GIS spatial types.

- Fixed-length and variable-length string types.
- Statements and Functions.
- Full operator and function support in the SELECT list and WHERE clause of queries. For example: `mysql> SELECT Cust(first_name, ' ', last_name) FROM citizen.`
- WHERE `income/dependents > 10000 AND age > 30`; Full support for SQL GROUP.
- Support for LEFT OUTER JOIN and RIGHT OUTER JOIN with both standard SQL and ODBC syntax.
- Support for aliases on tables and columns as required by standard SQL.
- Support for DELETE, INSERT, REPLACE, and UPDATE to return the number of rows that were changed (affected), or to return the number of rows matched instead by setting a flag when connecting to the server.
- Support for MySQL -specific SHOW statements that retrieve information about databases, storage engines, tables, and indexes. Support for the INFORMATION\_SCHEMA database, implemented according to standard SQL.
- An EXPLAIN statement to show how the optimizer resolves a query.
- Independence of function names from table or column names. For example, ABS is a valid column name. The only restriction is that for a function call, no spaces are permitted between the function names.
- You can refer to tables from different databases in the same statement.

## **SECURITY**

- A privilege and password system that is very flexible and secure, and that enables host-based verification.
- Password security by encryption of all password traffic when you connect to a server.

## **SCALABILITY AND LIMITS**

- Support for large databases. We use MySQL Server with databases that contain 50 million records. We also know of users who use MySQL Server with 200,000 tables and about 5,000,000,000 rows.
- Support for up to 64 indexes per table. Each index may consist of 1 to 16 columns or parts of columns. The maximum index width for tables is either 767 bytes or 3072 bytes. The

maximum index width for tables is 1000 bytes. An index may use a prefix of a column for, or column types.

## CONNECTIVITY

- Clients can connect to MySQL Server using several protocols:
- Clients can connect using TCP/IP sockets on any platform.
- On Windows systems, clients can connect using named pipes if the server is started with the option. Windows servers also support shared-memory connections if started with the option. Clients can connect through shared memory by using the protocol=memory option.
- On Unix systems, clients can connect using Unix domain socket files. MySQL client programs can be written in many languages. A client library written in C is available for clients written in C or C++, or for any language that provides C bindings. APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl are available, enabling MySQL clients to be written in many languages. The Connector/ODBC (MyODBC) interface provides MySQL support for client programs that use ODBC (Open Database Connectivity) connections. For example, you can use MS Access to connect to your MySQL server. Clients can be run on Windows or Unix. Connector/ODBC source is available. All ODBC 2. connectivity with MySQL. It implements the required ADO.NET interfaces and integrates into ADO.NET aware tools. Developers can build applications using their choice of .NET languages. MySQL Connector/Net is a fully managed ADO.NET driver written in 100% pure C#.

## LOCALIZATION

- The server can provide error messages to clients in many languages.
- Full support for several different character sets, including latin1 (cp1252), german, big5, ujis, several Unicode characters sets, and more. For example, the Scandinavian characters —åll, —äll and —öll are permitted in table and column names.
- All data is saved in the chosen 5 functions are supported, as are many others. MySQL Connector/Net enables developers to easily create .NET applications that require secure, high performance data character set.



- Sorting and comparisons are done according to the chosen character set and collation (using latin1 and Swedish collation by default). It is possible to change this when the MySQL server is started. To see an example of very advanced sorting, look at the Czech sorting code
- MySQL Server supports many different character sets that can be specified at compile time and runtime.
- The server time zone can be changed dynamically, and individual clients can specify their own time zone.

### **5.3 ABOUT THE PLATFORM**

- Windows is a series of Operating Systems developed by Microsoft. Each version of Windows includes a Graphical User Interface, with a desktop that allows users to view files and folders in Windows. For the past two decades, Windows has been the most widely used operating system for personal computers PCs.
- Microsoft Windows is designed for both home computing and professional purposes. Past versions of Windows home editions include Windows 3.0 (1990), Windows 3.1 (1992), Windows 95 (1995), Windows 98 (1998), Windows Me (2000), Windows XP (2001), and Windows Vista (2006). The current version, Windows 7, was released in 2009.
- The first business-oriented version of Windows, called Windows NT 3.1, was in 1993. This was followed by Windows 3.5, 4.0, and Windows 2000. When Microsoft released Windows XP in 2001, the company simply created different editions of the operating system for personal and business purposes. Windows Vista and Windows 7 have followed the same release strategy.
- Windows is designed to run on standard x86 hardware, such as Intel and AMD processors. Therefore, it can be installed on multiple brands of hardware, such as Dell, HP, and Sony computers, as well as home-built PCs. Windows 7 also includes several touch screen features, that allow the operating system to run on certain tablets and computers with touch screen displays. Microsoft's mobile operating system, Windows Phone, is designed specifically for smart phones and runs on several brands of phones,

including HTC, Nokia, and Samsung. Project management is a one-time carefully planned and organized effort to achieve a specific goal. Project management includes Developing a project plan, which includes defining project goals and objectives, specifying tasks or how goals will be achieved, what resources are need, and associating budgets and timelines for completion Implementing the project plan, carefully to make sure the plan is being managed according to plan.

## **.NET FRAMEWORK**

- .NET Framework (pronounced as "dot net") is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library named Framework
  - Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software memory management, and exception handling. As such, computer code written using .NET Framework is called "managed code". FCL and CLR together constitute the .NET Framework.
- FCL provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their source code with .NET Framework and other libraries. The framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an integrated development environment largely for .NET software called Visual Studio.
- .NET Framework began as proprietary software, although the firm worked to standardize the software stack almost immediately, even before its first release. Environment (in contrast to a hardware environment) named Common Language Runtime(CLR), an application virtual machine that provides services such as security,
- Despite the standardization efforts, developers, mainly those in the free and open-source software communities, expressed their unease with the selected terms and the prospects of any free and open-source implementation, especially regarding software patents. Since then, Microsoft has changed .NET development to follow a contemporary model of a community developed software project more closely, including issuing an update to its patent promising to address the concerns.

## **6. TESTING**

- Testing is a vital part of software development, and it is important to start it as early as possible, and to make testing a part of the process of deciding requirements. To get the most useful perspective on your development project, it is worthwhile devoting some thought to the entire lifecycle including how feedback from users will influence the future of the application. The tools and techniques we've discussed in this book should help your team to be more responsive to changes without extra cost, despite the necessarily wide variety of different development processes. Nevertheless, new tools and process improvements should be adopted gradually, assessing the results after each step.
- Testing is part of a lifecycle. The software development lifecycle is one in which you hear of a need, you write some code to fulfil it, and then you check to see whether you have pleased the stakeholders—the users, owners, and other people who have an interest in what the software does. Hopefully they like it, but would also like some additions or changes, so you update or augment your code; and so, the cycle continues. This cycle might happen every few days, as it does in Fabrice's ice cream vending project, or every few years, as it does in Contoso's carefully specified and tested healthcare support system. Software development lifecycle Testing is a proxy for the member. You could conceivably do your testing by releasing it into the wild and waiting for the complaints and compliments to come back. Some companies have been accused of having such a strategy as their business model even before it became fashionable. But overall, the books are better balanced by trying to make sure that the software will satisfy the member before we hand it over. We therefore design tests based on the stakeholders' needs and run the tests before the product reaches the users. Preferably well before then, so as not to waste our time working on something that isn't going to do the job. In this light, two important principles become clear:
- Tests represent requirements. Whether you write user stories on sticky notes on the wall, or use cases in a big thick document, your tests should be derived from and linked to those requirements. And as we've said, devising tests is a good vehicle for discussing the requirements.

- We're not done till the tests pass. The only useful measure of completion is when tests have been performed successfully.

## **SOFTWARE TESTING TYPES**

**BLACK BOX TESTING:** Internal system design is not considered in this type of testing.

Tests are based on requirements and functionality.

**WHITE BOX TESTING:** This testing is based on knowledge of the internal logic of an application's code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

**UNIT TESTING:** Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. may require developing test driver modules or test harnesses.

**INCREMENTAL INTEGRATION TESTING:** Bottom-up approach for testing i.e., continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately. done by programmers or by testers.

**INTEGRATION TESTIN:** Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

**FUNCTIONAL TESTING:** This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black-box type testing geared to functional requirements of an application.

**SYSTEM TESTING:** Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.

**END-TO-END TESTING:** Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate

**SANITY TESTING:** Testing to determine if a new software version is performing well enough to accept it for a major testing effort. If application is crashing for initial use, then system is not stable enough for further testing and build or application is assigned to fix.

**REGRESSION TESTING:** Testing the application for the modification in any module or functionality. Difficult to cover all the system in regression testing so typically automation tools are used for these testing types.

**ACCEPTANCE TESTING:** Normally this type of testing is done to verify if system meets the member specified requirements. User or member do this testing to determine whether to accept application.

**LOAD TESTING:** It's a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.

**STRESS TESTING:** System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.

**PERFORMANCE TESTING:** Term often used interchangeably with 'stress' and 'load' testing. To check whether system meets performance requirements. Used different performance and load tools to do this.

**USABILITY TESTING:** User-friendliness check. Application flow is tested, can new user understand the application easily, Proper help documented whenever user stuck at any point.

**INSTALL/UNINSTALL TESTING:** Tested for full, partial, or upgrade install/uninstall processes on different operating systems under different hardware, software environment.

**RECOVERY TESTIG:** Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

**SECURITY TESTING:** Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

**COMPABILITY TESTING:** Testing how well the software performs in a particular hardware/software/operating system/network environment and different combinations of the above.

**COMPARISON TESTING:** Comparison of product strengths and weaknesses with previous versions or other similar products.

**ALPHA TESTING:** In-house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made because of such testing.

**BETA TESTING:** Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.

## **7. CONCLUSION AND FUTURE ENHANCEMENT**

### **CONCLUSION**

In conclusion, the Gas Agency Management System is a comprehensive solution for managing gas agency operations, including customer management, delivery management, stock management, and financials. The system design phase involved a thorough requirement analysis, system architecture design, database design, user interface design, functional design, security design, integration design, and testing design.

The Gas Agency Management System has several benefits, including improved efficiency, enhanced customer service, better stock management, and financial management. The system also enables the gas agency to generate various reports related to customer details, delivery status, stock management, and financials, which can aid in decision-making.

### **FUTURE ENHANCEMENT**

The Gas Agency Management System can be enhanced in several ways to improve its functionality, usability, and adaptability. Some future enhancement ideas include:

**Mobile Application:** Develop a mobile application that enables customers to place orders, track deliveries, and access their account details.

**Predictive Analytics:** Implement predictive analytics to forecast customer demand, stock requirements, and delivery schedules.

**Integration with IoT Devices:** Integrate the system with IoT devices, such as smart meters, to enable real-time monitoring and control of gas consumption.

**Multi-language Support:** Add multi-language support to the user interface to enable customers from different regions to use the system.


## **8. BIBLIOGRAPHY**

- [tutorialspoint.com/](https://www.tutorialspoint.com/)
- [w3schools.com/asp](https://www.w3schools.com/asp)
- [getbootstrap.com/](https://getbootstrap.com/)
- programmingKnowledge




## 9. APPENDICES A - TABLE STRUCTURE


### Adding customers

|   | Column Name | Data Type      | Allow Nulls              |
|---|-------------|----------------|--------------------------|
|  | ID          | numeric(10, 0) | <input type="checkbox"/> |
|   | FName       | varchar(25)    | <input type="checkbox"/> |
|   | LName       | varchar(10)    | <input type="checkbox"/> |
|   | MobNo       | numeric(10, 0) | <input type="checkbox"/> |
|   | Add1        | varchar(50)    | <input type="checkbox"/> |
|   | Add2        | varchar(50)    | <input type="checkbox"/> |
|   | Pin         | numeric(7, 0)  | <input type="checkbox"/> |
|   | DOC         | date           | <input type="checkbox"/> |
|   | CType       | varchar(50)    | <input type="checkbox"/> |


### Booking

|   | Column Name | Data Type      | Allow Nulls              |
|---|-------------|----------------|--------------------------|
|  | Bid         | varchar(10)    | <input type="checkbox"/> |
|   | ID          | numeric(10, 0) | <input type="checkbox"/> |
|   | DOB         | date           | <input type="checkbox"/> |
|   | Qty         | numeric(1, 0)  | <input type="checkbox"/> |
|   | Price       | numeric(20, 0) | <input type="checkbox"/> |
|   | Cstatus     | varchar(50)    | <input type="checkbox"/> |
|   | CType       | varchar(50)    | <input type="checkbox"/> |
|   |             |                | <input type="checkbox"/> |

**Delivery**

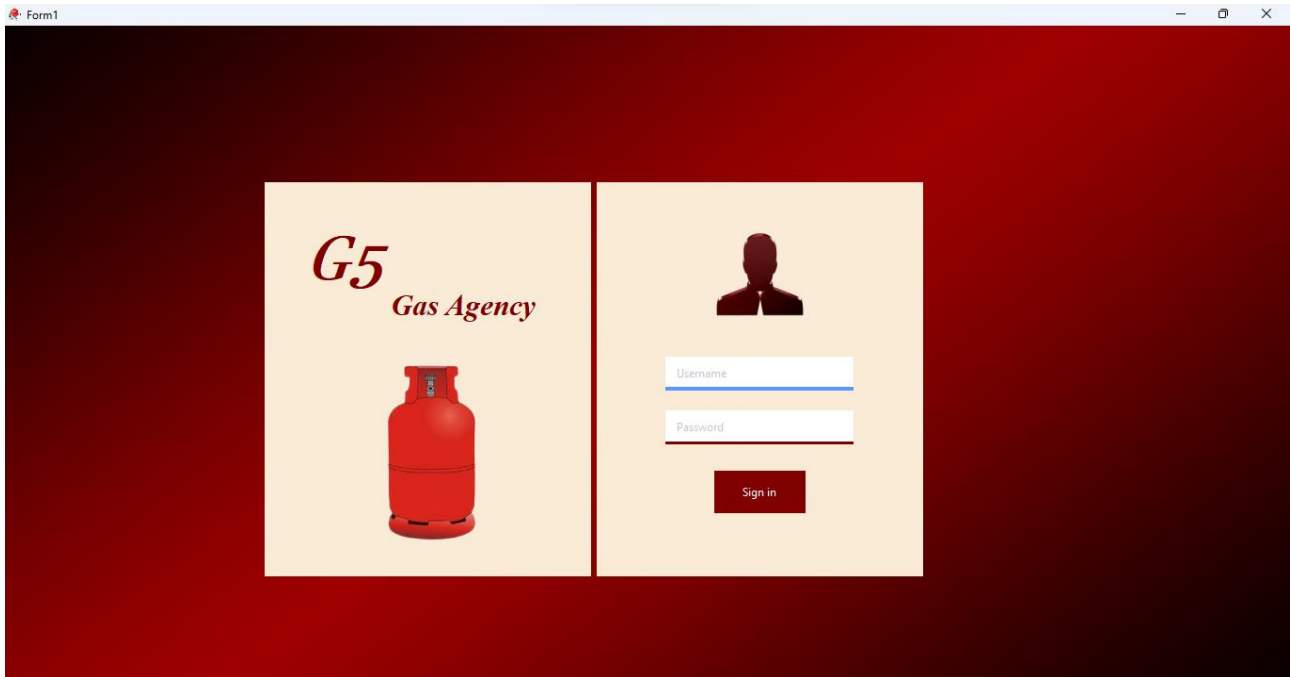
|   | Column Name | Data Type      | Allow Nulls              |
|---|-------------|----------------|--------------------------|
|  | DID         | numeric(18, 0) | <input type="checkbox"/> |
|   | CName       | nchar(10)      | <input type="checkbox"/> |
|   | Dod         | varchar(50)    | <input type="checkbox"/> |
|   | Dadd        | varchar(50)    | <input type="checkbox"/> |
|   | MobNO       | numeric(18, 0) | <input type="checkbox"/> |
|   | Dstat       | nchar(10)      | <input type="checkbox"/> |
|   | Pays        | nchar(10)      | <input type="checkbox"/> |
|   | T_amt       | numeric(18, 0) | <input type="checkbox"/> |

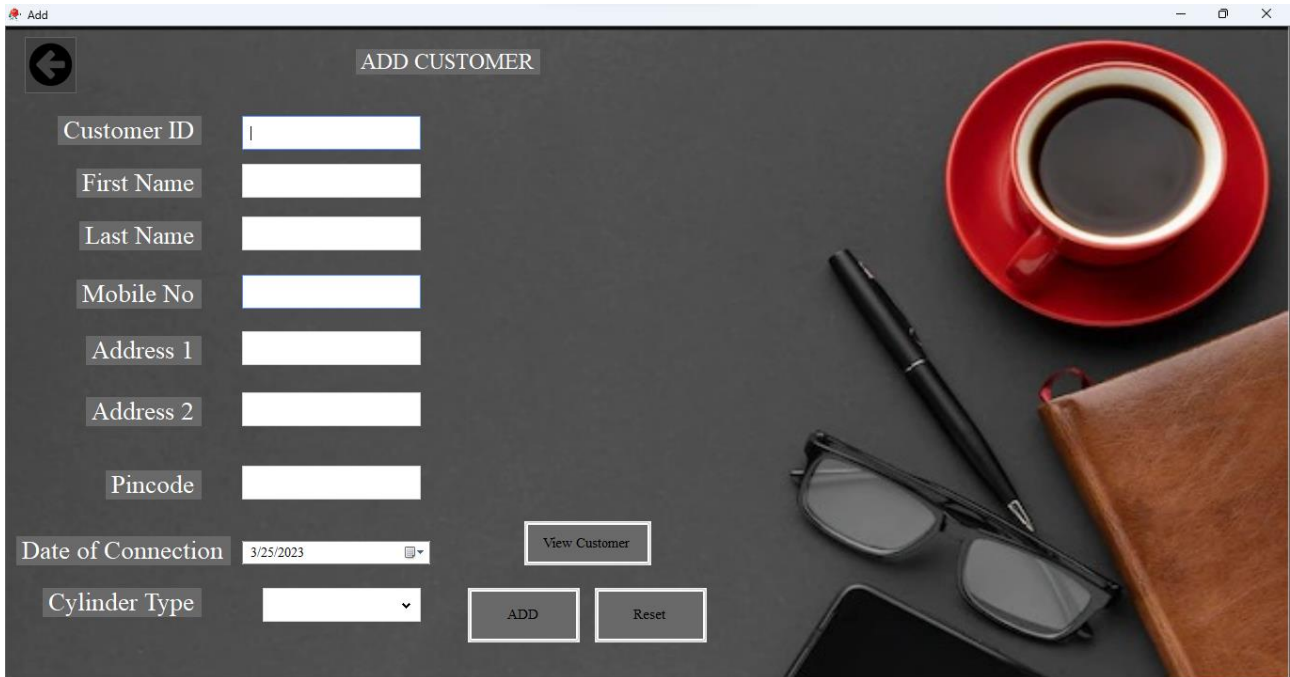
**Stock**

|   | Column Name | Data Type      | Allow Nulls              |
|---|-------------|----------------|--------------------------|
|  | CType       | varchar(50)    | <input type="checkbox"/> |
|   | Qty         | numeric(18, 0) | <input type="checkbox"/> |
|   | Ppc         | numeric(18, 0) | <input type="checkbox"/> |

## 10. APPENDICES B-SCREENSHOTS

### MAIN PAGE





**ADD CUSTOMER**

Customer ID

First Name

Last Name

Mobile No

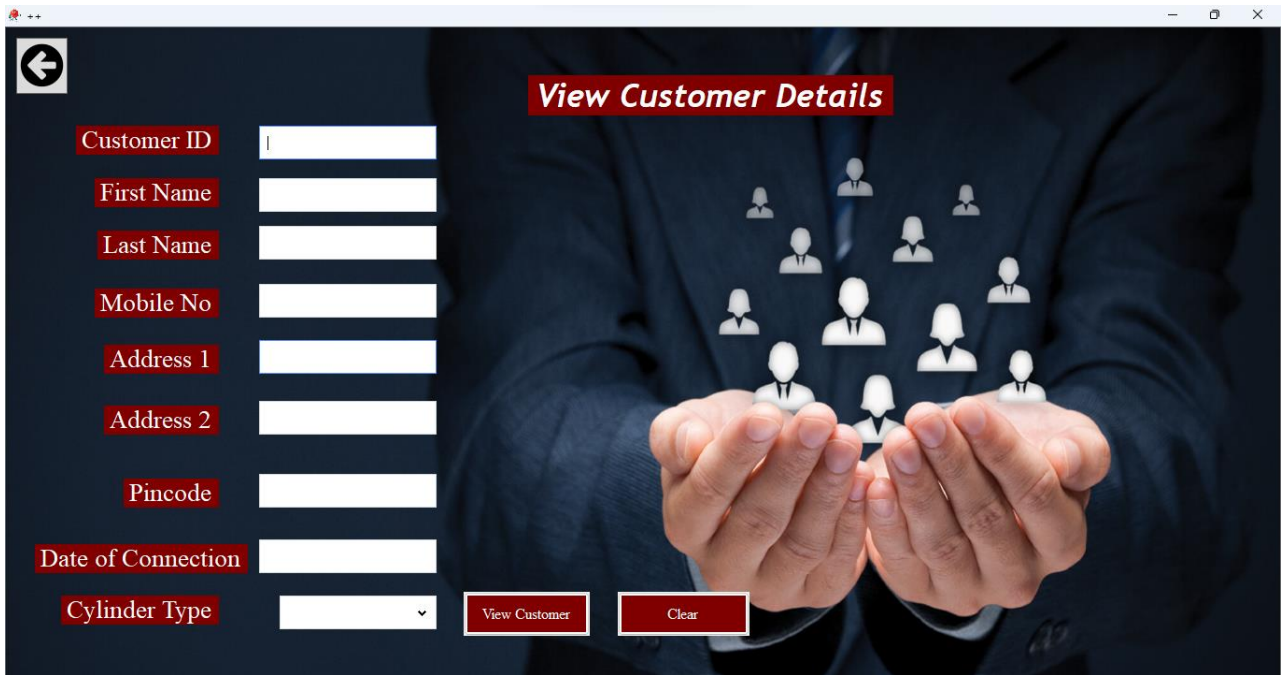
Address 1

Address 2

Pincode

Date of Connection

Cylinder Type



**View Customer Details**

Customer ID

First Name

Last Name

Mobile No

Address 1

Address 2

Pincode

Date of Connection

Cylinder Type

Booking

**← Booking**

*Booking ID*

*Customer ID*

*Date of Booking*

*Quantity*

*Price*

*Cylinder Type*

*Cylinder Status*



Stock

**←**

**Stock Details**

*Cylinder Type*

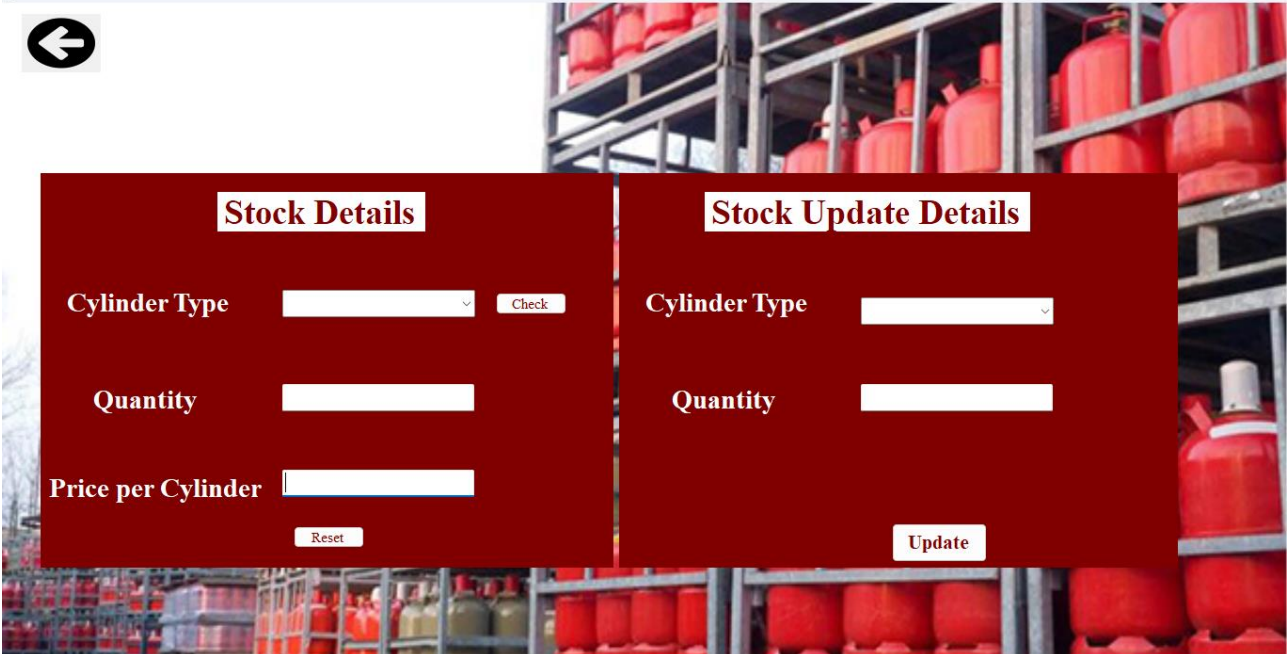
*Quantity*

*Price per Cylinder*

**Stock Update Details**

*Cylinder Type*

*Quantity*



**Delivery Details**

**Delivery ID**

**Customer Name**

**Date of Delivery**

**Address**

**Mobile No**

**Delivery Status**

**Payment Status**

**Proceed** **Clear**

**View**

**View Delivery Details**

**Delivery ID**

**Customer Name**

**Date of Delivery**

**Address**

**Mobile No**

**Delivery Status**

**Payment Status**

**View** **Clear**



The screenshot shows a web application window titled "Report". On the left, there is a "Filters" overlay with a "Customer ID" input field, a date selector set to "Sunday, April 9, 2023", and a "Reset" button. The main area contains a table with the following data:

| ID | DOB       | Qty | Price | CName     | CType      | Dod             | Dstat     | Pays    |
|----|-----------|-----|-------|-----------|------------|-----------------|-----------|---------|
| 11 | 3/1/2023  | 1   | 1100  | Vijay     | Domestic   | Apr 19 2023 ... | Delivered | Pending |
| 62 | 3/26/2023 | 2   | 900   | Vijay     | Commercial | Apr 19 2023 ... | Delivered | Pending |
| 11 | 3/26/2023 | 2   | 1000  | Vijay     | Domestic   | Apr 19 2023 ... | Delivered | Pending |
| 11 | 3/26/2023 | 2   | 1000  | Vijay     | Domestic   | Apr 19 2023 ... | Delivered | Pending |
| 12 | 3/26/2023 | 2   | 1100  | Vijay     | Domestic   | Apr 19 2023 ... | Delivered | Pending |
| 11 | 4/9/2023  | 1   | 1000  | Vijay     | Domestic   | Apr 19 2023 ... | Delivered | Pending |
| 14 | 3/26/2023 | 1   | 900   | Vijay     | Domestic   | Apr 19 2023 ... | Delivered | Pending |
| 19 | 3/29/2023 | 4   | 1100  | Vijay     | Domestic   | Apr 19 2023 ... | Delivered | Pending |
| 54 | 3/26/2023 | 6   | 900   | Vijay     | Commercial | Apr 19 2023 ... | Delivered | Pending |
| 11 | 3/1/2023  | 1   | 1100  | Trisha US | Domestic   | Apr 8 2023 ...  | Delivered | Pending |
| 62 | 3/26/2023 | 2   | 900   | Trisha US | Commercial | Apr 8 2023 ...  | Delivered | Pending |
| 11 | 3/26/2023 | 2   | 1000  | Trisha US | Domestic   | Apr 8 2023 ...  | Delivered | Pending |
| 11 | 3/26/2023 | 2   | 1000  | Trisha US | Domestic   | Apr 8 2023 ...  | Delivered | Pending |

The background of the report window features a blurred image of a calculator and a bar chart with numerical values such as \$600.00, \$462.20, \$366.20, \$240.20, \$284.48, \$316.40, \$197.20, \$116.40, \$197.20, and \$197.20.

**11. APPENDICES C-SAMPLE REPORT OF TEST CASES**

| SL No. | Test case ID                     | Steps to execute  | Test data                       | Expected Result                                  | Actual Result                                    | Status |
|--------|----------------------------------|---|---------------------------------|--|--|--------|
| 1      | TC-1<br>(Field level validation) | Enter the username, password                              | Correct username and password   | Login successful                                 | Login successful                                 | Pass   |
| 2      | TC-2<br>(Form level validation)  | Click on return to Home form                              | It should Come to Home page     | It will take you to the user Home page           | It will take you to the user Home page           | Pass   |
| 3      | TC-3<br>(Data saving validation) | Enter Customers details                                   | Register new Customers details  | 'Customer details successfully saved to database | 'Customer details successfully saved to database | Pass   |
| 4      | TC-4<br>(Field level validation) | Enter the username, password                              | Incorrect username and password | Login unsuccessful                               | Incorrect username and Password                  | Pass   |
| 5      | TC-5<br>(Range validation)       | In Customer registration try entering a 12-digit phone no | Random 12 digits                | Only 10 digits can be entered                    | Only 10 digits can be entered                    | Pass   |
| 6      | TC-6(Masked input validation)    | Enter password in login page                              | Enter some char                 | The password is masked                           | The password is masked                           | Pass   |